Robot Operating System (ROS). Практическое 2

Беляевский Кирилл Олегович

Васильянов Георгий Сергеевич

ВШиСИС, ИКНТ, СПБПУ, 2021

Robot Operating System

Robot Operating System (ROS) - это гибкая платформа (фреймворк) для разработки программного обеспечения роботов. Это набор разнообразных инструментов, библиотек и определенных правил, целью которых является упрощение задач разработки ПО роботов.

Файловая система ROS

- Пакет
  - Пакеты - это программная организационная единица кода ROS.
  - Каждый пакет может содержать библиотеки, исполняемые файлы, сценарии или другие артефакты.
  - Манифест: описание (метаданные) пакета, основная роль которого заключается в определении зависимостей между пакетами (package.xml)
- Мета-пакеты
  - Коллекция пакетов, образующих библиотеку более высокого уровня

Рабочее пространство catkin

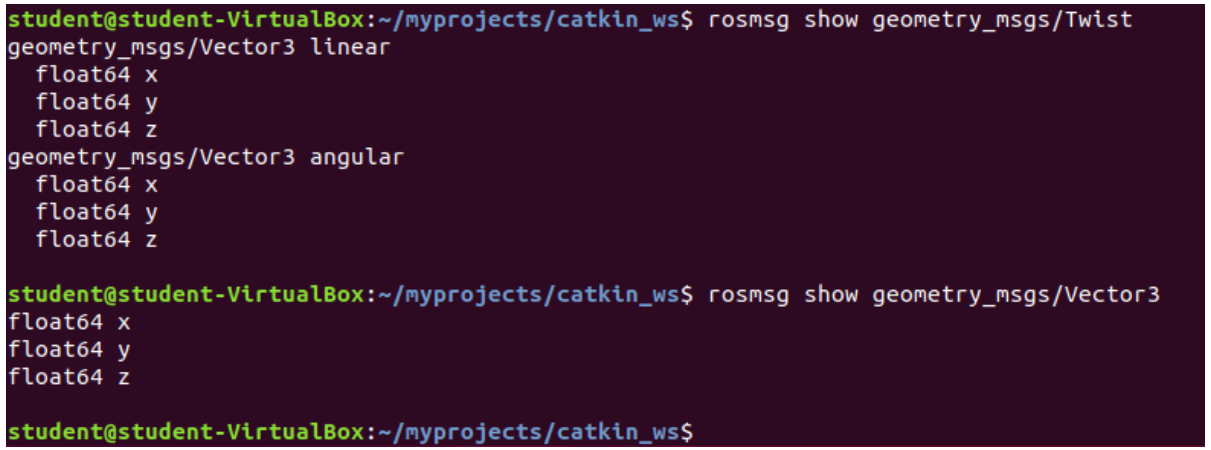
```
workspace_folder/                - Рабочее пространство
  build/                         - Пространство для сборки. CMake вызывается для сборки пакетов catkin
  devel/                         - Здесь размещаются построенные цели перед установкой
  src/                           - Исходники
    CMakeLists.txt               - файл CMake 'Toplevel', предоставленный catkin (не обяз.)
    package_1/
      CMakeLists.txt             - файл CMakeLists.txt для package_1
      package.xml                - манифест пакета для package_1
    ...
    package_n/
      CMakeLists.txt             - файл CMakeLists.txt для package_n
      package.xml                - манифест пакета для package_n
    meta_package/                - коллекции пакетов (подпроект, отдельный проект)
        sub_package_1/
            CMakeLists.txt       - файл CMakeLists.txt для sub_package_1
            package.xml          - манифест пакета для sub_package_1
        …
        sub_package_n/
            CMakeLists.txt       - файл CMakeLists.txt для sub_package_n
            package.xml          - манифест пакета для sub_package_n
      meta_package/
            package.xml          - манифест пакета meta_package
```

Пример пакета «myPkg/»

- **CMakeLists.txt**: настройки сборки CMake для пакета myPkg
- **package.xml**: метаданные и зависимости, необходимые для пакета
- **mainpage.dox**: информация о пакете myPkg
- **include/myPkg**: файлы заголовков c++
- **src/**: каталог исходного кода
- **launch/**: файлы запуска (при необходимости)
- **msg/**: типы сообщений (.msg)
- **srv/**: типы служб (.srv)
- **scripts/**: исполняемые скрипты

Утилиты командной строки Сообщения

- Информация о сообщениях
  - rosmsg list
  - rosmsg show geometry_msgs/Twist
  - rosmsg show geometry_msgs/Vector3



Утилиты командной строки Сервисы

- Информация о сервисах
  - rossrv list
  - rossrv show rosserial_msgs/RequestMessageInfo

```
student@student-VirtualBox:~/myprojects/catkin_ws$ rossrv show rosserial_msgs/RequestMessageInfo
string type
---
string md5
string definition
```

Утилиты командной строки roscore

- **roscore** - это набор узлов и программ, которые являются базовыми для системы на основе ROS. У вас должен быть запущен **roscore**, чтобы узлы ROS могли общаться. Он запускается с помощью команды **roscore**.

```
student@student-VirtualBox:~/myprojects/catkin_ws$ roscore
... logging to /home/student/.ros/log/e8546164-fd2c-11ea-9d82-080027860af9/roslaunch-student-Virtual
Box-8282.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://student-VirtualBox:36567/
ros_comm version 1.14.9


SUMMARY
========

PARAMETERS
 * /rosdistro: melodic
 * /rosversion: 1.14.9

NODES

auto-starting new master
process[master]: started with pid [8292]
ROS_MASTER_URI=http://student-VirtualBox:11311/

setting /run_id to e8546164-fd2c-11ea-9d82-080027860af9
process[rosout-1]: started with pid [8303]
started core service [/rosout]
```

Утилиты командной строки rosrun

- **rosrun** позволяет запускать исполняемый файл в произвольном пакете, не зная его местоположения
- Пример:
  - rosrun cmd_vel_publisher cmd_vel_publisher_node
- Также возможна передача параметров:
  - rosrun package node _parameter:=value
  - rosrun cmd_vel_publisher cmd_vel_publisher_node _Max_Constant_Vel:=0.5

Утилиты командной строки roslaunch

- **roslaunch** позволяет запускать launch файл в произвольном пакете, не зная его местоположения
- lauch-файл обычно вызывают набор узлов для пакета, которые обеспечивают некоторую совокупную функциональность.
- Пример:
  - roslaunch package_name file.launch
- Также возможна передача параметров:
  - roslaunch package_name file.launch _parameter:=value

Утилиты командной строки rosnode

- Текущий список поддерживаемых команд:
  - **rosnode kill** – остановить запущенный узел
  - **rosnode list** – вывести список активных узлов
  - **rosnode machine** – вывести список машин, на которых запущены узлы
  - **rosnode ping** – проверить подключение к узлу
  - **rosnode info** – вывести информацию об узле

Утилиты командной строки rostopic

- Текущий список поддерживаемых команд:
  - **rostopic bw** - отображать полосу пропускания топика
  - **rostopic echo** – вывести сообщение на экран (в консоль)
  - **rostopic find** – найти топик по его типу
  - **rostopic hz** – вывести частоту публикации топика
  - **rostopic info** – вывести информацию о топике
  - **rostopic list** – вывести список активных топиков
  - **rostopic pub** – опубликовать сообщение в топик

http://wiki.ros.org/roslaunch/XML

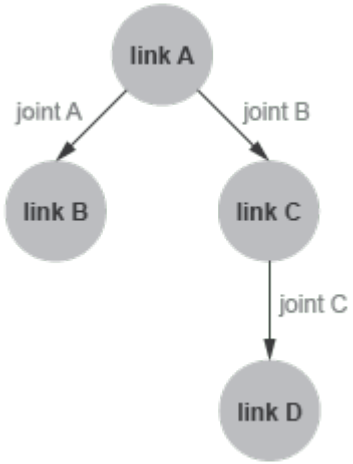
roslaunch
Launch-файл

- Используя roslaunch возможно дополнительно настраивать исполняемые файлы в момент их запуска (передавать параметры, изменять имена и тп)
- roslaunch использует файлы с расширением .launch, которые представляет собой обычный XML файл.
- Элемент `include`
  - Позволяет включить в файл содержимое другого launch файла
- Элемент `param`
  - Определяет параметр для установки в сервере параметров. При использовании `command` результат команды будет загружен в сервер параметров
- Элемент `node`
  - Позволяет запустить узел ROS

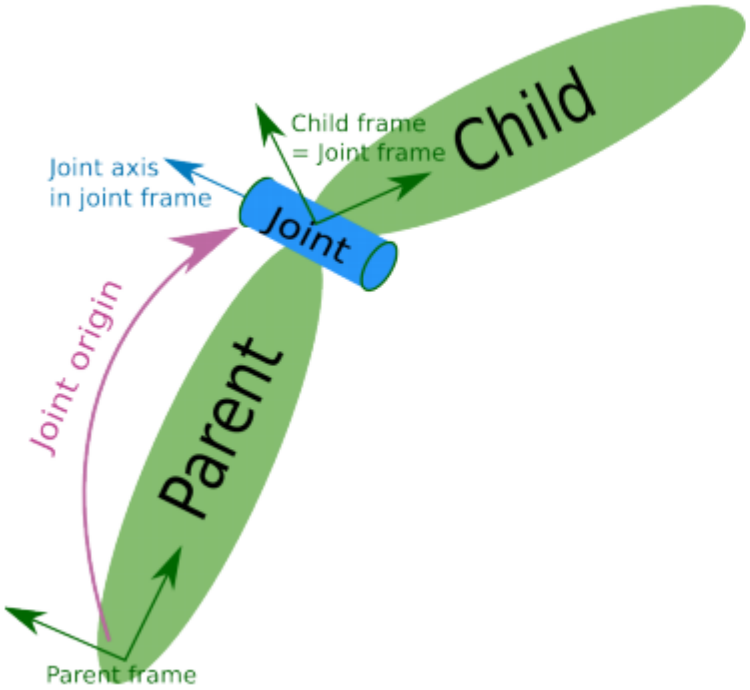
```
<?xml version="1.0"?>
<launch>
    <include file="$(find gazebo_ros)/launch/empty_world.launch"/>
    <param name="robot_description" command="$(find xacro)/xacro '$(find myrobot_simulator)/urdf/myrobot.
gazebo.xacro'" />
    <node pkg="test_package" type="topic_subscriber" name="topic_subscriber1"/>
</launch>
```

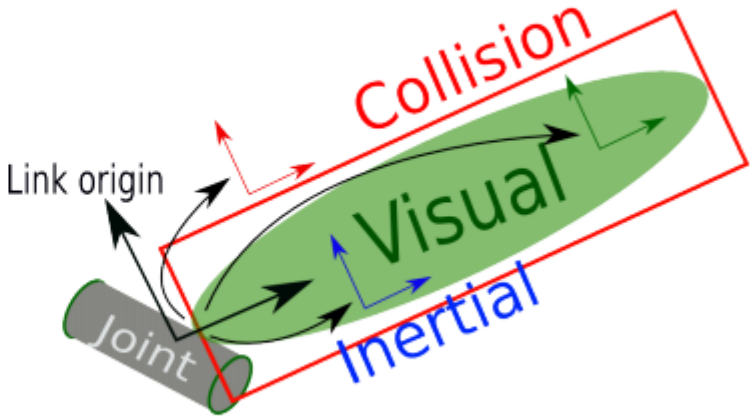
Формат описания URDF

- **Universal Robotic Description Format (URDF)** это формат файла XML, используемый в ROS в качестве собственного формата для описания всех элементов робота. Xacro (XML-macro) - это язык макросов XML. Позволяет делать более короткие и понятные описания роботов.
- Содержит кинематическое и статическое описание робота в древовидной структуре
- Основные элементы:
  - Link – объект
  - Joint – сустав/шарнир



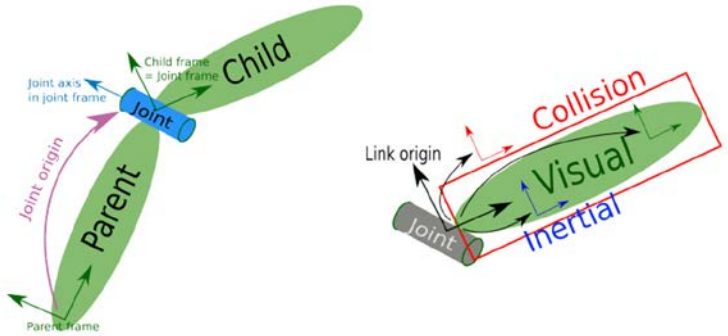
Link и Joint


Child frame
= Joint frame
Child
Joint axis
in joint frame
Joint
Joint origin
Parent
Parent frame


Collision
Link origin
Visual
Joint
Inertial

http://wiki.ros.org/urdf/XML/link

Link



- Атрибуты
  - name
- Элемент **visual**
  - Визуальное описание link'a
  - Может быть несколько подэлементов
    - примитивы геометрии (box, cylinder, sphere)
    - полигональные сетки (ресурсы
    - stl / dae)
  - origin: размещение в системе координат link'a (rpy = вращение)
  - Материал

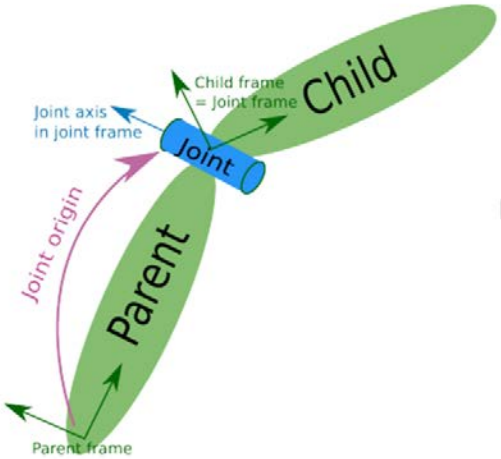
```
<link name="forearm">
    <visual>
        <geometry>
            <origin xyz="0 0 0.1" rpy="0 0 0" />
            <box size="0.1 .2 .5"/>
        </geometry>
        <material name="Cyan">
            <color rgba="0 1.0 1.0 1.0"/>
        </material>
    </visual>
</link>

<link name="gripper">
    <visual>
        <geometry>
         <mesh filename="package://pkg/m.dae"/>
        </geometry>
    </visual>
    <visual>
        <geometry>
            <cylinder length="0.6" radius="0.2"/>
        </geometry>
    </visual>
</link>
```

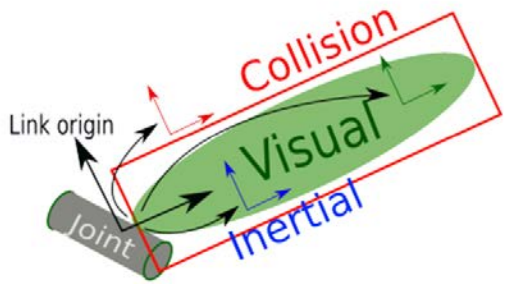
http://wiki.ros.org/urdf/XML/joint

Joint

- Атрибуты
  - name
  - type: continuous , fixed, revolute, prismatic, planar, floating
- Элемент **parent**
- Элемент **child**
- Элемент **origin**
  - в системе координат parent'а
- Элемент **axis**
  - Для prismatic и revolute типов
  - В общей системе координат

```
<joint name="joint1" type="revolute">
    <parent link="forearm"/>
    <child link="gripper"/>
    <origin xyz="0.5 0 0" rpy="0 0 -1.57" />
    <axis xyz="0 0 1" />
</joint>
```

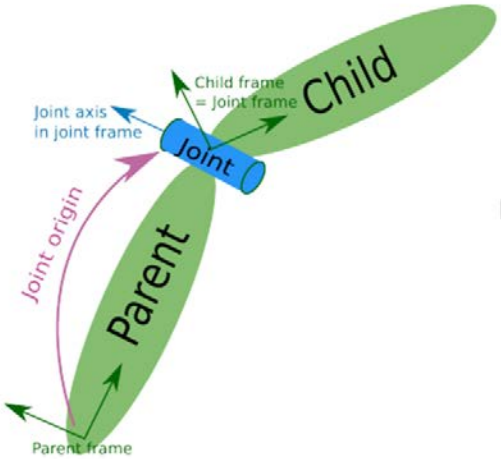


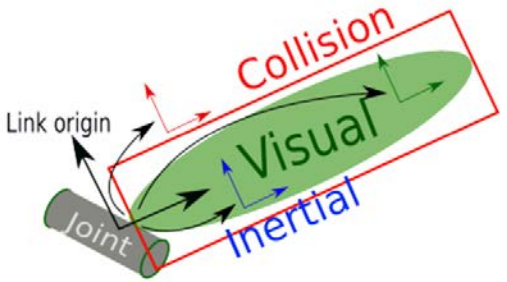


Link collision

- Описывает модель столкновений
- Элемент **collision**
  - Похож на элемент **visual**
  - Может быть несколько подэлементов
    - примитивы геометрии (box, cylinder, sphere)
    - полигональные сетки (ресурсы
    - stl / dae)
  - При использовании сетки – разрешение должно быть низким

```
<collision>
    <geometry>
        <origin xyz="0 0 0.1" rpy="0 0 0"/>
        <mesh filename="package://pkg/x.dae"/>
    </geometry>
</collision>
```

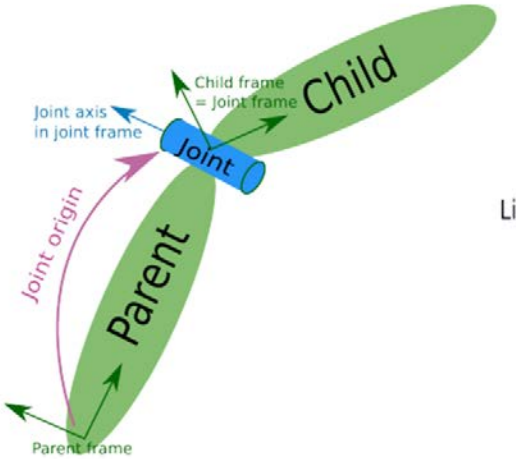


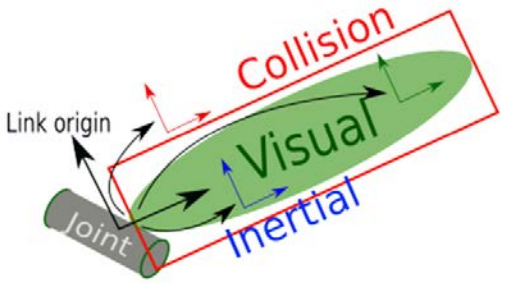


Link inertia

- Описывает инерцию
- Элемент **inertion**
  - Центр масс
  - Масса
  - Матрица инерции

```
<inertial>
    <origin xyz="0.5 0 0" rpy="0 -1.57 0"/>
    <mass value="10"/>
    <inertia ixx="0.4" ixy="0.0" ixz="0.0"
iyy="0.4" iyz="0.0" izz="0.2"/>
</inertial>
```





http://wiki.ros.org/xacro

XACRO

- Макроязык XML, используемый для упрощения URDF
- Увеличивает модульность
- Уменьшить избыточность
- Позволяет параметризацию

XACRO
Базовое использование

- Параметры (xacro:property)
  - Именованные значения, которые можно вставить где угодно в XML-документ.
- В фигурных скобках (${ }) можно также писать простые математические выражения.
- В настоящее время поддерживается простая арифметика и подстановка переменных.

```
<xacro:property name="width" value=".2"/>

<cylinder radius="${width}" length=".1"/>

<link name="${robotname}s_leg" />

<cylinder radius="${diam/2}" length=".1"/>
```

```
<xacro:macro name="default_origin">
    <origin xyz="0 0 0" rpy="0 0 0"/>
</xacro:macro>

<xacro:default_origin />

<xacro:macro name="default_inertial" params="mass">
    <inertial>
        <xacro:default_origin />
        <mass value="${mass}" />
        <inertia ixx="0.4" ixy="0.0" ixz="0.0"
            iyy="0.4" iyz="0.0" izz="0.2"/>
    </inertial>
</xacro:macro>

<xacro:default_inertial mass="10"/>
```

XACRO
Блоки условий

- В Xacro есть условные блоки, похожие на используемые в roslaunch. Это полезно для таких вещей, как настраиваемые роботы или загрузка различных плагинов Gazebo.

```
<xacro:if value="<expression>">
  <... some xml code here ...>
</xacro:if>
<xacro:unless value="<expression>">
  <... some xml code here ...>
</xacro:unless>
```

```
<xacro:property name="var" value="useit"/>
<xacro:if value="${var == 'useit'}"/>
<xacro:if value="${var.startswith('use') and
var.endswith('it')}"/>

<xacro:property name="allowed" value="${[1,2,
3]}"/>
<xacro:if value="${1 in allowed}"/>
```

URDF Макро
Расчет инерции

```
<?xml version="1.0"?>
<robot xmlns:xacro="http://ros.org/wiki/xacro">

<xacro:property name="PI" value="3.1415926535897931"/>

<xacro:macro name="cylinder_inertial" params="radius length mass *origin">
    <inertial>
      <mass value="${mass}"/>
      <xacro:insert_block name="origin"/>
      <inertia ixx="${0.0833333 * mass * (3 * radius * radius + length * leng
th)}" ixy="0.0" ixz="0.0" iyy="${0.0833333 * mass * (3 * radius * radius + le
ngth * length)}" iyz="0.0" izz="${0.5 * mass * radius * radius}"/>
    </inertial>
  </xacro:macro>

  <!-- wheel macro code here (p.24) -->

</robot>
```

myrobot_description/urdf/macro.xacro

Физические параметры инерций объектов

URDF Макро Колеса (ч.1)

```
<xacro:macro name="wheel" params="wheel_prefix parent_link left_right radius width
 *joint_origin">

  <link name="${wheel_prefix}_wheel">
    <xacro:cylinder_inertial length="${width}" mass="${wheel_mass}" radius="${radius}">
      <origin rpy="${PI/2} 0 0" xyz="0 0 0"/>
    </xacro:cylinder_inertial>

    <visual>
      <origin rpy="${left_right * PI/2}  0 0" xyz="0 0 0"/>
      <geometry>
        <mesh filename="package://myrobot_description/meshes/tire.dae" scale="${radius} ${radius} ${width/2}"/>
      </geometry>
    </visual>

    <collision>
      <origin rpy="${PI/2} 0 0" xyz="0 0 0"/>
      <geometry>
        <cylinder length="${width}" radius="${radius}"/>
      </geometry>
    </collision>
  </link>
```

myrobot_description/urdf/macro.xacro

URDF Макро Колеса (ч.2)

```
<joint name="${wheel_prefix}_wheel_joint" type="continuous">
  <parent link="${parent_link}"/>
  <child link="${wheel_prefix}_wheel"/>
  <xacro:insert_block name="joint_origin"/>
  <axis rpy="0 0 0" xyz="0 1 0"/>
</joint>

<transmission name="${wheel_prefix}_wheel_trans" type="SimpleTransmission">
  <type>transmission_interface/SimpleTransmission</type>
  <actuator name="${wheel_prefix}_wheel_motor">
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
  <joint name="${wheel_prefix}_wheel_joint">
    <hardwareInterface>hardware_interface/VelocityJointInterface</hardwareInterface>
  </joint>
</transmission>

</xacro:macro>
```

myrobot_description/urdf/macro.xacro

URDF Описание Параметры

```
<?xml version="1.0"?>
<robot name="myrobot" xmlns:xacro="http://www.ros.org/wiki/xacro">

  <xacro:property name="wheel_radius" value="0.0325"/>
  <xacro:property name="wheel_mass" value="1"/>
  <xacro:property name="wheel_width" value="0.02"/>
  <xacro:property name="wheelbase_length" value="0.16"/>
  <xacro:property name="wheelbase_offset" value="0.0"/>

  <!-- ... -->
```

myrobot_description/urdf/myrobot.xacro (1)

URDF Описание
Шасси и использование макро

```
<!-- ... -->

<xacro:property name="chassis_length" value="0.2"/>
<xacro:property name="chassis_width" value="0.125"/>
<xacro:property name="chassis_height" value="0.04"/>
<xacro:property name="chassis_clearance" value="0.02"/>
<xacro:property name="chassis_mass" value="10"/>

<xacro:include filename="$(find myrobot_description)/urdf/macro.xacro"/>

<!-- ============ BASE =============== -->

<link name='base_link'>
<xacro:make_box_no_inertial sx="${chassis_length}" sy="${chassis_width}" sz="${chassis_height}">
  <origin rpy="0 0 0" xyz="0 0 ${chassis_clearance}"/>
</xacro:make_box_no_inertial>

<xacro:box_inertial sx="${chassis_length}" sy="${chassis_width}" sz="${chassis_height}" mass="${chassis_mass}">
  <origin rpy="0 0 0" xyz="0 0 ${chassis_clearance}"/>
</xacro:box_inertial>
</link>

<!-- ... -->
```

myrobot_description/urdf/myrobot.xacro (2)

URDF Описание
Колеса и использование макро

```
 <!-- ... -->

  <!-- ============ WHEELS =============== -->

  <xacro:wheel wheel_prefix="left" parent_link="base_link" left_right="-
1" radius="${wheel_radius}" width="${wheel_width}">
    <origin xyz=
      "${wheelbase_offset}
       ${wheelbase_length/2}
       ${wheel_radius - chassis_clearance}"/>
  </xacro:wheel>

  <xacro:wheel wheel_prefix="right" parent_link="base_link" left_right="1" radius="${wheel_rad
ius}" width="${wheel_width}">
    <origin xyz=
      "${wheelbase_offset}
       ${-wheelbase_length/2}
       ${wheel_radius - chassis_clearance}"/>
  </xacro:wheel>

</robot>
```

myrobot_description/urdf/myrobot.xacro (3)

Управление Diff Drive Controller

- http://wiki.ros.org/diff_drive_controller

```
# Publish all joint states ----------------------------------
joint_state_controller:
    type: joint_state_controller/JointStateController
    publish_rate: 50

controller:
  type: "diff_drive_controller/DiffDriveController"
  left_wheel: 'left_wheel_joint'
  right_wheel: 'right_wheel_joint'
  pose_covariance_diagonal: [0.001, 0.001, 1000000.0, 1000000.0, 1000000.0, 10
00.0]
  twist_covariance_diagonal: [0.001, 0.001, 1000000.0, 1000000.0, 1000000.0, 1
000.0]
```

myrobot_control/control/diffdrive_controller.yaml

Управление
Launch-файл

```
<?xml version="1.0"?>
<launch>

  <!-- Load joint controller configurations from YAML file to parameter server -->
  <rosparam command="load" file="$(find myrobot_control)/config/diffdrive_controller.yaml"/>
  <param name="/controller/enable_odom_tf" value="true"/>

  <!-- Spawn controller -->
  <node args="/controller /joint_state_controller" name="controller_spawner" output="screen" pkg="contro
ller_manager" type="spawner"/>
  <param command="xacro --
inorder '$(find myrobot_simulator)/urdf/myrobot.gazebo.xacro'" name="robot_description"/>

  <node name="robot_state_publisher" pkg="robot_state_publisher" type="robot_state_publisher"/>

</launch>
```

myrobot_control/launch/control.launch

Симуляция
Параметры робота в симуляторе

myrobot_simulator/urdf/myrobot.gazebo.xacro

```
<?xml version="1.0"?>
<robot name="myrobot" xmlns:xacro="http://ros.org/wiki/xacro">

  <xacro:include filename="$(find myrobot_description)/urdf/myrobot.xacro" />
  <xacro:include filename="$(find myrobot_simulator)/urdf/macro.gazebo.xacro" />

  <!-- Gazebo plugin for ROS Control -->
  <gazebo>
    <plugin name="gazebo_ros_control" filename="libgazebo_ros_control.so"/>
  </gazebo>

</robot>
```

myrobot_simulator/urdf/macro.gazebo.xacro

```
<?xml version="1.0"?>
<robot xmlns:xacro="http://www.ros.org/wiki/xacro">

  <gazebo reference="left_wheel">
    <mu1>4</mu1>
    <mu2>4</mu2>
  </gazebo>

  <gazebo reference="right_wheel">
    <mu1>4</mu1>
    <mu2>4</mu2>
  </gazebo>

  <gazebo reference="base_link">
    <material>Gazebo/Gray</material>
  </gazebo>

</robot>
```

Симуляция
Запуск мира

```
<?xml version="1.0"?>
<launch>
  <include file="$(find gazebo_ros)/launch/empty_world.launch"/>

  <param name="robot_description"
         command="$(find xacro)/xacro '$(find myrobot_simulator)/urdf/myrobot.gazebo.xacro'" />

  <!-- Run a python script to the send a service call to gazebo_ros to spawn a URDF robot -->
  <node name="urdf_spawner" pkg="gazebo_ros" type="spawn_model" respawn="false" output="screen"
        args="-urdf -model myrobot -param robot_description -x -3.0 -y -1.5 -z 2.719 -R 0 -P 0 -Y -0"/>

</launch>
```

myrobot_simulator/launch/gazebo_testwalls.la
unch

rviz
Запуск

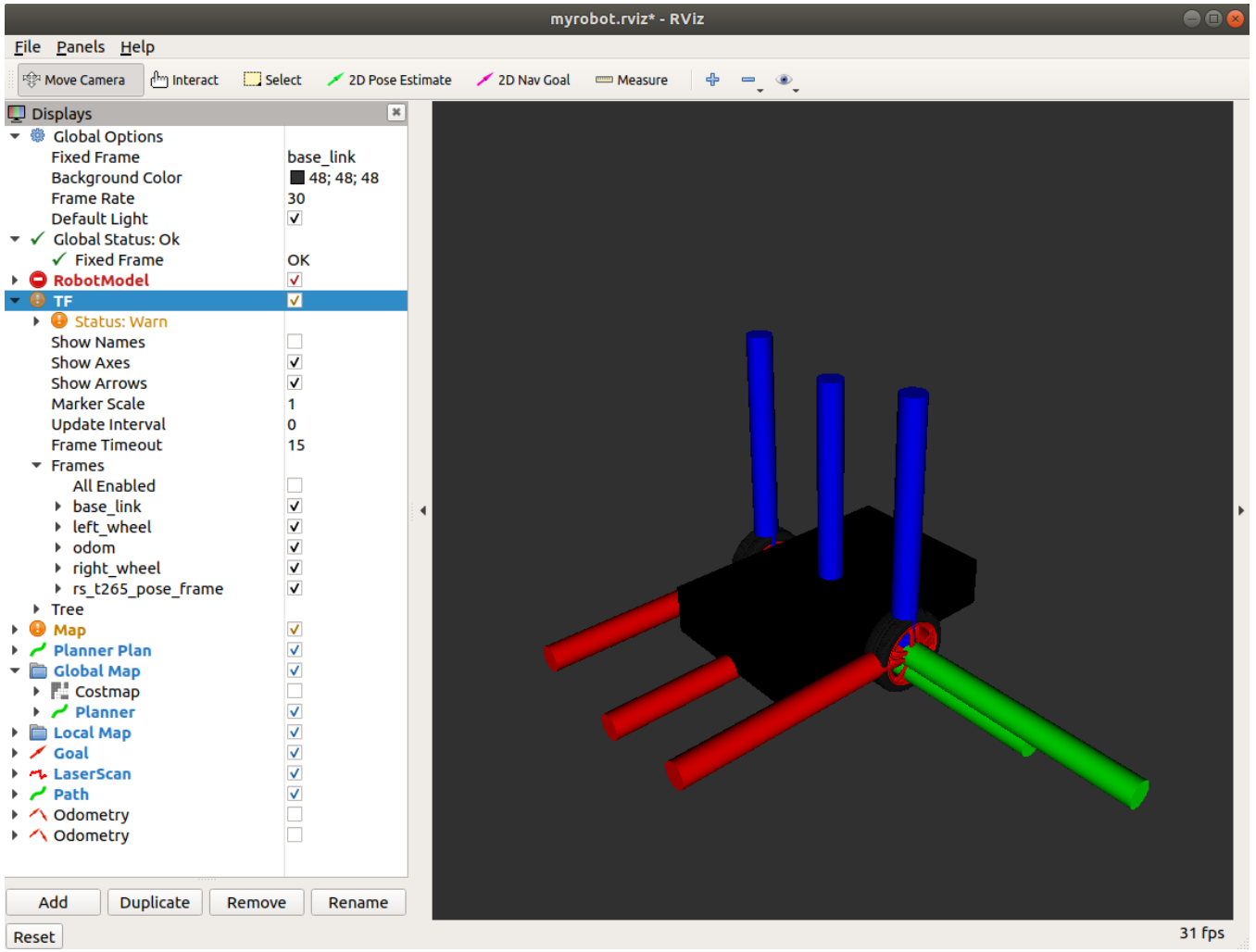
```
<?xml version="1.0"?>
<launch>

  <arg name="config" default="myrobot"/>
  <node pkg="rviz" type="rviz" name="rviz"
          args="-d $(find myrobot_description)/rviz/$(arg config).rviz"/>

</launch>
```

myrobot_description/launch/rviz.launch

Задача

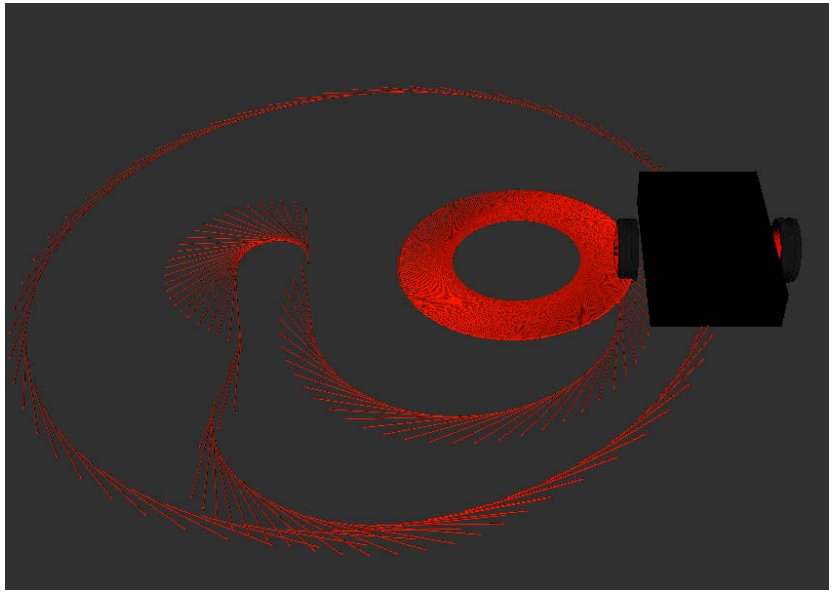
- Создать URDF описание простого колёсного робота. При этом я вас не ограничиваю в том, как это сделать и что вы будете использовать. Хотите – используйте код, который есть в презентации, хотите – напишите без macro. При этом есть 3 условия: диаметр колёс робота должен быть меньше длины робота + колеса должно быть 2 + линейные размеры робота не должны превышать 0,5м;
- Связать его с Gazebo
- Запустить созданного робота

rviz


myrobot.rviz* - RViz
File Panels Help
Move Camera
Interact
Select
2D Pose Estimate
2D Nav Goal
Measure
Displays
Global Options
Fixed Frame
base_link
Background Color
48; 48; 48
Frame Rate
30
Default Light
Global Status: Ok
Fixed Frame
OK
RobotModel
TF
Status: Warn
Show Names
Show Axes
Show Arrows
Marker Scale
1
Update Interval
0
Frame Timeout
15
Frames
All Enabled
base_link
left_wheel
odom
right_wheel
rs_t265_pose_frame
Tree
Map
Planner Plan
Global Map
Costmap
Planner
Local Map
Goal
LaserScan
Path
Odometry
Odometry
Add
Duplicate
Remove
Rename
Reset
31 fps

Запуск робота

```
# Консоль 1 - контроллер
roslaunch myrobot_control control.launch

# Консоль 2 - симулятор
roslaunch myrobot_simulator gazebo_testwalls.la
unch

# Консоль 3 - rviz
roslaunch myrobot_description rviz.launch

# Консоль 4 - rqt
rqt
```



Спасибо за внимание!