

Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и кибербезопасности
Высшая школа компьютерных технологий и информационных систем

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

Дисциплина: Программное обеспечение встраиваемых систем

Тема: Разработка описания робота

Выполнил
студент гр. 5140901/21501

<подпись>

А.М. Кобыжев

Преподаватель

<подпись>

Г.С. Васильянов

« ____ » _____ 2023 г.

Санкт-Петербург

2023

СОДЕРЖАНИЕ

1.	Цели работы	3
2.	Задание.....	3
3.	Ход работы	3
3.1.	Описание робота	4
3.2.	Запуск робота	7
3.2.1.	Запуск контроллера	7
3.2.2.	Запуск симулятора.....	7
3.2.3.	Запуск rviz	8
3.2.4.	Запуск панелей управления роботом.....	9
4.	Выводы	11

1. ЦЕЛИ РАБОТЫ

Разработка описания робота на языке URDF.

2. ЗАДАНИЕ

Вам дан усечённый проект колёсного робота со всеми основными файлами для запуска.

Изучить макросы в файле `myrobot_description/urdf/macro.xacro`.

В файле `myrobot_description/urdf/myrobot.xacro` расположена заготовка робота, в которой вам необходимо создать:

- Корпус робота с названием `base_link`;
- Колёса робота;
- Сенсоры будущего робота – лидар и ик-сенсоры.

Перед запуском симуляции, не забудьте проверить следующее:

- Что имена ваших `link`'ов совпадают с теми, что существуют в файлах `myrobot_simulator/urdf/macro.gazebo.xacro` и `myrobot_control/config/diffdrive_controller.yaml`;
- Что все `joint` и `link` имеют иерархичную структуру.

Таким образом, вам необходимо:

- Создать URDF описание простого колёсного робота:
 - диаметр колёс робота должен быть меньше длины робота;
 - колес должно быть как минимум 2;
 - линейные размеры робота не должны превышать 0,5м.
- Связать его с Gazebo
- Запустить созданного робота

3. ХОД РАБОТЫ

Репозиторий с исходным кодом для данной лабораторной работы можно посмотреть по ссылке:

https://github.com/alexneveskiy/urdf_labs/tree/lab2

3.1. Описание робота

Описание простого двухколёсного робота на языке URDF представлено в листинг 3.1. Весь код можно разбить на два условных блока: объявление параметров и описание частей робота.

В первом блоке описываются все необходимые для описания робота переменные, а именно:

- Начальная позиция;
- Габариты тела робота (параллелепипед);
- Габариты колёс и их позиция относительно тела робота;
- Габариты лидара и его позиция относительно тела робота;
- Габариты ик-сенсоров и их позиции относительно тела робота.

Во втором блоке описываются части робота: тело, колёса, лидар и ик-сенсоры.

Для создания тела робота использовался заготовленный макрос *make_box*, который строит параллелепипед с указанными параметрами. Тело робота получилось длиной 21 сантиметр, шириной 16 сантиметров и высотой 5 сантиметров.

Для создания колёс также использовался заранее заготовленный макрос *wheel*, в котором потребовалось добавить параметр массы из-за его отсутствия в виде атрибута, хотя в теле макроса он использовался. Всего у робота имеется два колеса, шасси которых располагается по середине тела. Ширина каждого колеса составляет 2 сантиметра, а радиус 3 сантиметра. Данные габариты укладываются в заданные рамки, описанные ранее.

Для создания лидара написан *link* с заданными габаритами и коллизией, а также *joint* с фиксированным типом, где родителем является тело робота.

Для создания трёх ик-сенсоров использовался также ранее заготовленный макрос *ir_sensor* с заданными параметрами. Два ик-сенсора располагаются спереди по краям робота и повёрнуты на 45, а третий – по центру тела робота и направлен прямо по оси X.

Листинг 3.1. Описание робота

```

<?xml version="1.0"?>
<robot name="myrobot" xmlns:xacro="http://www.ros.org/wiki/xacro">
  <!-- ===== Enter xacro properties here (if you want) ===== -->

  <xacro:property name="start_x" value="0.0"/>
  <xacro:property name="start_y" value="0.0"/>
  <xacro:property name="start_z" value="0.03"/>

  <xacro:property name="body_name" value="body_name"/>
  <xacro:property name="body_sx" value="0.21"/>
  <xacro:property name="body_sy" value="0.16"/>
  <xacro:property name="body_sz" value="0.05"/>
  <xacro:property name="body_mass" value="5"/>

  <xacro:property name="wheel_radius" value="0.03"/>
  <xacro:property name="wheel_width" value="0.02"/>
  <xacro:property name="wheel_spacer" value="0.005"/>
  <xacro:property name="wheelbase_x" value="0"/>
  <xacro:property name="wheelbase_y" value="${body_sy / 2 + wheel_width / 2 +
wheel_spacer}"/>
  <xacro:property name="wheelbase_z" value="${-body_sz / 4}"/>
  <xacro:property name="wheel_mass" value="1"/>

  <xacro:property name="lidar_rad" value="0.001"/>
  <xacro:property name="lidar_width" value="0.001"/>
  <xacro:property name="lidar_x" value="0"/>
  <xacro:property name="lidar_y" value="0"/>
  <xacro:property name="lidar_z" value="${body_sz / 2}"/>
  <xacro:property name="lidar_pos_z" value="${start_z + 0.001}"/>

  <xacro:property name="ir_sx" value="0.003"/>
  <xacro:property name="ir_sy" value="0.008"/>
  <xacro:property name="ir_sz" value="0.008"/>
  <xacro:property name="ir_x" value="${body_sx / 2}"/>
  <xacro:property name="ir_y" value="${body_sy / 2}"/>
  <xacro:property name="ir_z" value="${-body_sz / 2}"/>

  <!-- ===== Including macros and materials ===== -->

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

  <!-- ===== Some magic. Do not touch ===== -->

  <link name="rs_t265_pose_frame">
  </link>

  <joint name="rs_t265_joint" type="fixed">
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <parent link="rs_t265_pose_frame"/>
    <child link="base_link"/>
  </joint>

  <!-- ===== Enter base_link code here ===== -->

  <link name="base_link">
    <xacro:make_box sx="${body_sx}" sy="${body_sy}" sz="${body_sz}" mass="${body_mass}"
xyz="${start_x} ${start_y} ${start_z}"/>
  </link>

  <!-- ===== Enter wheels code here ===== -->

  <xacro:wheel wheel_prefix="left" parent_link="base_link" left_right="-1"
radius="${wheel_radius}" width="${wheel_width}" wheel_mass="${wheel_mass}">

```

```

    <origin xyz="${start_x + wheelbase_x} ${start_y + wheelbase_y} ${start_z +
wheelbase_z}"/>
  </xacro:wheel>

  <xacro:wheel wheel_prefix="right" parent_link="base_link" left_right="1"
radius="${wheel_radius}" width="${wheel_width}" wheel_mass="${wheel_mass}">
    <origin xyz="${start_x + wheelbase_x} ${start_y - wheelbase_y} ${start_z +
wheelbase_z}"/>
  </xacro:wheel>

  <!-- ===== Enter sensors code here ===== -->
  <!-- ===== LIDAR ===== -->
  <link name="rplidar_a2_frame">
    <visual>
      <origin rpy="0 0 0" xyz="0 0 ${lidar_pos_z}"/>
      <geometry>
        <mesh filename="package://myrobot_description/meshes/rplidar.dae"
scale="${lidar_rad} ${lidar_rad} ${lidar_width}"/>
      </geometry>
    </visual>
    <collision>
      <origin rpy="0 0 0" xyz="0 0 ${lidar_pos_z}"/>
      <geometry>
        <cylinder length="${lidar_width}" radius="${lidar_rad}"/>
      </geometry>
    </collision>
  </link>

  <joint name="rplidar_a2_joint" type="fixed">
    <axis xyz="0 1 0"/>
    <origin rpy="0 0 ${PI}" xyz="${lidar_x} ${lidar_y} ${lidar_z + 0.003}"/>
    <parent link="base_link"/>
    <child link="rplidar_a2_frame"/>
  </joint>

  <!-- ===== IR sensors ===== -->

  <xacro:ir_sensor name="front_0" parent="base_link" sx="${ir_sx}" sy="${ir_sy}"
sz="${ir_sz}">
    <origin rpy="0 0 ${45 * PI / 180}"
xyz="${start_x + ir_x} ${start_y + ir_y} ${start_z + ir_z}"/>
  </xacro:ir_sensor>

  <xacro:ir_sensor name="front_1" parent="base_link" sx="${ir_sx}" sy="${ir_sy}"
sz="${ir_sz}">
    <origin rpy="0 0 0"
xyz="${start_x + ir_x} ${start_y} ${start_z + ir_z}"/>
  </xacro:ir_sensor>

  <xacro:ir_sensor name="front_2" parent="base_link" sx="${ir_sx}" sy="${ir_sy}"
sz="${ir_sz}">
    <origin rpy="0 0 ${-45 * PI / 180}"
xyz="${start_x + ir_x} ${start_y - ir_y} ${start_z + ir_z}"/>
  </xacro:ir_sensor>

</robot>

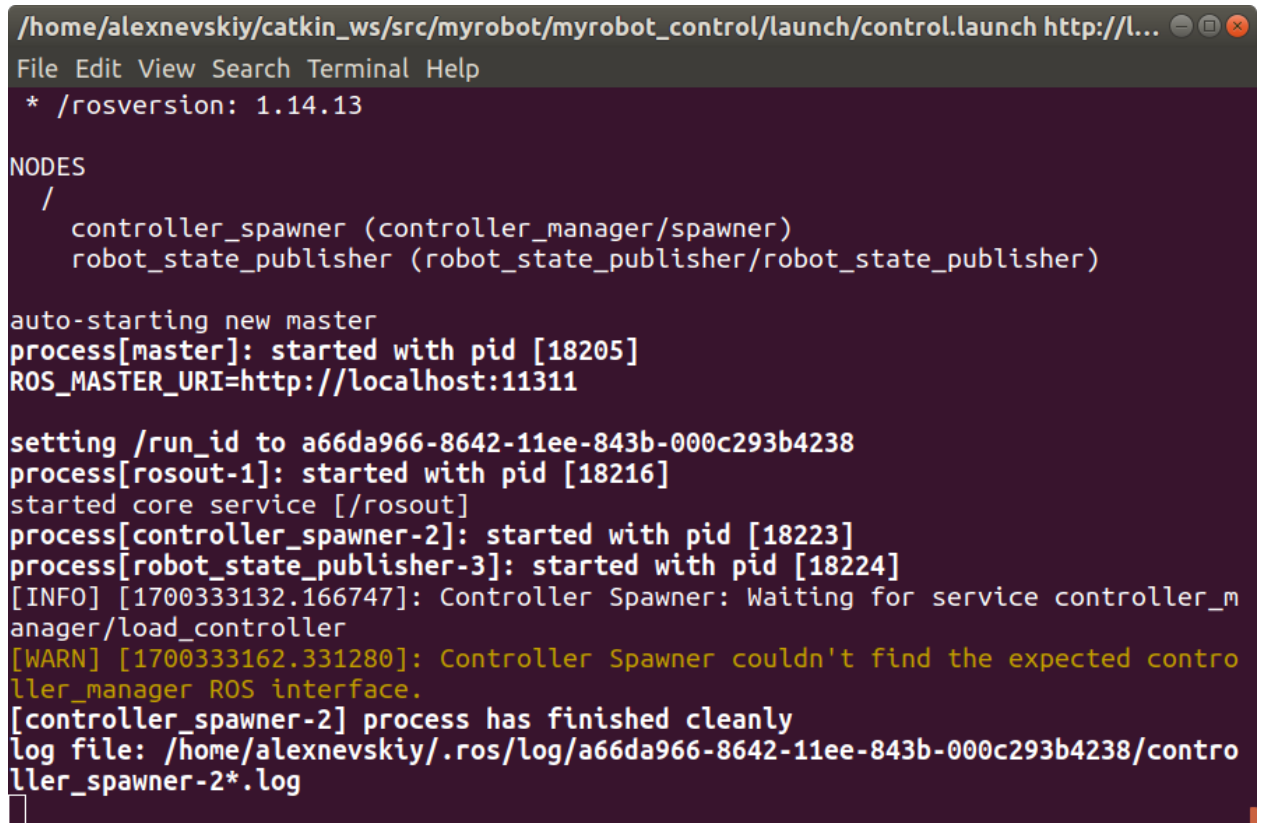
```

Имена всех описанных ранее link'ов совпадают с теми, что существуют в файлах myrobot_simulator/urdf/macro.gazebo.xacro и myrobot_control/config/diffdrive_controller.yaml.

3.2. Запуск работа

3.2.1. Запуск контроллера

Для запуска контроллера, который отвечает за управление роботом, необходимо ввести команду *roslaunch myrobot_control control.launch*. Запуск контроллера в терминале представлен на рис. 3.1.



```

/home/alexnevskiy/catkin_ws/src/myrobot/myrobot_control/launch/control.launch http://l...
File Edit View Search Terminal Help
* /rosversion: 1.14.13

NODES
/
  controller_spawner (controller_manager/spawner)
  robot_state_publisher (robot_state_publisher/robot_state_publisher)

auto-starting new master
process[master]: started with pid [18205]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to a66da966-8642-11ee-843b-000c293b4238
process[rosout-1]: started with pid [18216]
started core service [/rosout]
process[controller_spawner-2]: started with pid [18223]
process[robot_state_publisher-3]: started with pid [18224]
[INFO] [1700333132.166747]: Controller Spawner: Waiting for service controller_m
anager/load_controller
[WARN] [1700333162.331280]: Controller Spawner couldn't find the expected contro
ller_manager ROS interface.
[controller_spawner-2] process has finished cleanly
log file: /home/alexnevskiy/.ros/log/a66da966-8642-11ee-843b-000c293b4238/contro
ller_spawner-2*.log

```

Рис. 3.1. Окно терминала с контроллером

3.2.2. Запуск симулятора

Для запуска симулятора gazebo, в котором будет происходить симуляция описанного ранее робота в заданном помещении, необходимо ввести команду *roslaunch myrobot_simulator gazebo_testwalls.launch*. По окончании запуска симулятора открывается окно gazebo, где будет представлен описанный ранее робот, как показано на рис. 3.2.



Рис. 3.2. Окно gazebo

3.2.3. Запуск rviz

Для запуска rviz, в котором будет отображён описанный ранее робот с предоставленным описанием, необходимо ввести команду `roslaunch myrobot_description rviz.launch`. Окно rviz с роботом представлено на рис. 3.3. Красными точками помечены контуры помещения при помощи лидара.

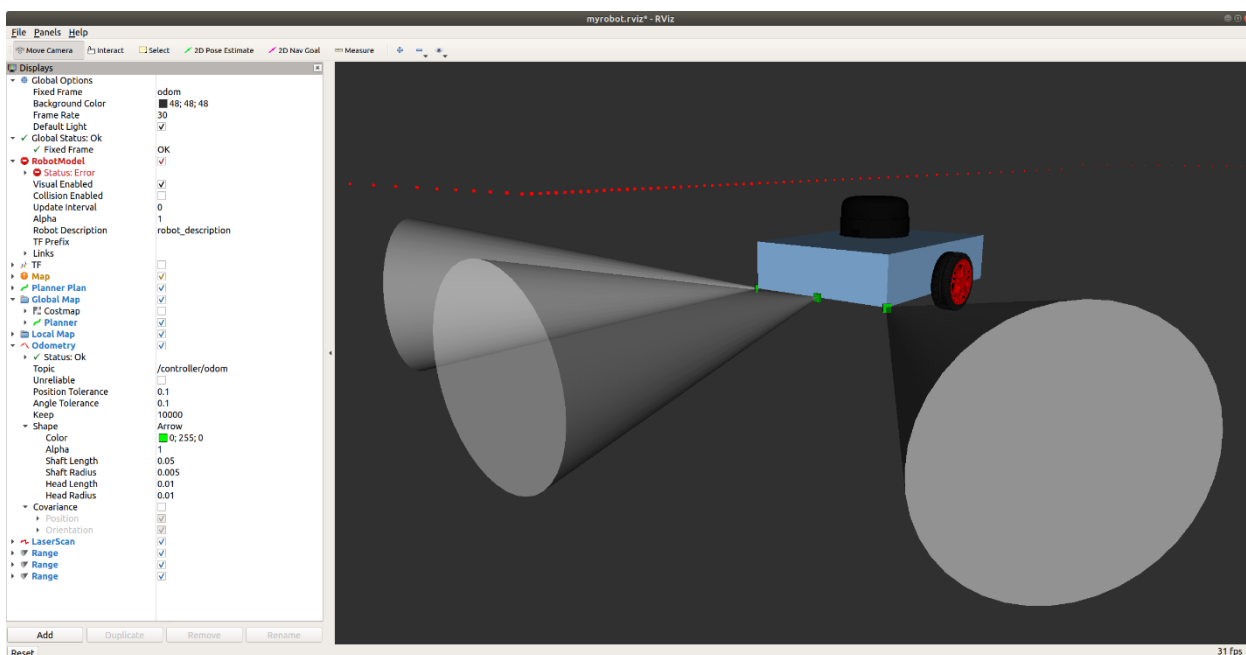


Рис. 3.3. Окно rviz

3.2.4. Запуск панелей управления роботом

Для запуска панели, откуда будет производиться управление роботом, необходимо запустить утилиту *rqt*. Окно *rqt* с плагином Robot Steering, позволяющего управлять роботом, представлено на рис. 3.4. Для управления созданным роботом необходимо указать топик `/controller/cmd_vel`.

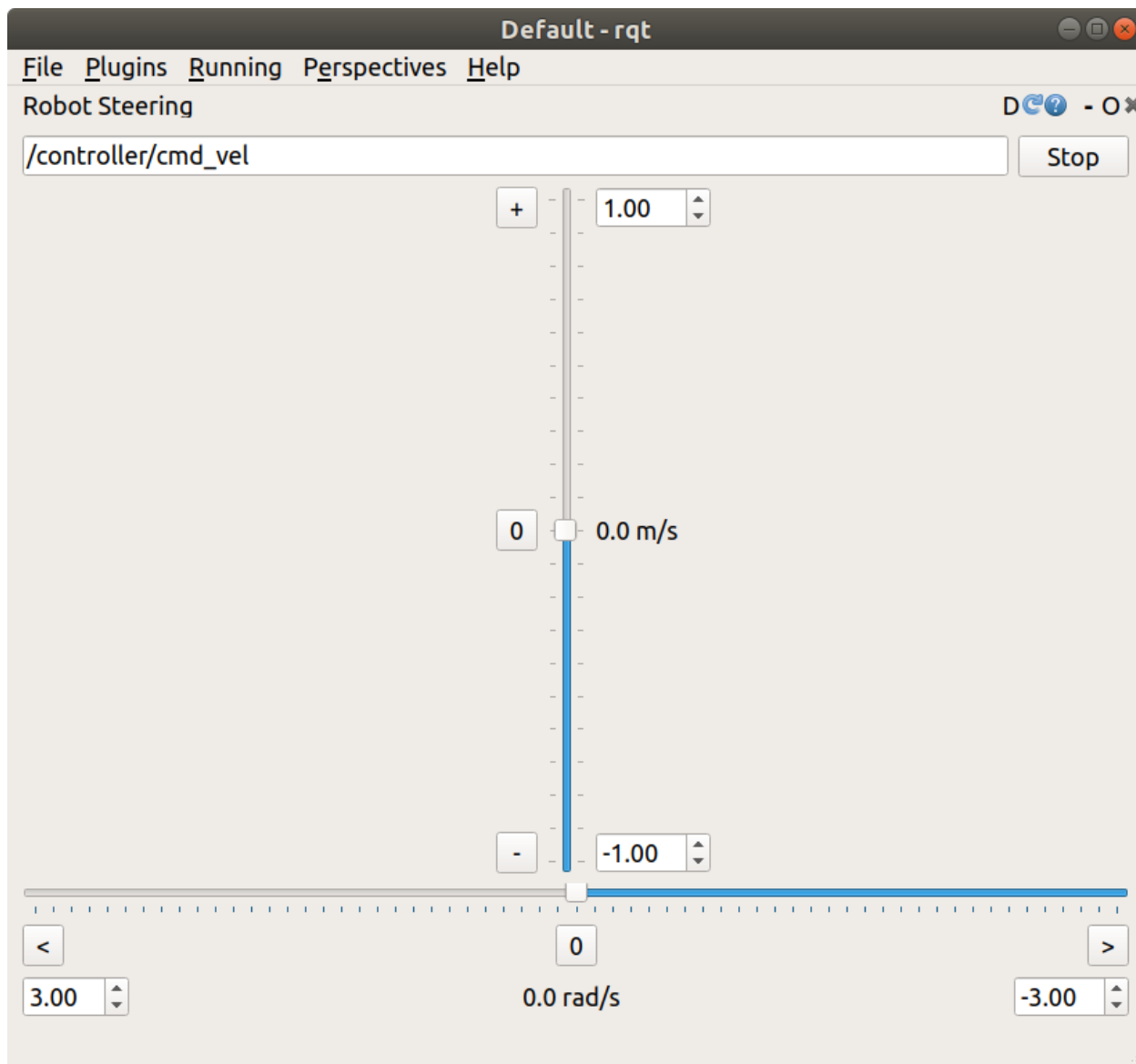


Рис. 3.4. Окно *rqt* с плагином Robot Steering для управления роботом

При помощи данного плагина можно управлять роботом как в *rviz*, так и в *gazebo*, так как они имеют одно и то же помещение в симуляции. Пример управления роботом при помощи плагина в *rviz* представлен на рис. 3.5.

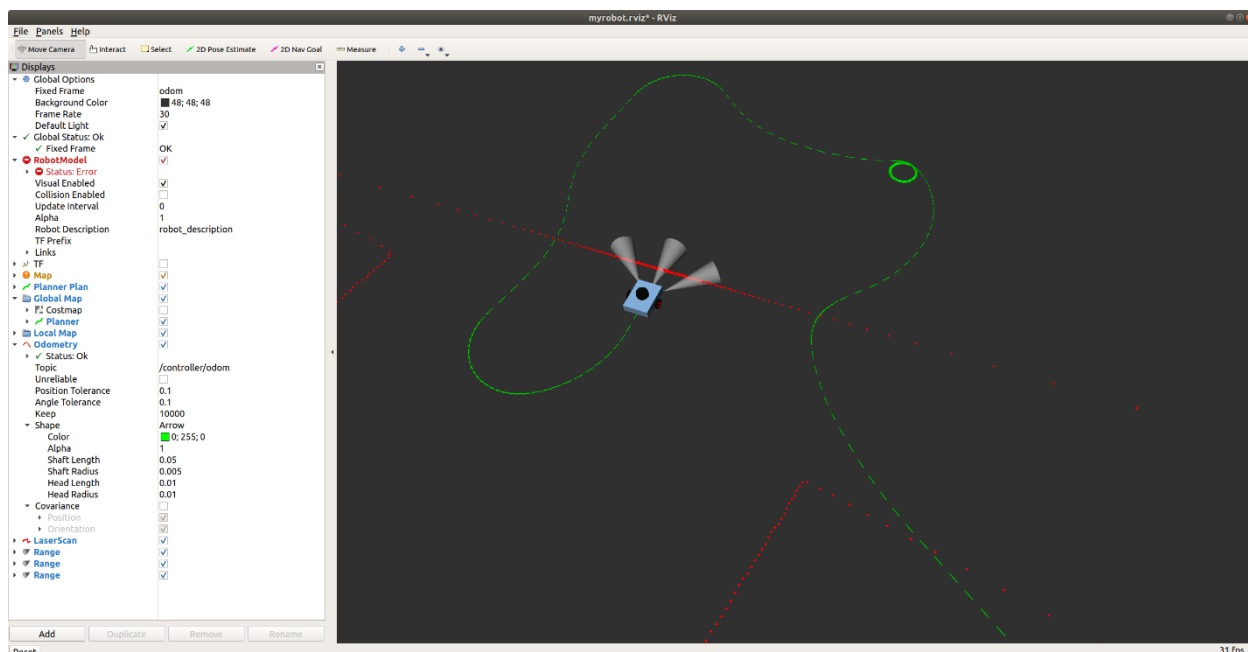


Рис. 3.5. Окно rviz с траекторией движения робота

Пример управления роботом при помощи управления плагина в gazebo представлен на рис. 3.6. Из рисунка видно, что робот находится на том же месте в помещении, что и в rviz.

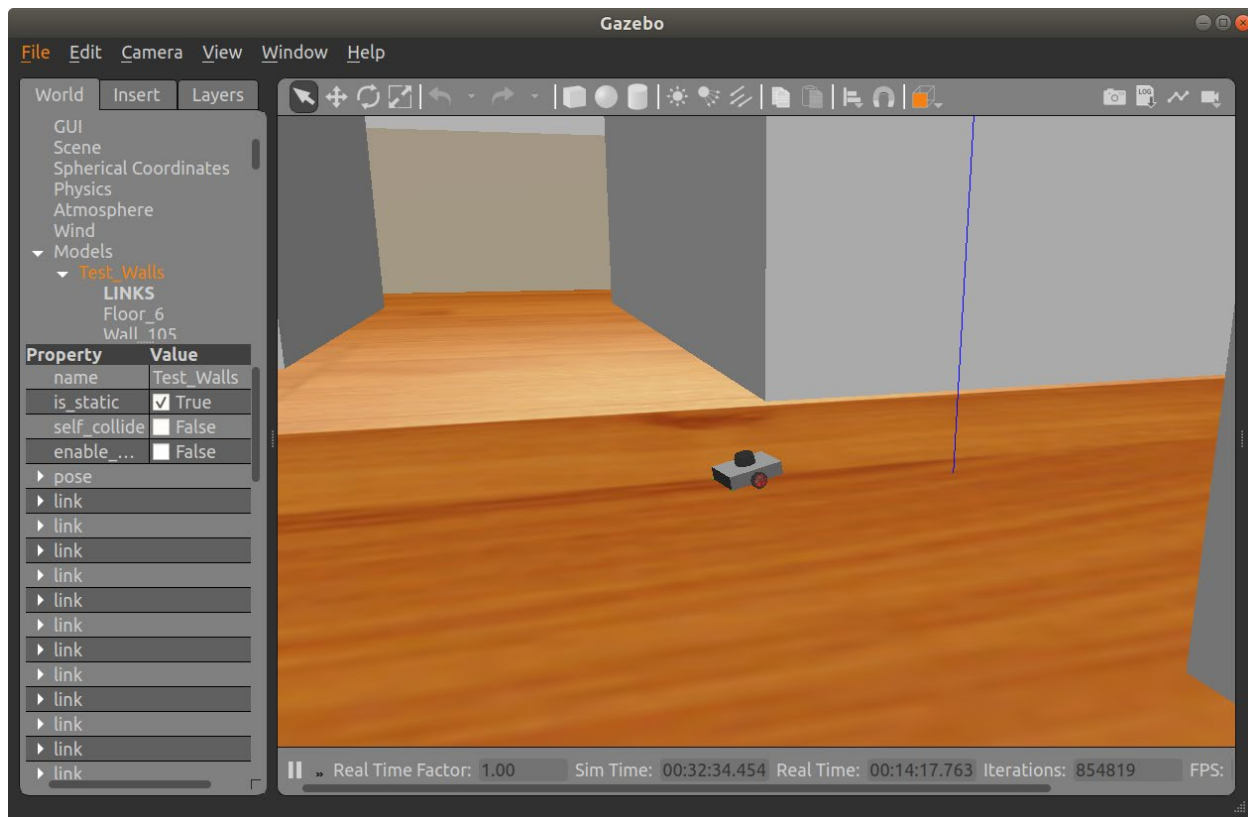


Рис. 3.6. Окно gazebo со смещённым роботом

4. ВЫВОДЫ

В ходе выполнения данной лабораторной работы произведено описание двухколёсного робота с лидаром и тремя ик-сенсорами на языке URDF. Созданный робот соответствует всем критериям, описанных в задании. Также произведён успешный запуск робота в симуляции gazebo и rviz. Управление роботом производилось при помощи плагина Robot Steering.