

Санкт-Петербургский политехнический университет Петра Великого  
Институт компьютерных наук и технологий  
Высшая школа интеллектуальных систем и суперкомпьютерных технологий

Лабораторная работа  
Предмет: Программное обеспечение встраиваемых систем  
Тема: Robot Operating System

Студенты группы № 3540901/02001:

Бараев Д. Р.  
Дроздов Н. Д.  
Клюев А. М.  
Чёрный В. Г.

Преподаватель: Васильянов Г. С.

Санкт-Петербург  
2021

## Оглавление

1.	Справочная информация.....	3
1.1.	Описание платформы.....	3
2.	Цель работы .....	3
3.	Ход работы .....	3
3.1.	Демонстрационный проект turtlesim .....	3
3.1.1.	Установка демонстрационного проекта .....	3
3.1.2.	Запуск демонстрационного проекта.....	4
3.2.	Колёсный робот .....	6
3.2.1.	Создание описаний .....	6
3.2.2.	Управление роботом .....	9
3.2.3.	Запуск робота.....	10
4.	Выводы .....	12

## 1. Справочная информация

### 1.1. Описание платформы

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

## 2. Цель работы

Целью работы является создание описания колёсного робота, имеющего 4 колеса.

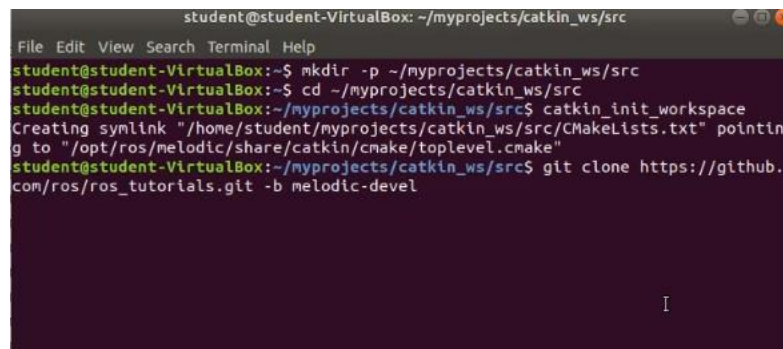
Запуск робота из описания будет производиться в симуляторе Gazebo, управление будет производиться с клавиатуры компьютера.

## 3. Ход работы

### 3.1. Демонстрационный проект turtlesim

#### 3.1.1. Установка демонстрационного проекта

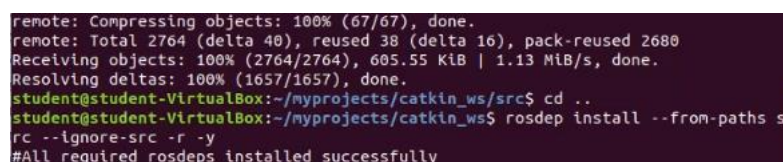
Затем необходимо загрузить репозиторий с примерами. Git репозиторий расположен по адресу [https://github.com/ros/ros\\_tutorials](https://github.com/ros/ros_tutorials), ветка должна соответствовать установленной версии ROS.



```
student@student-VirtualBox: ~/myprojects/catkin_ws/src
File Edit View Search Terminal Help
student@student-VirtualBox:~$ mkdir -p ~/myprojects/catkin_ws/src
student@student-VirtualBox:~$ cd ~/myprojects/catkin_ws/src
student@student-VirtualBox:~/myprojects/catkin_ws/src$ catkin_init_workspace
Creating symlink "/home/student/myprojects/catkin_ws/src/CMakeLists.txt" pointing
to "/opt/ros/melodic/share/catkin/cmake/toplevel.cmake"
student@student-VirtualBox:~/myprojects/catkin_ws/src$ git clone https://github.
com/ros/ros_tutorials.git -b melodic-devel
```

Рис. 1 Скачивание из репозитория

После этого нужно установить зависимости:



```
remote: Compressing objects: 100% (67/67), done.
remote: Total 2764 (delta 40), reused 38 (delta 16), pack-reused 2680
Receiving objects: 100% (2764/2764), 605.55 KiB | 1.13 MiB/s, done.
Resolving deltas: 100% (1657/1657), done.
student@student-VirtualBox:~/myprojects/catkin_ws/src$ cd ..
student@student-VirtualBox:~/myprojects/catkin_ws$ rosdep install --from-paths s
rc --ignore-src -r -y
#All required rosdeps installed successfully
```

Рис. 2 Скачивание зависимостей

Последним шагом идет сборка проекта:

```

student@student-VirtualBox:~/myprojects/catkin_ws$ catkin_make
Base path: /home/student/myprojects/catkin_ws
Source space: /home/student/myprojects/catkin_ws/src
Build space: /home/student/myprojects/catkin_ws/build
Devel space: /home/student/myprojects/catkin_ws/devel
Install space: /home/student/myprojects/catkin_ws/install
####
### Running command: "cmake /home/student/myprojects/catkin_ws/src -DCATKIN_DEV
EL_PREFIX=/home/student/myprojects/catkin_ws/devel -DCMAKE_INSTALL_PREFIX=/home/
student/myprojects/catkin_ws/install -G Unix Makefiles" in "/home/student/myproj
ects/catkin_ws/build"
####
-- The C compiler identification is GNU 7.5.0

```

Рис. 3 Сборка проекта

### 3.1.2. Запуск демонстрационного проекта

Для запуска проекта необходимо открыть 3 терминала. В первом терминале следует необходимо запустить корневой узел ROS. Во втором терминале нужно запустить проект. В третьем терминале следует ввести одну команду для запуска панели управления роботом.

Команды приведены на изображении ниже.

```

student@student-VirtualBox:~/myprojects/catkin_ws$ cd ~/myprojects/catkin_ws
student@student-VirtualBox:~/myprojects/catkin_ws$ source ./devel/setup.sh
student@student-VirtualBox:~/myprojects/catkin_ws$ roscore
... logging to /home/student/.ros/log/922dfd80-5fa1-11ec-aec5-000c2910bec8/rosla
unch-student-VirtualBox-13271.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:38647/
ros_comm version 1.14.9

SUMMARY
=====
student@student-VirtualBox: ~/myprojects/catkin_ws
File Edit View Search Terminal Help
student@student-VirtualBox:~$ cd ~/myprojects/catkin_ws
student@student-VirtualBox:~/myprojects/catkin_ws$ source ./devel/setup.sh
student@student-VirtualBox:~/myprojects/catkin_ws$ roslaunch turtlesim_no
de
[ INFO] [1639790869.809451794]: Starting turtlesim with node name /turtlesim
[ INFO] [1639790869.813965479]: Spawning turtle [turtle1] at x=[5,544445], y=[5,
544445], theta=[0,000000]

```

Рис. 4 Запуск turtlesim

Чтобы начать управлять роботом-черепахой необходимо добавит плагин *Robot Steering*.

Чтобы запустить черепашку в строке утилиты следует прописать */turtle1* перед */cmd\_vel* как показано на :

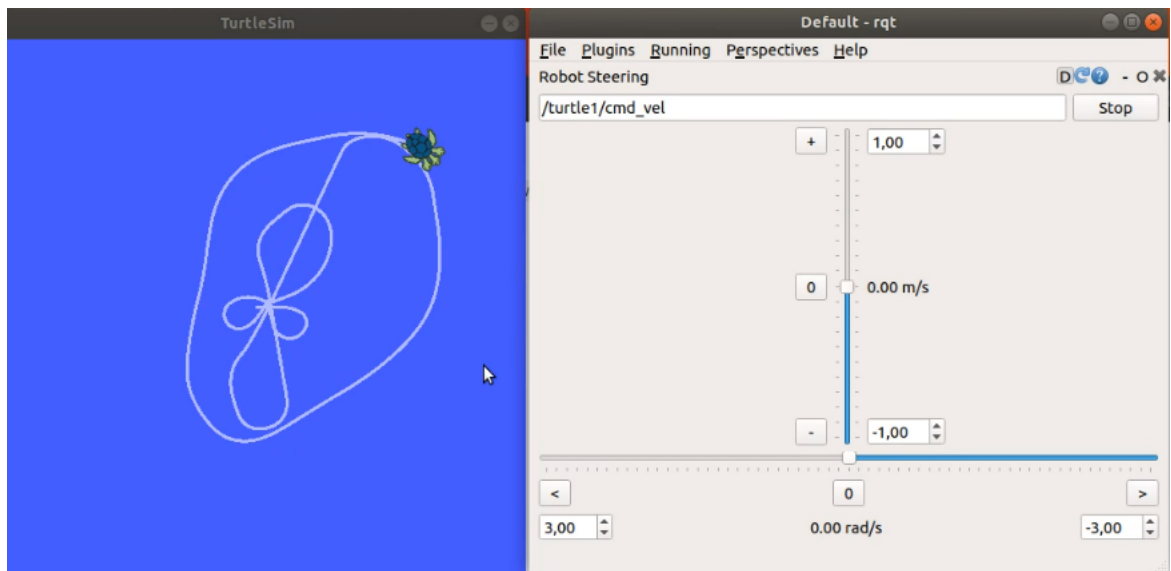


Рис. 5 Управление роботом turtlesim

На синем фоне можно увидеть, как двигается робот, для управления им используются ползунки: вертикальный задает скорость черепахи, а горизонтальный – угловую скорость.

Также помимо управления вручную есть возможность отправлять сообщения роботу с помощью панели Message Publisher. Указав нужный топики выбрав тип сообщений, можно с указанной частотой (Freq.) отправлять заданные сообщения

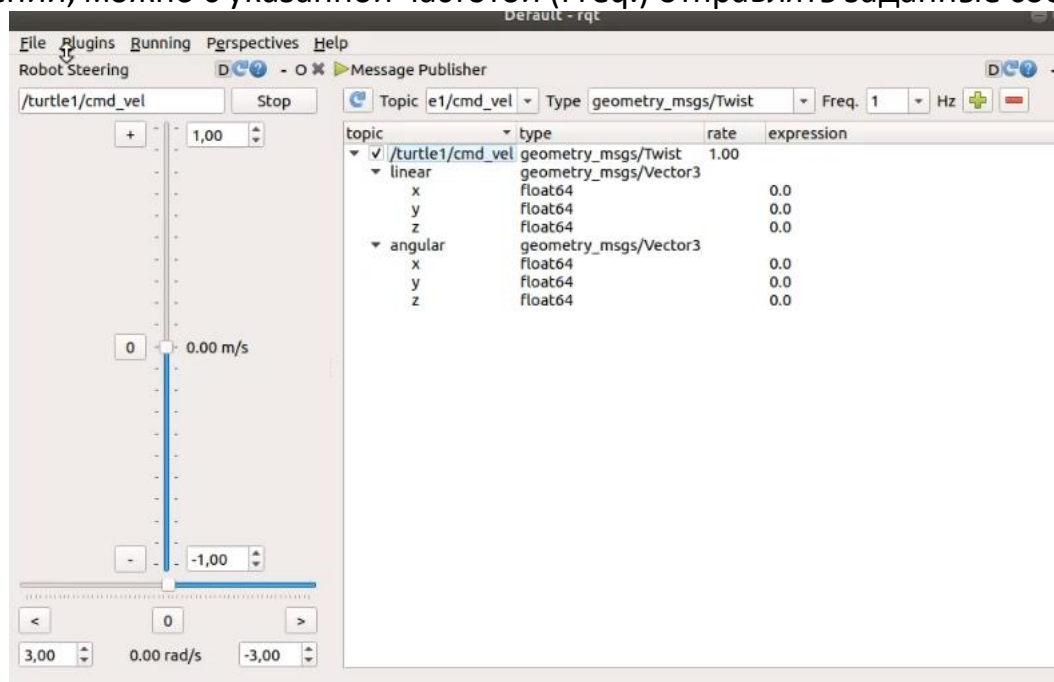


Рис. 6 Панель Message Publisher

## 3.2. Колёсный робот

### 3.2.1. Создание описаний

Приведём основные моменты описания робота.

Макрос для создания кубоида — корпус робота:

```
<!-- Inertial for solid cuboid with dimensions x y z -->
<xacro:macro name="solid_cuboid_inertial" params="rpy xyz mass x y z">
  <inertial>
    <origin rpy="{rpy}" xyz="{xyz}" />
    <mass value="{mass}" />
    <inertia
      ixx="{mass * (y * y + z * z) / 12.0}" ixy="0.0" ixz="0.0"
      iyy="{mass * (x * x + z * z) / 12.0}" iyz="0.0"
      izz="{mass * (x * x + y * y) / 12.0}" />
    </inertial>
  </xacro:macro>
```

Макрос для создания передней поворотной оси:

```
<xacro:macro name="front_wheel_base" params="prefix l_r">
  <link name="{prefix}_front_base">
    <visual>
      <geometry>
        <cylinder radius=".03" length="{wheelrad-0.1}" />
      </geometry>
      <origin xyz="0 0 0" rpy="0 0 0" />
      <material name="white" />
    </visual>
    <collision>
      <geometry>
        <cylinder radius=".03" length="{wheelrad-0.1}" />
      </geometry>
      <origin xyz="0 0 0" rpy="0 0 0" />
    </collision>
    <xacro:default_inertial mass="2" />
  </link>

  <joint name="{prefix}_steer_joint" type="revolute">
    <parent link="base_link" />
    <child link="{prefix}_front_base" />
    <origin xyz="{(length/2-.05)} {l_r*(width/2-.026)} 0" rpy="0 0 0" />
    <axis xyz="0 0 1" />
    <limit effort="10"
      lower="-0.59" upper="0.59"
      velocity="5" />
  </joint>

  <gazebo reference="{prefix}_front_base">
    <material>Gazebo/Grey</material>
  </gazebo>

  <xacro:wheel prefix="{prefix}" suffix="front" left_right="{l_r}" />
</xacro:macro>
```

Макрос для рулевых колес «wheel»:

```
<xacro:macro name="wheel" params="prefix suffix left_right">

  <link name="{prefix}_{suffix}_wheel">
```

```

<visual>
  <origin xyz="0 0 0" rpy="{pi/2} 0 0" />
  <geometry>
    <mesh filename="package://hot_wheels/meshes/tire.dae" scale="{wheelrad} {wheelrad}
    ${left_right*wheelwidth/2}" />
  </geometry>
  <material name="red" />
</visual>
<collision>
  <origin xyz="0 0 0" rpy="{pi/2} 0 0" />
  <geometry>
    <cylinder radius="{wheelrad}" length="{wheelwidth}" />
  </geometry>
</collision>
<xacro:solid_cylinder_inertial
  rpy="0 0 0" xyz="0 0 0"
  mass="5"
  radius="{wheelrad}" length="{wheelwidth}" />
</link>

<joint name="{prefix}_{suffix}_wheel_joint" type="continuous">
  <axis xyz="0 1 0" rpy="0 0 0" />
  <parent link="{prefix}_{suffix}_base" />
  <child link="{prefix}_{suffix}_wheel" />
  <origin xyz="0 ${left_right*wheelwidth} 0" rpy="0 0 0" />
</joint>

<!-- This block provides the simulator (Gazebo) with information on a few additional
physical properties. See http://gazebo.org/tutorials/?tut=ros_urdf for more-->
<gazebo reference="{prefix}_{suffix}_wheel">
  <mu1 value="4.0" />
  <mu2 value="2.0" />
  <kp value="1000000.0" />
  <kd value="1.0" />
  <material>Gazebo/Road</material>
</gazebo>

<!-- This block connects the wheel joint to an actuator (motor), which informs both
simulation and visualization of the robot -->
<transmission name="{prefix}_{suffix}_wheel_trans">
  <type>transmission_interface/SimpleTransmission</type>
  <actuator name="{prefix}_{suffix}_wheel_motor">
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
  <joint name="{prefix}_{suffix}_wheel_joint">
    <hardwareInterface>VelocityJointInterface</hardwareInterface>
  </joint>
</transmission>

```

### Макрос для создания задней оси:

```

<xacro:macro name="back_wheel_base" params="prefix l_r">
  <link name="{prefix}_back_base">
    <visual>
      <geometry>
        <cylinder radius=".05" length="{wheelrad-0.1}" />
      </geometry>
      <origin xyz="0 0 0" rpy="0 0 0" />
      <material name="white" />
    </visual>
  </link>

```

```

</visual>
<collision>
  <geometry>
    <cylinder radius=".05" length="{wheelrad-0.1}"/>
  </geometry>
  <origin xyz="0 0 0" rpy="0 0 0"/>
</collision>
<xacro:default_inertial mass="2"/>
</link>

<joint name="base_to_{prefix}_back" type="fixed">
  <parent link="base_link"/>
  <child link="{prefix}_back_base"/>
  <origin xyz="-{length/2-.05} {l_r*(width/2-.026)} 0" rpy="0 0 0"/>
</joint>

<gazebo reference="{prefix}_back_base">
  <material>Gazebo/Grey</material>
</gazebo>

<xacro:wheel prefix="{prefix}" suffix="back" left_right="{l_r}"/>
</xacro:macro>

```

### Описание робота с использованием приведённых выше макросов:

```

<link name="base_link">
<visual>
  <geometry>
    <box size="{length} {width} {thickness}"/>
  </geometry>
  <material name="blue"/>
</visual>
<collision>
  <geometry>
    <box size="{length} {width} {thickness}"/>
  </geometry>
</collision>
<xacro:solid_cuboid_inertial
  rpy="0 0 0" xyz="0 0 0"
  mass="100"
  x="{length}" y="{width}" z="{thickness}" />
</link>
<xacro:front_wheel_base prefix="left" l_r="1"/>
<xacro:front_wheel_base prefix="right" l_r="-1"/>
<xacro:back_wheel_base prefix="left" l_r="1"/>
<xacro:back_wheel_base prefix="right" l_r="-1"/>

```

В результате робот определяется следующим графом зависимостей между компонентами:



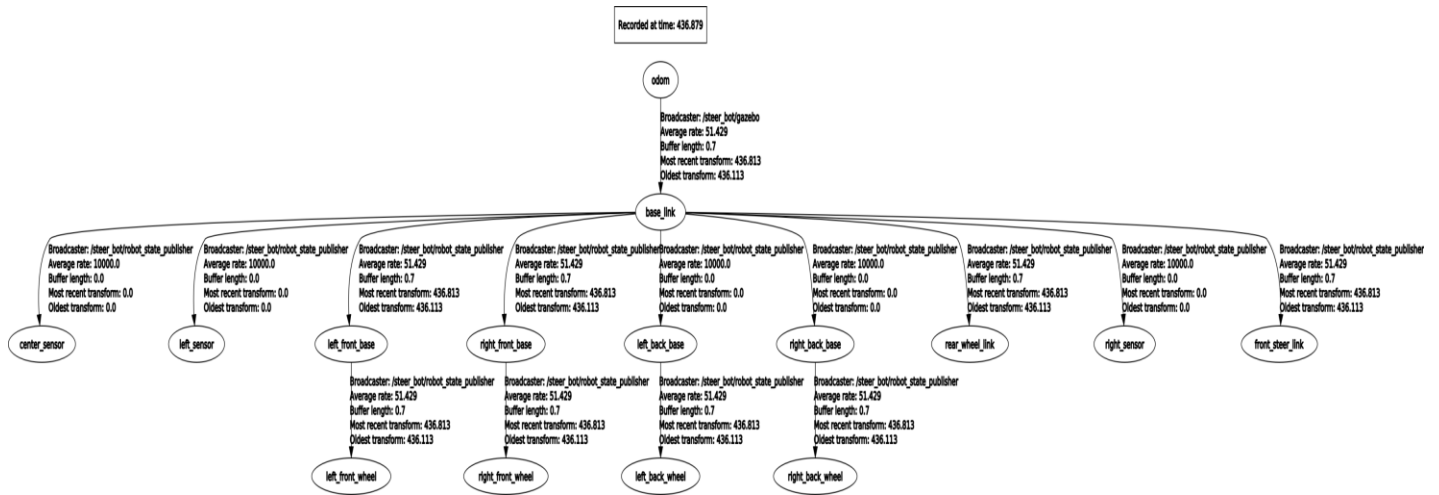


Рис. 7 Граф зависимостей колёсного робота

### 3.2.2. Управление роботом

Для управления робота используется плагин `ackermann_steering_controller`.

Файл запуска:

```
<launch>
<arg name="model" default="$(find hot_wheels)/src/macro.xacro"/>
<arg name="namespace" default="/steer_bot" />
<arg name="world_name" default="worlds/empty_world" />
<arg name="cmd_timeout" default="0.5"/>
<arg name="x" default="0.0"/>
<arg name="y" default="0.0"/>
<arg name="z" default="1.0" />
<arg name="roll" default="0.0"/>
<arg name="pitch" default="0.0"/>
<arg name="yaw" default="0.0"/>

<group ns="$(arg namespace)">

<!-- Gazebo -->
<include file="$(find gazebo_ros)/launch/empty_world.launch">
  <arg name="world_name" default="$(arg world_name)"/>
</include>

<!-- Load the robot description -->
<param name="robot_description" command="$(find xacro)/xacro $(arg model)"/>

<!-- Load ros_controllers configuration parameters -->
<rosparam file="$(find hot_wheels)/config/ctrl_ackermann_steering_controller.yaml" command="load" ns="$(arg namespace)" />
<rosparam file="$(find hot_wheels)/config/ctrl_gains.yaml" command="load" ns="$(arg namespace)" />
<rosparam file="$(find hot_wheels)/config/ctrl_joint_state_publisher.yaml" command="load" ns="$(arg namespace)" />
<rosparam file="$(find hot_wheels)/config/ctrl_steer_bot_hardware_gazebo.yaml" command="load" ns="$(arg namespace)" />

<!-- Spawn the controllers -->
<node pkg="controller_manager" type="spawner" name="controller_spawner" ns="$(arg namespace)"
  args="joint_state_publisher ackermann_steering_controller"
  output="screen" respawn="false" />
```

```

<!-- Launch the robot state publisher -->
<node name="robot_state_publisher" pkg="robot_state_publisher" type="robot_state_publisher">
  <param name="publish_frequency" value="50.0"/>
</node>

<!-- Launch a rqt steering GUI for publishing to /steer_bot/steer_drive_controller/cmd_vel -->
<node pkg="rqt_robot_steering" type="rqt_robot_steering" name="rqt_robot_steering" >
  <param name="default_topic" value="$(arg namespace)/ackermann_steering_controller/cmd_vel"/>
</node>

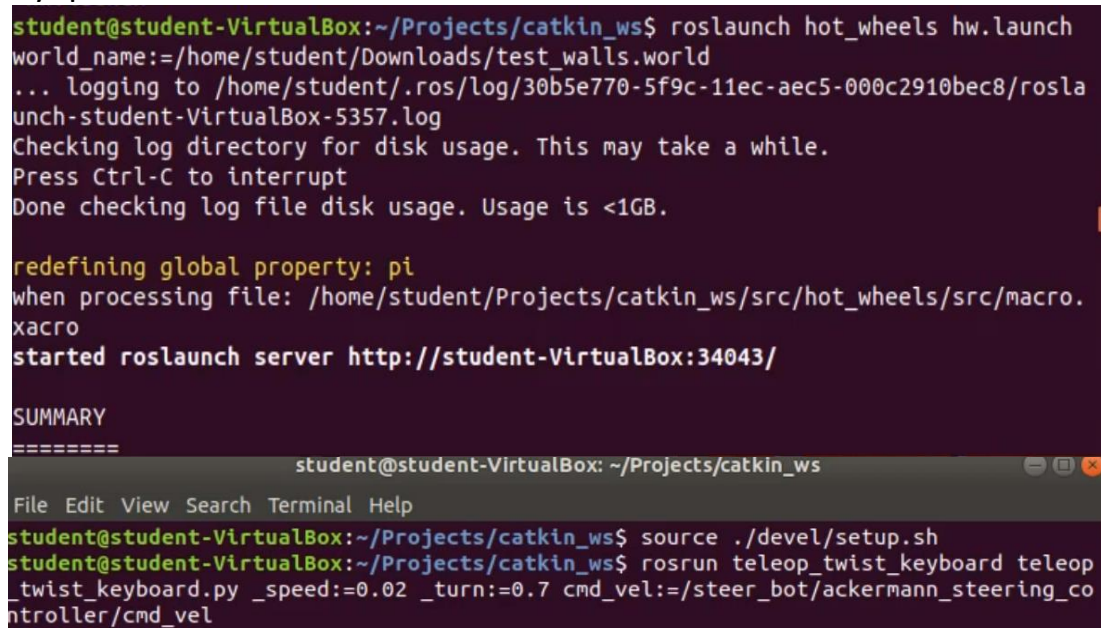
<!-- Spawn robot in Gazebo -->
<node name="spawn_vehicle" pkg="gazebo_ros" type="spawn_model"
  args="-urdf -param robot_description -model steer_bot
    -gazebo_namespace /$(arg namespace)/gazebo
    -x $(arg x) -y $(arg y) -z 5
    -R $(arg roll) -P $(arg pitch) -Y $(arg yaw)"
  respawn="false" output="screen" />

</group>
</launch>

```

### 3.2.3. Запуск работа

Продemonстрируем работоспособность созданного описания. Запустим работа и плагин для управления.



```

student@student-VirtualBox: ~/Projects/catkin_ws$ roslaunch hot_wheels hw.launch
world_name:=/home/student/Downloads/test_walls.world
... logging to /home/student/.ros/log/30b5e770-5f9c-11ec-aec5-000c2910bec8/rosla
unch-student-VirtualBox-5357.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.

redefining global property: pi
when processing file: /home/student/Projects/catkin_ws/src/hot_wheels/src/macro.
xacro
started roslaunch server http://student-VirtualBox:34043/

SUMMARY
=====
student@student-VirtualBox: ~/Projects/catkin_ws
File Edit View Search Terminal Help
student@student-VirtualBox: ~/Projects/catkin_ws$ source ./devel/setup.sh
student@student-VirtualBox: ~/Projects/catkin_ws$ roslaunch teleop_twist_keyboard teleop
_twist_keyboard.py _speed:=0.02 _turn:=0.7 cmd_vel:=/steer_bot/ackermann_steering_co
ntroller/cmd_vel

```

Рис. 8 Запуск колёсного работа

Управление происходит с помощью отправки нажатий клавиш в терминал, в котором запущен teleop\_twist\_keyboard.



Рис. 9 Модель машины

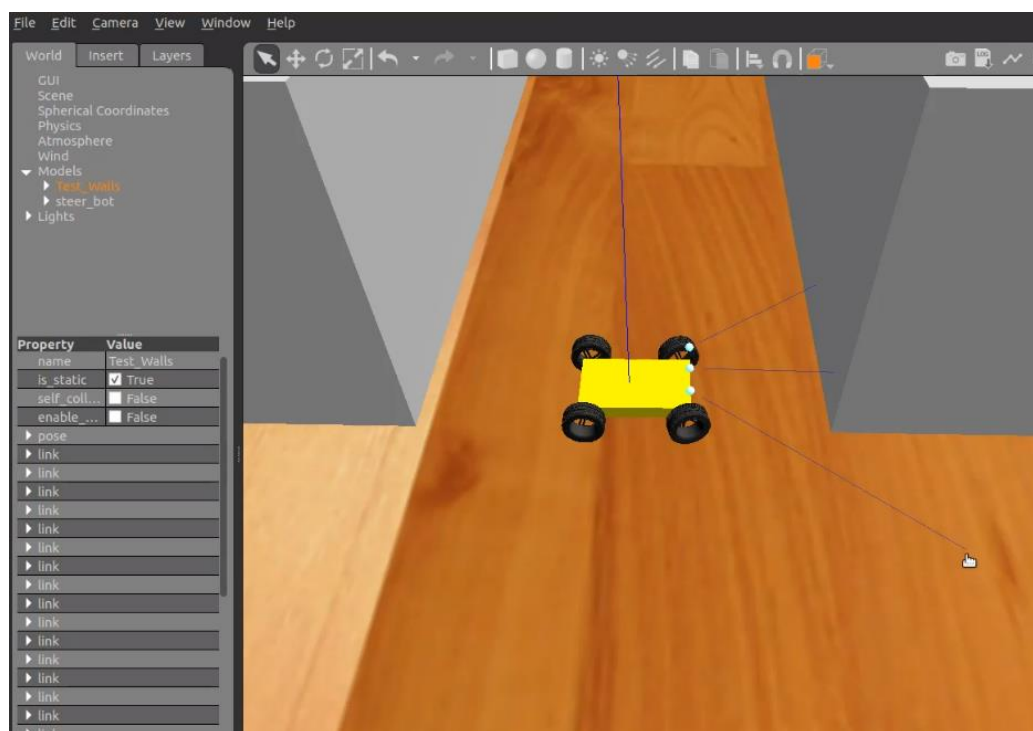


Рис. 10 Демонстрация работы в симуляторе

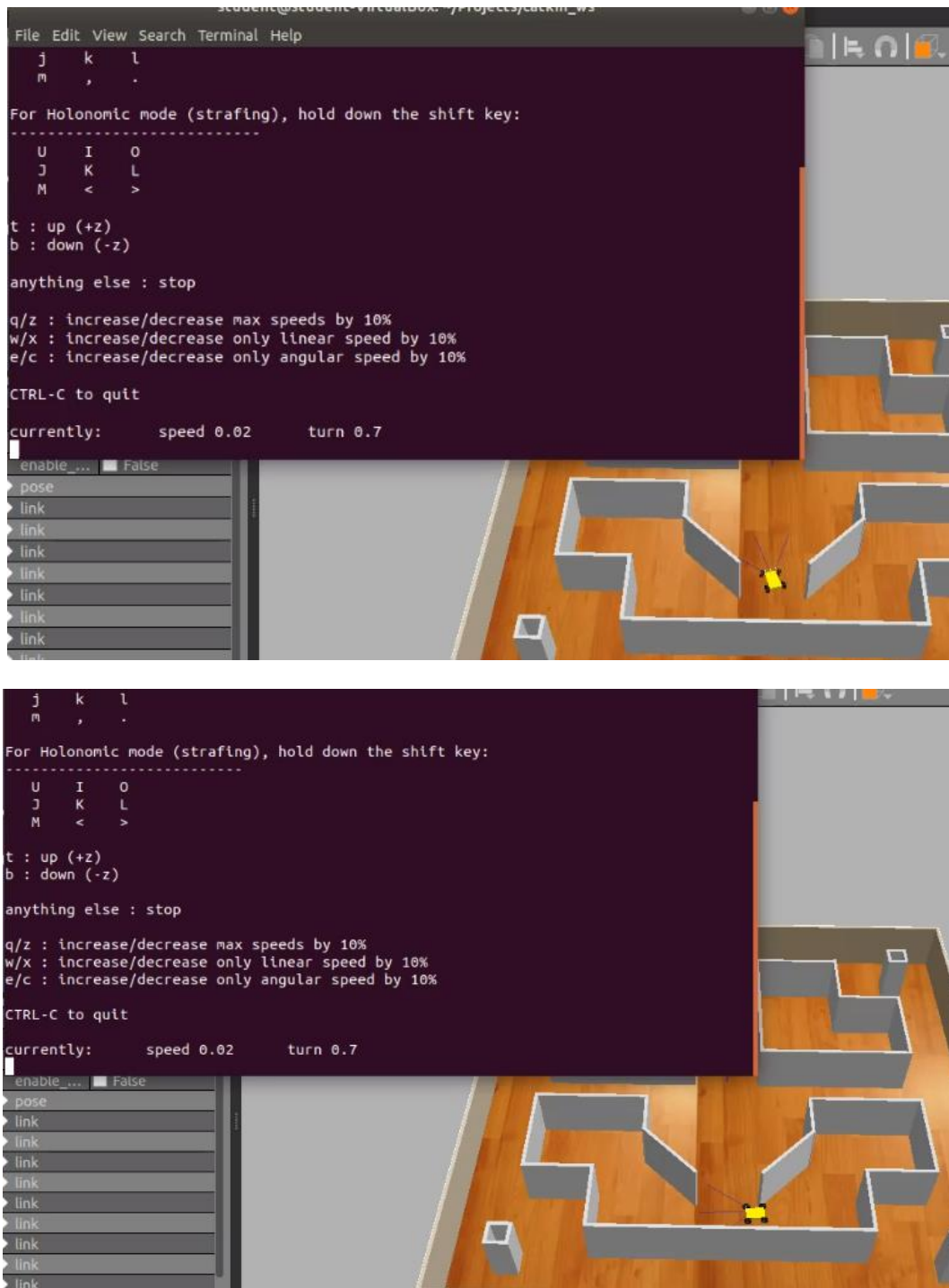


Рис. 11 Демонстрация перемещения робота

#### 4. Выводы

В данной работе мы ознакомились с основами ROS:

- сборка проекта
- запуск проекта
- запуск симулятора

После этого был создан колёсный робот, состоящий из:

- корпуса
- передней виртуальной оси
- задней виртуальной оси

Возможно управление с помощью клавиатуры и на экранных элементов.

Работоспособность созданного решения была проверена в симуляторе Gazebo.