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](#_Toc90556291)

[1.1. Описание платформы 3](#_Toc90556292)

[2. Цель работы 3](#_Toc90556293)

[3. Ход работы 3](#_Toc90556294)

[3.1. Демонстрационный проект turtlesim 3](#_Toc90556295)

[3.1.1. Установка демонстрационного проекта 3](#_Toc90556296)

[3.1.2. Запуск демонстрационного проекта 4](#_Toc90556297)

[3.2. Колёсный робот 6](#_Toc90556298)

[3.2.1. Создание описаний 6](#_Toc90556299)

[3.2.2. Управление роботом 8](#_Toc90556300)

[3.2.3. Запуск робота 9](#_Toc90556301)

[4. Выводы 10](#_Toc90556302)

1. Справочная информация
   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.

1. 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.

1. Ход работы
   1. Демонстрационный проект turtlesim
      1. Установка демонстрационного проекта

Затем необходимо загрузить репозиторий с примерами. Git репозиторий расположен по адресу <https://github.com/ros/ros_tutorials>, ветка должна соответствовать установленной версии ROS.

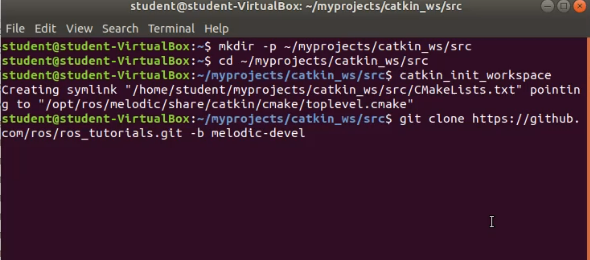


Рис. 1 Скачивание из репозитория

После этого нужно установить зависимости:

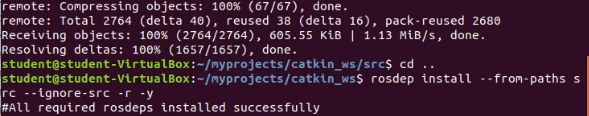


Рис. 2 Скачивание зависимостей

Последним шагом идет сборка проекта:



Рис. 3 Сборка проекта

* + 1. 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.

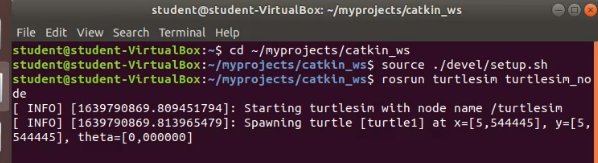


Рис. 4 Запуск turtlesim

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 :

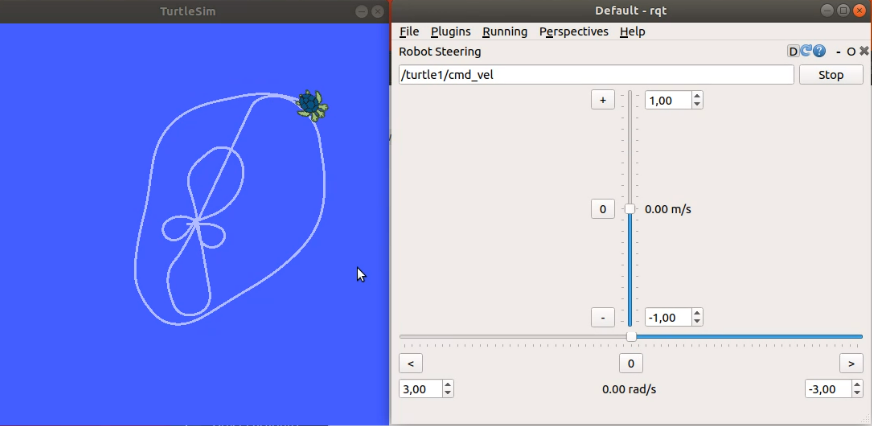


Рис. 5 Управление роботом turtlesim

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

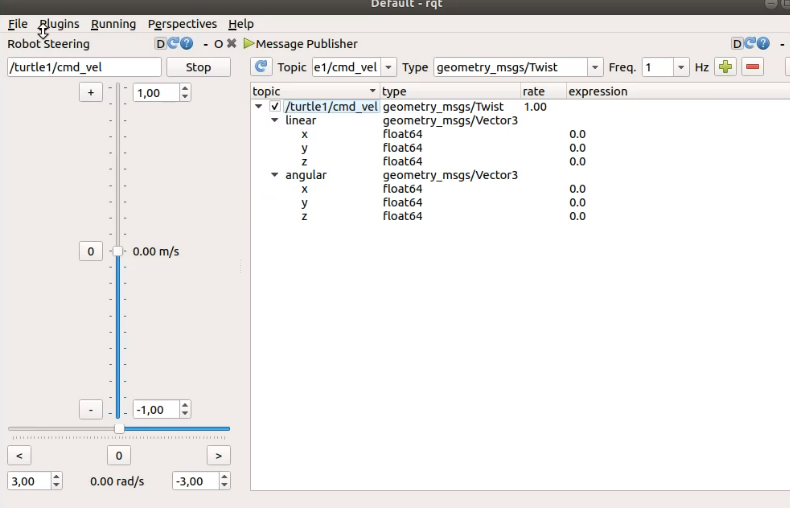


Рис. 6 Панель Message Publisher

* 1. Колёсный робот
     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://gazebosim.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>

            <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:

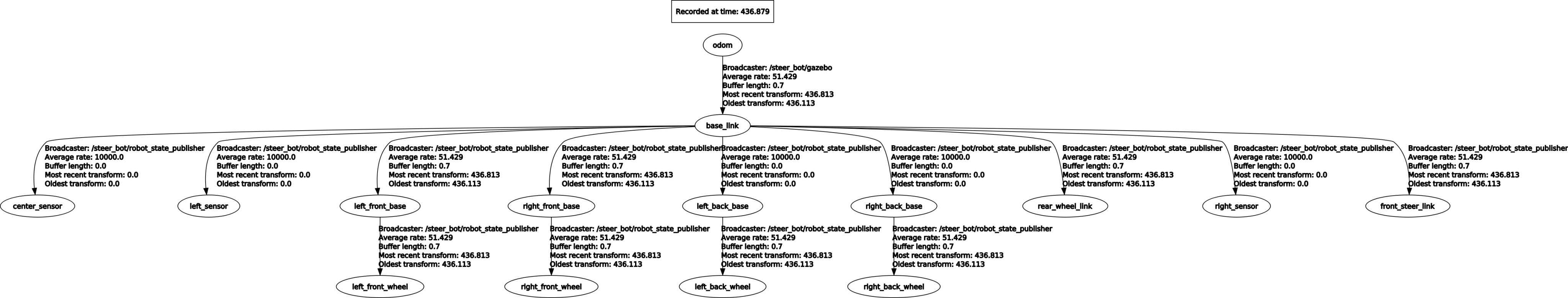


Рис. 7 Граф зависимостей колёсного робота

* + 1. 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)"

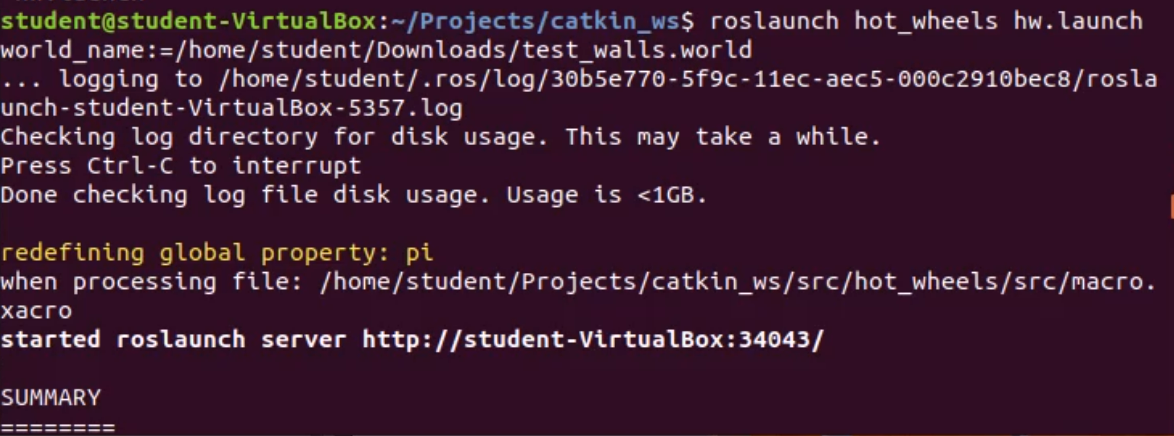
          respawn="false" output="screen" />

  </group>

</launch>

* + 1. Запуск робота

Продемонстрируем работоспособность созданного описания. Запустим робота и плагин для управления.



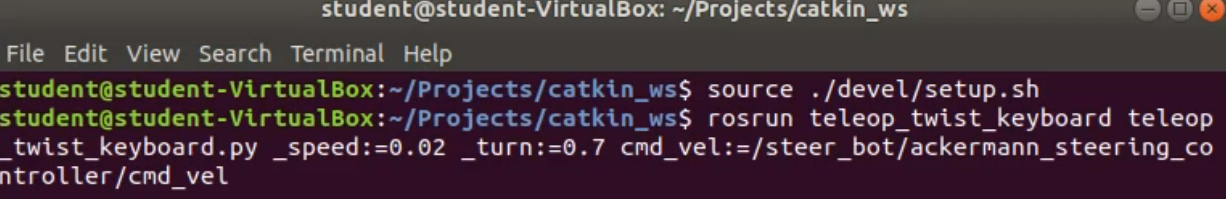


Рис. 8 Запуск колёсного робота

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



Рис. 9 Модель машины

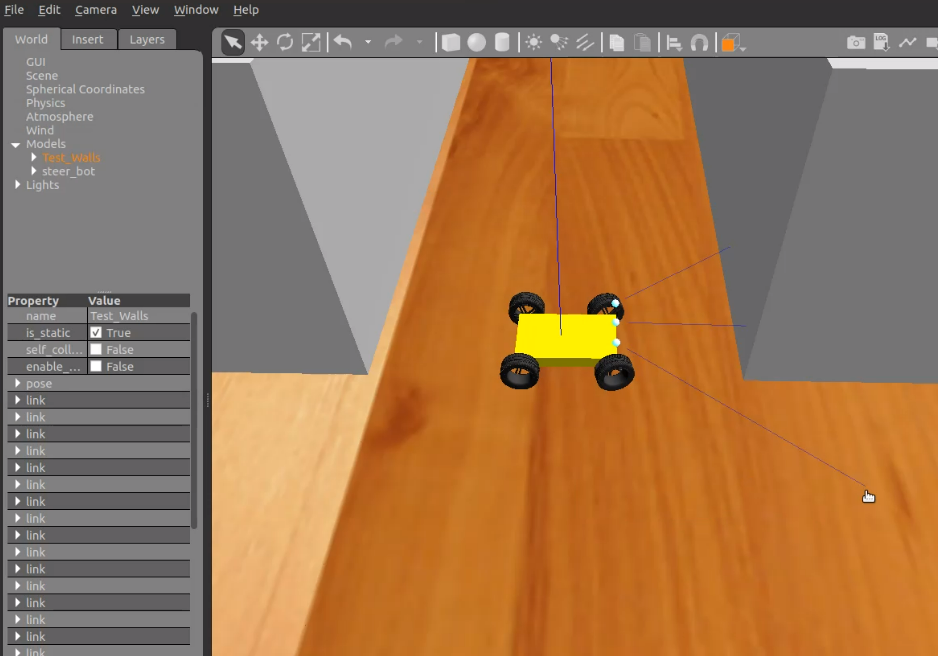
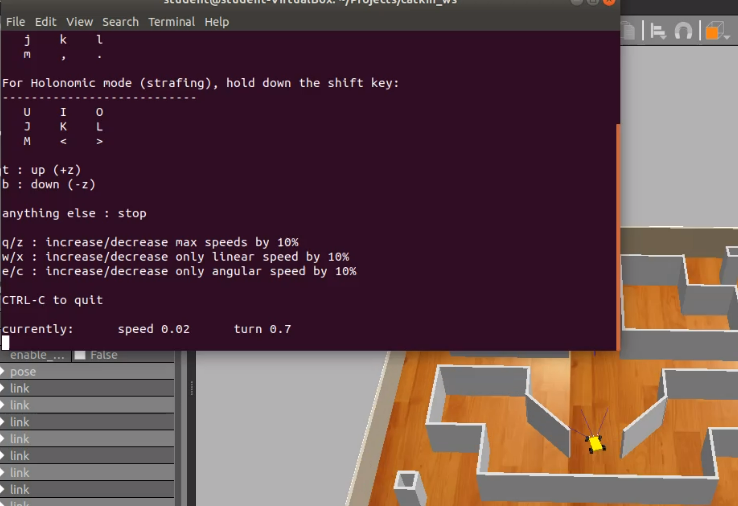


Рис. 10 Демонстрация работы в симуляторе



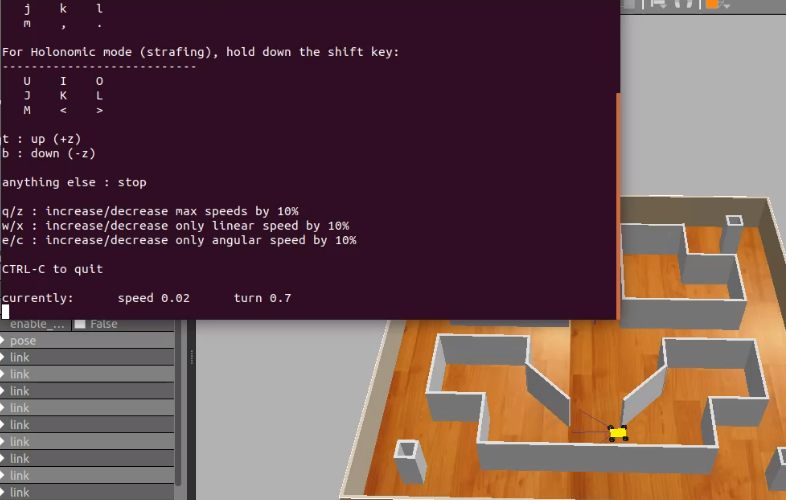


Рис. 11 Демонстрация перемещения робота

1. 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.