**ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ВЫСШЕГО Профессионального образования**

**“ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”**

“Электропривод и автоматизация промышленных установок”

(полное название кафедры)

**КУРСОВАЯ РАБОТА**

по дисциплине: «Моделирование электромеханических систем»

(название дисциплины)

на тему: «Моделирование робота на Mecanum колесах в Gazebo»

Студента 4 курса\_\_ЭАПУ-18\_\_\_группы

направления подготовки\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

профиль\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_Ходаковского Я.С\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(фамилия и инициалы)

Руководитель\_\_\_ст. пр. Бажутин Д.В.\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(должность, ученое звание, научная степень, фамилия и инициалы)

Национальная шкала\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Количество баллов:\_\_\_\_\_\_Оценка ECTS\_\_\_\_\_

Члены комиссии \_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (фамилия и инициалы)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (фамилия и инициалы)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (фамилия и инициалы)

г.Донецк – 2021

ГОУ ВПО «Донецкий национальный технический университет»

Кафедра: Электропривод и автоматизация промышленных установок

Дисциплина: Моделирование электромеханических систем

Специальность: Электропривод и автоматика d

Курс: 4 Группа: ЭАПУ-18 Семестр: 7 d

**ЗАДАНИЕ**

**на курсовой проект студента**

Ходаковского Ярослава Сергеевича

**(Ф.И.О. студента)**

1.Тема: Моделирование робота на Mecanum колесах в Gazebo

2. Cрок сдачи студентом законченного проекта (работы): 28.12.2021

3. Исходные данные к проекту (работе):

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке):

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

5. Перечень графического материала:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

6.Дата выдачи задания: 25.03.2021

**КАЛЕНДАРНЫЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Название этапов курсовой работы | Срок  выполнения | Примечания |
| 1 |  |  |  |
| 2 |  |  |  |
| 3 |  |  |  |
| 4 |  |  |  |
| 5 |  |  |  |
| 6 |  |  |  |
| 7 |  |  |  |
| 8 |  |  |  |
| 9 |  |  |  |
| 10 |  |  |  |
| 11 |  |  |  |
| 12 |  |  |  |
| 13 |  |  |  |
| 14 |  |  |  |

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Руководитель Бажутин Д.В. (Ф.И.О. преподавателя) (подпись)

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20 21 год

# РЕФЕРАТ

Пояснительная записка к курсовой работе

31 с., 13 рис., 1 табл., 2 приложения, 7 источников

Объектом разработок и исследования является робот на Mecanum колесах.

Цель работы – разработка модели робота на Mecanum колесах. Использование модели робота в среде симуляции Gazebo для исследования его работы в различных режимах, а также для проверки работоспособности программы, без использования реального объекта.

В курсовой работе использовались такие методы: моделирование (замена реального объекта его эквивалентом в той или иной степени точности повторяющим свойства оригинала). В качестве модели робота на Mecanum колесах использовалась имитационная модель (группа компьютерных моделей, отображающая физические свойства реальных объектов на основании встроенных алгоритмов без непосредственного участия пользователя).

В пояснительной записке представлены следующие результаты выполнения курсовой работы: исследование кинематики четырехколесного робота на Mecanum колесах, знакомство с операционной системой для роботов – ROS, установка ROS на Linux Ubunty, создание модели робота в URDF – формате, моделирование модели робота в среде Rviz, создание launch-файла для запуска Gazebo, создание модели в формате .gazebo, создание ноды управления роботом с клавиатуры, исследование работы модели робота на Mecanum колесах в среде Gazebo.

Математическое моделирование, ИМИТАЦИОННАЯ МОДЕЛЬ, mecanum колеса, ROBoT OPERATING SYSTEM, УЗЛЫ,

URDF, RVIZ, GAZEBO

содержание

Введение 5

1 четырехколесный робот на mecanum колесах 6

2 ROBOT OPERATING SYSTEM 11

2.1 Общие сведения 11

2.2 Выбор и установка дистрибутива ROS 13

2.3 Среда визуализации Rviz 16

2.4 Среда моделирования Gazebo 20

ЗАКЛЮЧЕНИЕ 27

СПИСОК ЛИТЕРАТУРЫ 28

ПРИЛОЖЕНИЕ А - ПЕРЕЧЕНЬ ЗАМЕЧАНИЙ НОРМОКОНТРОЛЕРА 29

ПРИЛОЖЕНИЕ Б - КОД ОСНОВНОЙ ПРОГРАММЫ 30

ВВЕДЕНИЕ

Моделирование – это замена реального объекта его эквивалентом в той или иной степени точности повторяющим свойства оригинала. Моделированием также называют анализ процессов, протекающих в реальном объекте путем изучение свойств его эквивалентной модели.

В моделировании часто применяются имитационные модели, которые отображают физические свойства реальных объектов на основании встроенных алгоритмов без непосредственного участия пользователя.

Зачастую, на практике моделирование используют в случаях, когда необходимо проверить работоспособность системы, не подвергая реальный объект опасности. Так же оно применяется в тех случаях, когда работа непосредственно над реальной системой проблематична (например, исследование процессов в крупных механизмах промышленных предприятий).

В курсовой работе использовался программный комплекс Gazebo, позволяющий симулировать работу роботов, работу датчиков, а также взаимодействие роботов с окружающей средой. Применение Gazebo позволило исследовать работоспособность разработанной программы управления роботом, без непосредственного использования реального объекта.

Таким образом, использование моделирования существенно упрощает процесс разработки, исследования, настройки различных систем.

1 ЧЕТЫРЕХКОЛЕСНЫЙ РОБОТ НА MECANUM КОЛЕСАХ

Mecanum колесо, или колесо Илона (также употребляется название “Шведское колесо”) – это роликонесущее колесо, которое способно обеспечить движение в любом направлении.



Рисунок 1.1 – Шведское колесо

Название колесо получило от своего изобретателя - Бенгта Илона, шведского изобретателя, представителя компании «Mecanum AB» в 1973 году [1].

Конструктивно представляет из себя колесо с смонтированными на нем роликами, установленными на протяжении всей окружности под углом 45˚[2]. Стоит отметить, что при установке колес Илона необходимо соблюдать следующее правило: все колеса должны быть установлены так, чтобы ось вращения верхнего ролика указывала на центр платформы.

Основное преимущество таких колес – возможность движения в любом направлении: вперед-назад, вправо-влево, по диагонали и др. [3]. Это достигается путем изменения направления и скорости вращения отдельных колес как показано на рисунке 1.2.

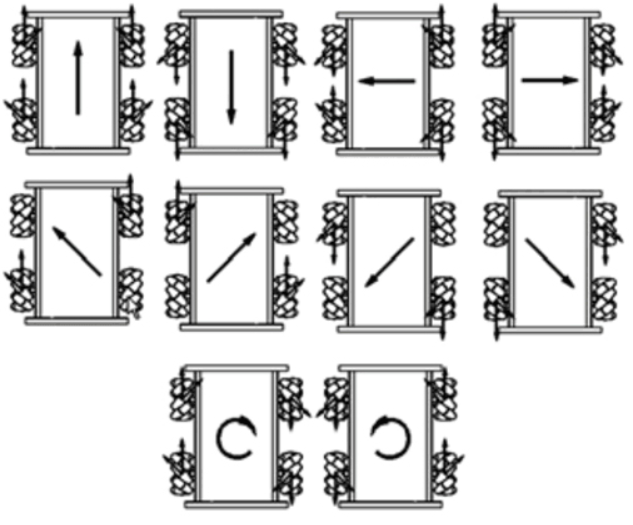


Рисунок 1.2 – Принцип работы колес Илона

Т.е. четыре колеса платформы двигаются по отдельности с разными скоростями и в разном направлении. Ролики преобразуют силы вращения колеса в силу, перпендикулярную направлению движения колеса. Векторная сумма сил формирует суммарный вектор силы для каждого направления [4].

Примем следующие обозначения:

1. R – радиус колеса;
2. ω1, ω2, ω3, ω4 – угловые скорости колес;
3. νg1, νg2, νg3, νg4 - скорости роликов на каждом из колес;
4. νx, νy– скорость платформы в направлениях X, Y;
5. ω0 – угловая скорость платформы;
6. O – центр платформы (исходная точка глобальных координат);
7. O1, O2 , O3, O4,– исходные точки локальных коорд. каждого колеса;
8. L1 – расстояние между центром платформы и центром колеса;
9. L2 – расстояние между центром платформы и осью вращения колеса;
10. α– угол расположения роликов (45˚)

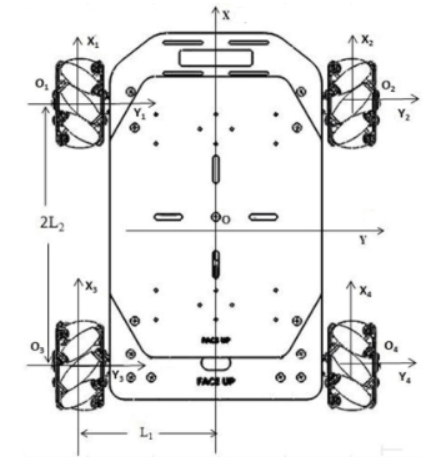


Рисунок 1.3 – Глобальная система координат

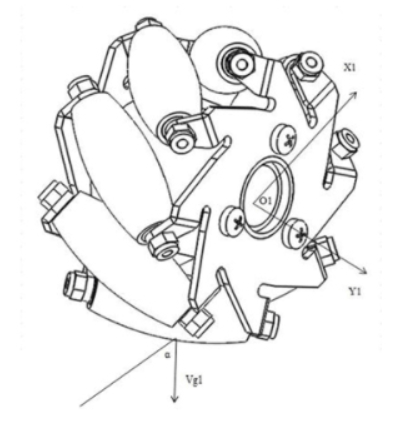


Рисунок 1.4 – Локальная система координат

В таком случае, кинематика движения четырехколесного робота на Mecanum колесах будет описываться следующими выражениями:

Скорость в центре 1-го колеса (в глобальных координатах):



В локальных координатах:



Объединив выражения (1.1) и (1.2), получим:



Тогда, угловая скорость первого колеса определяется путем преобразований формул (1.3). Угловая скорость первого колеса:



Аналогично рассматриваются угловые скорости трех других колес. Итоговое выражение имеет вид, представленный в формуле (1.5). Формула (1.5) демонстрирует взаимоотношения между движением платформы и скоростями вращения колес.



Из приведенных формул и рисунка 1.2 можно сделать вывод, что взаимосвязь между движением колес и платформы определяется следующими выражениями:

1. Если колеса на одной стороне вращаются в сторону, противоположную вращению колес на другой стороне, платформа будет вращаться;
2. При четырех колесах, движущихся в одном направлении платформа будет двигаться вперед или назад (в зависимости от направления движения колес);
3. Для движения платформы по диагонали необходимо, чтобы два колеса, расположенных диагонально друг к другу, вращались в одну сторону. А два других колеса двигаться не должны.
4. Если колеса на одной стороне движутся друг к другу, а на другой стороне – друг от друга, то платформа будет двигаться вбок.



Рисунок 1.5 – робот на Mecanum колесах URANUS

2 ROBOT OPERATING SYSTEM

2.1 Общие сведения

ROS (Robot Operating System) - операционная система для роботов.

ROSбыла разработана в 2007 году в лаборатории искусственного интеллекта Стэнфорда для поддержки стэнфордского проекта *AI Robot*. C 2008 года, разработка продолжилась в основном в исследовательском институте Willow Garage, сотрудничающим с более чем двадцатью различными институтами в рамках модели совместного развития.

Разработка роботов – задача, требующая множество различных инструментов, имеющих конкретное назначение. Это может быть инструмент для работы с компьютерным зрением, инструмент симуляции и т.д.

ROS объединяет в себе эти инструменты и позволяет максимально комфортно их использовать. Он основан на архитектуре графов (создаются так называемые узлы, которые могут обмениваться сообщениями друг с другом), и обеспечивает такие функции, как аппаратная абстракция, управление пакетами и низкоуровневый контроль устройств.

Ключевой особенностью является возможность работы с различными роботами, лишь незначительно изменяя основную программу [5]. Также, удобным достоинством ROS является то, то для большого количества задач уже имеются готовые решения, написанные другими людьми и находящиеся в свободном доступе. Это позволяет тратить на разработку значительно меньшее количество времени и сил.

Основными понятиями Robot Operating System являются:

1. Мастер (Master) – выполняет роль сервера имен, позволяет нодам подключаться друг к другу, связывает издателей и подписчиков. Запускается с помощью команды roscore.
2. Нода (Node) –наименьшая рабочая единица в ROS. Ноды объединяются между собой в один граф, и обращаются друг к другу через топики и сервисы. Как правило, для управления роботом необходимо большое число нод, каждая из которых выполняет определенную функцию. Например, одна из нод отвечает за планирование маршрута, другая – за управление колесами робота, третья – за графическое представление всей системы.

Использование такой структуры дает уйму преимуществ. Например, ошибку в системе легче обнаружить, если известно, в какой ноде она находится. Упрощается написание программы, так как она разбита на несколько частей. Это позволяет использовать одну ноду сразу в нескольких проектах, схожих по смыслу. Каждая нода имеет свое собственное название и тип. С помощью специальных утилит можно посмотреть, с чем связана та или иная нода.

1. Сообщения (Messages) – тип данных для ROS. Используется нодами как язык общения друг с другом. Сообщение – это структура, представляющая собой список переменных и их типов. Сообщения могут быть как простых типов (integer, float, boolean), так и могут состоять из сложных структур, содержащих вложенные сообщения и массивы сообщений). Например, для сообщение с координатами объекта (XYZ)есть существующий тип сообщения geometry\_msgs/Point.msg
2. Топик (Topic) – представляют собой шину для общения нод между собой. Нода издателя регистрирует свою тему на мастере и начинает публиковать сообщения в эту тему. Ноды-подписчики подключаются к данному топику с помощью того же мастера и получают из него сообщения.



Рисунок 2.1 – Демонстрация работы топика на примере датчика температуры

1. Издатель (Publisher) – процесс, создающий топик и отправляющий в него сообщения. Одна нода может содержать несколько издателей, публикующих сообщения в разные топики.
2. Подписчик (Subscriber) – процесс, который получает сообщения из определенного топика. Подписчик должен зарегистрироваться на мастере и указать какие топики ему необходимы. Связь подписчика и издателя является асинхронной. Это значит, что отправка сообщений в топик происходит независимо от статуса подписчиков. Это удобно использовать для работы с датчиками, непрерывно передающими полученные значения.
3. Пакет (Package) – формируют атомарный уровень ROS. Пакеты представляют собой группу нод, объединенных одной целью (аналог программы).
4. Сервис (Service) – способ коммуникации, отличный от топика. Работает по принципу синхронной связи (клиент запрашивает данные и получает на них ответ).
5. Действие (Action) – еще одна модель связи. Применяется в тех случаях, когда необходимо получать информацию о состоянии выполнения запроса (обратная связь). Клиент устанавливает цель, сервер начинает выполнять ее, и в процессе информирует клиента о ходе выполнения. По окончании обработки, клиент получает от сервера сообщение о завершении процесса.

2.2 Выбор и установка дистрибутива ROS.

Существует несколько версий ROS. Это обусловлено необходимостью предоставить разработчикам современную и стабильную версию системы. Как правило, после выпуска дистрибутива разработчик старается не изменять основных параметров, а только исправить ошибки.

Поддержка некоторых из дистрибутивов уже прекращена, в то время как дистрибутивы ROS Noetic Ninjemys и ROS Melodic Morenia поддерживаются и являются актуальными.

В данной работе использовался дистрибутив ROS Melodic. Список дистрибутивов приведен в таблице 2.1.

Таблица 2.1 – Список дистрибутивов ROS

|  |  |
| --- | --- |
| Название сборки | Дата выхода |
| ROS Noetic Ninjemys | 23 мая 2020 года |
| ROS Melodic Morenia | 23 мая 2018 года |
| ROS Lunar Loggerhead | 23 мая 2017 года |
| ROS Kinetic Kame | 23 мая 2016 года |
| ROS Jade Turtle | 23 мая 2015 года |
| ROS Indigo Igloo | 22 июля 2014 года |
| ROS Hydro Medusa | 4 сентября 2013 года |

Установка ROS производится в следующем порядке:

В новом терминале Ubuntu вводится команда, дающая разрешение на установку файлов на компьютер с сайта packages.ros.org:

sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb\_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'

Команда настройки сетевых ключей:

sudo apt install curl

curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt-key add -

Непосредственная установка начинается с проверки наличия обновлений:

sudo apt update

Далее производится выбор пакета и необходимых инструментов. Для полной установки ROS, включающей в себя полезные библиотеки для визуализации и симуляции, используется команда:

sudo apt install ros-melodic-desktop-full

Чтобы получить доступ к инструментам и командам *ROS* в текущей оболочке *bash*, требуется добавить переменные среды *ROS* в файл *bashrc*. Эта операция будет выполняться в начале каждой сессии *bash*. Ниже приведена команда для добавления переменной *ROS* .*bashrc*:

echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc

source ~/.bashrc

Для того, чтобы создавать и изменять собственные пакеты, необходимо установить некоторые инструменты, которые поставляются по-отдельности. Для их установки используется команда:

sudo apt install python-rosdep python-rosinstall python-rosinstall-generator python-wstool build-essential

Следующий шаг – инициализация *rosdep*. Это позволит легко установить системные зависимости для исходных пакетов ROS, или скомпилировать некоторые компоненты ROS, необходимые для работы:

sudo apt install python-rosdep

sudo rosdep init

rosdep update

Теперь необходимо создать рабочую область ROS. Для этого воспользуемся системой сборки catkin. Система *catkin* генерирует исполняемую или общую библиотеку из исходного кода.

Рабочее пространство *Сatkin* – это папка, в которой можно создавать, изменять и устанавливать пакеты *catkin*. Представленная ниже команда создаст родительский каталог *catkin\_ws* и вложенную папку с именем *src.*

$ mkdir -p ~/catkin\_ws/src

$ cd ~/catkin\_ws/

$ catkin\_make

$ source devel/setup.bash

2.3 Среда визуализации RViz

Rviz (ROS Visualization) – мощная среда 3D визуализации для ROS. Среда позволяет пользователю визуализировать модель своего робота, отобразить сигналы, приходящие с датчиков и многое другое [6].

Если робот взаимодействует с рабочей станцией, на которой открыт RViz, программа отобразит текущее состояние робота на его модели. Это означает, что в случае использования, например, манипулятора, RViz способен отобразить его текущее положение в пространстве.

Установка RViz была производится автоматически, если при установке ROS был выбран вариант desktop-full.

Для использования RViz необходимо сначала запустить ROS Master (команда roscore), а затем ввести команду:

**$ rosrun rviz rviz**

Среда визуализации RViz изображена на рисунке 2.2.1.

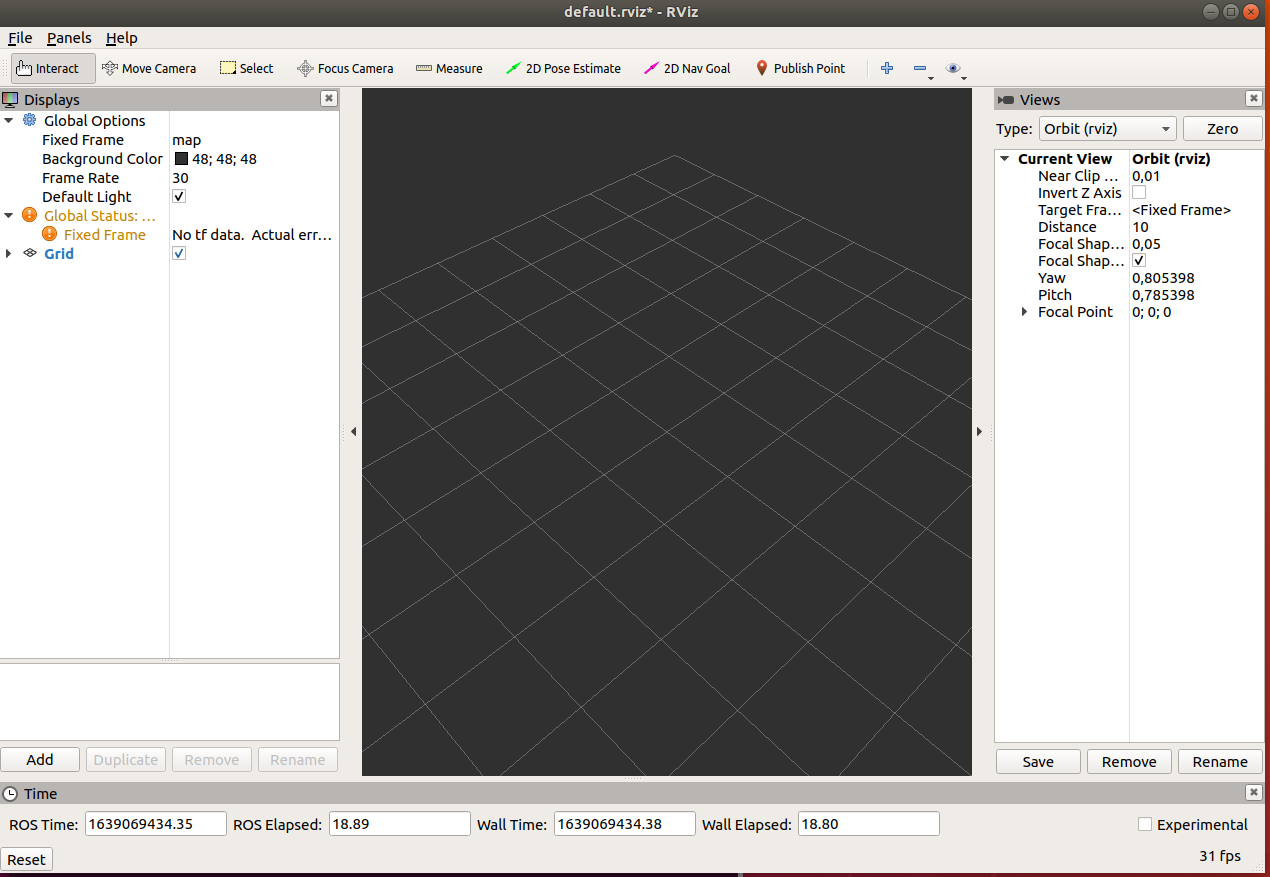


Рисунок 2.2 – Главное меню программы RViz

Для создания моделей роботов, которые в последствии будут использоваться в RViz, необходимо воспользоваться форматом URDF (Unified Robot Description Format). Данный формат позволяет создавать роботов различной конфигурации, настраивать кинематические связи и узлы в роботе. В URDF существует шесть возможных типов узлов, которые представлены ниже:

1. Неподвижный: Узел не может двигаться. Все степени свободы заблокированы. Этот тип узла не требует оси, калибровки, динамики, ограничений или контроллера безопасности.
2. Вращательный: Шарнир вращается вокруг одной оси и имеет диапазон, задаваемый верхним и нижним пределами.
3. Непрерывный: Это непрерывный шарнир, который вращается вокруг оси и не имеет верхнего и нижнего пределов.
4. Призматический: Это скользящий шарнир, который скользит вдоль оси и имеет ограниченный диапазон, заданный верхним и нижним пределами.
5. Плавающий: Этот шарнир обеспечивает движение по всем шести степеням свободы.
6. Планарный: Этот шарнир обеспечивает движение в плоскости, перпендикулярной оси.

Полный текст URDF файла занимает более 500 строк, поэтому ниже приведены основные пункты. Выбор имени робота, и создание «материалов» - т.е. цветовых параметров, которые могут быть применены к той или иной детали робота (в данном случае желтый цвет для тела робота, и черный цвет колес):

|  |
| --- |
| <robot name="nexus\_4wd\_mecanum" xmlns:xacro="http://ros.org/wiki/xacro"> |
| <material name="yellow"> |
| <color rgba="0.8 0.8 0.0 1.0"/> |
| </material> |
| <material name="black"> |
| <color rgba="0.15 0.15 0.15 1.0"/> |

Создание тела робота с помощью заранее созданной модели тела в SolidWorks и его окрашивание в желтый цвет.

|  |
| --- |
| <visual> |
|  | <origin rpy="0 0 0" xyz="0 0 0"/> |
|  | <geometry> |
|  | <mesh filename="package://nexus\_4wd\_mecanum\_description/meshes/  nexus\_base\_link.STL" scale="0.001 0.001 0.001"/> |
|  | </geometry> |
|  | <material name="yellow"/> |
|  | </visual> |

Создание узла переднего левого колеса:

<joint name="upper\_left\_wheel\_shaft\_joint" type="fixed">

|  |  |
| --- | --- |
|  | <origin rpy="0 0 0" xyz="0.150 0.108 0.0"/> |
|  | <parent link="base\_link"/> |
|  | <child link="upper\_left\_wheel\_shaft"/> |

</joint>

Описание момента инерции Mecanum-колеса

|  |
| --- |
| <inertial> |
|  | <origin rpy="0 0 0" xyz="0 0.01221 0"/> |
|  | <mass value="0.055"/> |
|  | <inertia ixx="0.000021370" ixy="0" ixz="0" iyy="0.000009966" iyz="0" izz="0.000021370"/> |
|  | </inertial> |

Описание collision-модели переднего левого колеса, которое в последствии будет использоваться при моделировании движения работа (напр. остановка вследствие препятствия):

<collision>

|  |  |
| --- | --- |
|  | <origin xyz="0 0 0"/> |
|  | <geometry> |
|  | <mesh filename="package://nexus\_4wd\_mecanum\_description/meshes/wheel\_shaft.STL" scale="0.001 0.001 0.001"/> |
|  | </geometry> |
|  | </collision> |

Таким образом описываются и все остальные колеса, а также дополнительные визуальные элементы модели.

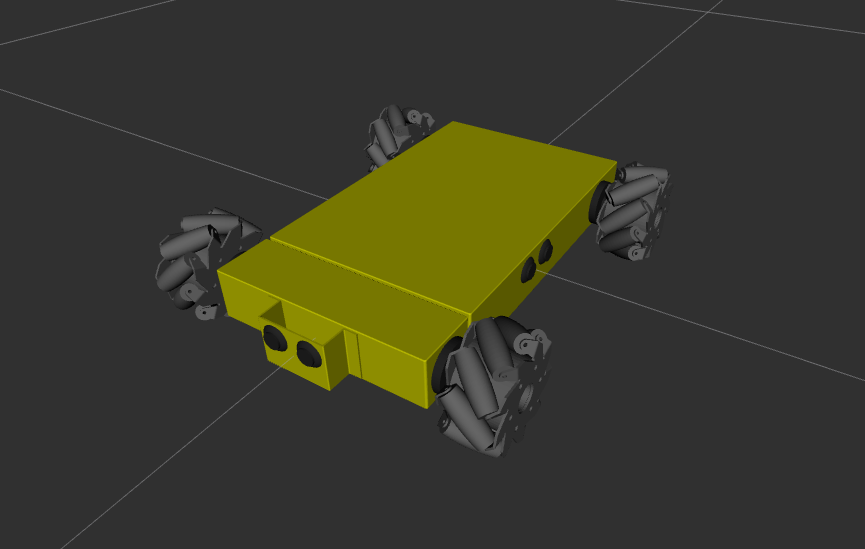


Рисунок 2.3 – Модель робота на Mecanum-колесах в RViz

2.4 Среда моделирования Gazebo

Gazebo – бесплатное программное обеспечения для моделирования, разработанное компанией Willow Garage [7]. Gazebo предоставляет следующие возможности:

* Создание простейших моделей роботов;
* Быстрое прототипирование и проверка работоспособности;
* Регрессионное тестирование на основе реальных сценариев;
* Симуляция окружающей среды любых типов;
* Моделирование работы сенсоров, лазеров, 2D/3D камер, датчиков;

Отдельная установка Gazebo не требуется, если при установке ROS был выбран дистрибутив (desktop - full). Запуск среды производится командой:

**$ gazebo**

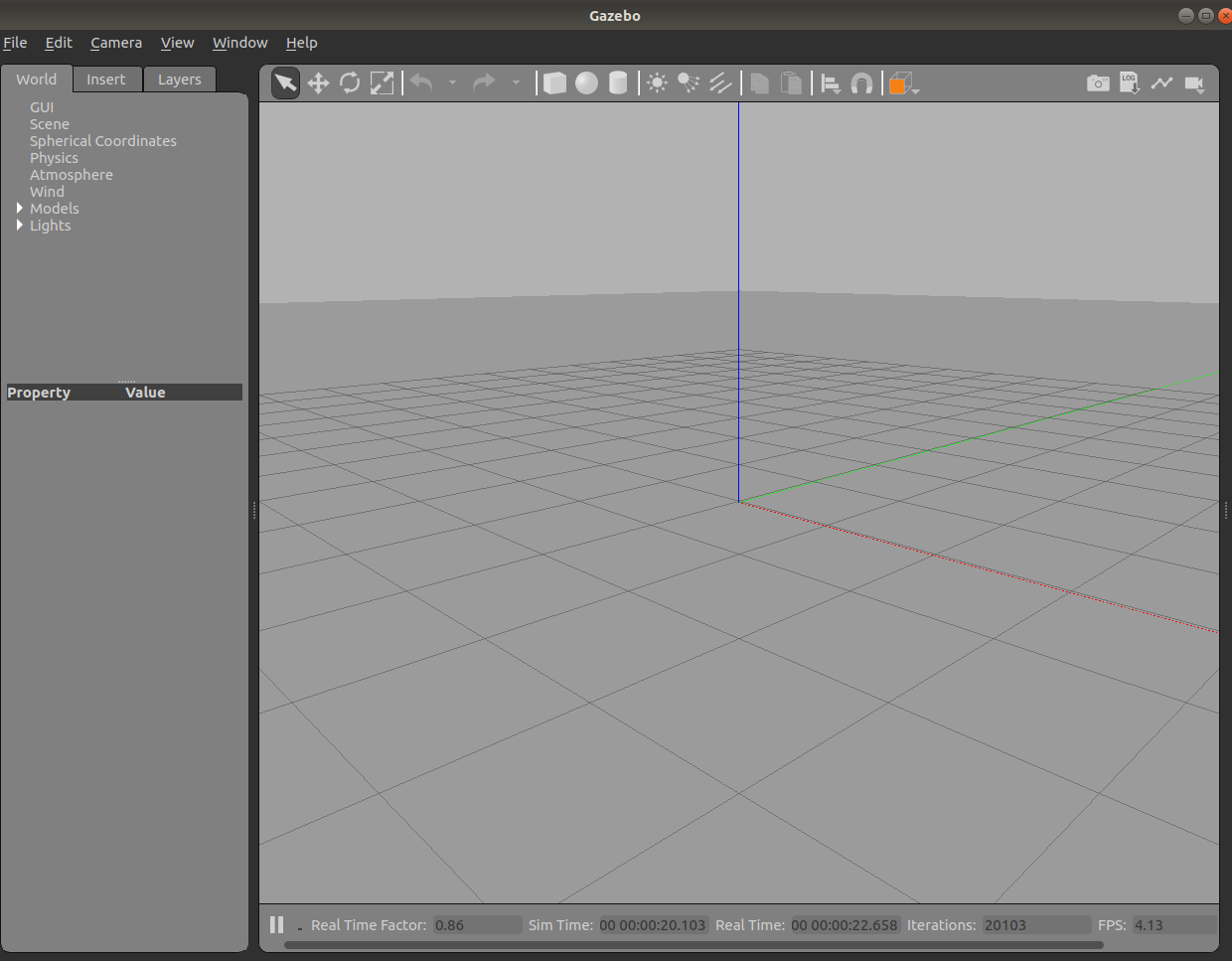


Рисунок 2.4 – Среда моделирования Gazebo

Для запуска конкретной модели используются launch-файлы. Они позволяют запустить сразу несколько служб, необходимых для функционирования робота.

|  |
| --- |
| <launch> |
|  | <arg name="use\_sim\_time" default="true" /> |
|  | <arg name="gui" default="true" /> |
|  | <arg name="headless" default="false" /> |
|  | <!-- <arg name="world\_name" default="$(find nexus\_gazebo)/worlds/nexus\_4wd\_mecanum.world" /> --> |
|  |  |
|  | <include file="$(find gazebo\_ros)/launch/empty\_world.launch"> |
|  | <arg name="debug" value="0" /> |
|  | <arg name="gui" value="$(arg gui)" /> |
|  | <arg name="use\_sim\_time" value="$(arg use\_sim\_time)" /> |
|  | <arg name="headless" value="$(arg headless)" /> |
|  | <!-- <arg name="world\_name" value="$(arg world\_name)" /> --> |
|  | <arg name="paused" value="false"/> |
|  | </include> |
|  |  |
|  | <!-- Load robot description --> |
|  | <include file="$(find nexus\_4wd\_mecanum\_description)/launch/nexus\_4wd\_mecanum\_description.launch" /> |
|  |  |
|  | <!-- Spawn the robot --> |
|  | <node name="urdf\_spawner" pkg="gazebo\_ros" type="spawn\_model" |
|  | args="-urdf -model nexus\_4wd\_mecanum -param robot\_description -x 0 -y 0 -z 0.5" /> |
|  |  |
|  | </launch> |

Приведенный код делает следующее:

1. use\_sim\_time – ноды ROS получают доступ к времени симуляции Gazebo, отправляемое в топик /clock;
2. gui – запуск графического интерфейса Gazebo;
3. headless – включения записи журнала состояния Gazebo;
4. empty\_world.launch – запуск пустого мира в Gazebo;
5. Запуск модели робота на Mecanum-колесах в нулевом положении (x = 0; y = 0; z = 0.5);

Для корректной симуляции работы робота в Gazebo необходимо модифицировать urdf – файл робота, проинициализировав трение колес:

|  |
| --- |
|  |
| <gazebo reference="upper\_left\_wheel"> |
|  | <material>Gazebo/Grey</material> |
|  | <mu1>0.0</mu1> |
|  | <mu2>0.0</mu2> |
|  | </gazebo> |
|  |  |

Для управления роботом в среде Gazebo создается специальная нода teleop\_twist\_keyboard, позволяющая отправлять задание на направление движения робота с помощью клавиатуры. Нода универсальна: она позволяет работать как с роботами на обычных колесах, так и с роботами на Mecanum колесах. Выбор типа движения осуществляется нажатием клавиши Caps Lock. Если буквы в верхнем регистре – робот движется в направлениях, доступных Mecanum колесам. Текст программы управления на языке C++: одключение всех необходимых заголовков для корректной работы ROS:

#include <ros/ros.h>

Подключение сообщения типа geometry\_twist, который состоит из линейных координат x, y, z (тип float64) и угловых координат, а также подключение остальных необходимых для работы библиотек:

#include <geometry\_msgs/Twist.h> //

#include <stdio.h>

#include <unistd.h>

#include <termios.h>

#include <map>

Далее описываются клавиши управления роботом. Буквы верхнего регистра описывают движение робота на Mecanum-колесах: “I” – движение вперед, “<” – движение назад, “J” – движение влево, “L” – движение вправо, “U”, “>”, “M”, “O” – определяют движение по диагонали. “K” – остановка.

std::map<char, std::vector<float>> moveBindings

{ {'i', {1, 0, 0, 0}}, {'o', {1, 0, 0, -1}}, {'j', {0, 0, 0, 1}},

{'l', {0, 0, 0, -1}}, {'u', {1, 0, 0, 1}}, {',', {-1, 0, 0, 0}},

{'.', {-1, 0, 0, 1}}, {'m', {-1, 0, 0, -1}}, {'O', {1, -1, 0, 0}},

{'I', {1, 0, 0, 0}}, {'J', {0, 1, 0, 0}}, {'L', {0, -1, 0, 0}},

{'U', {1, 1, 0, 0}}, {'<', {-1, 0, 0, 0}}, {'>', {-1, -1, 0, 0}},

{'M', {-1, 1, 0, 0}}, {'t', {0, 0, 1, 0}}, {'b', {0, 0, -1, 0}},

{'k', {0, 0, 0, 0}}, {'K', {0, 0, 0, 0}} };

Описываются клавиши ускорения и замедления. Так, “q” - увеличивает линейную и угловую скорость в 1.5 раза, “x” – уменьшает только линейную скорость в 0.5 раз:

std::map<char, std::vector<float>> speedBindings

{ {'q', {1.5, 1.5}}, {'z', {0.5, 0.5}}, {'w', {1.5, 1}},

{'x', {0.5, 1}}, {'e', {1, 1.5}}, {'c', {1, 0.5}} };

Установка начальных значений:

float speed(0.5); // Линейная скорость (m/s)

float turn(1.0); // Угловая скорость (rad/s)

float x(0), y(0), z(0), th(0); // Переменные направления

char key(' ');

Код основной программы с дополнительными пояснениями приведен в приложении Б.

Теперь необходимо в трех разных терминалах запустить Gazebo с помощью созданного launch файла, ноду управления с помощью команды rosrun и отслеживать данные, которые отправляются в топик /cmd\_vel:

roslaunch nexus\_4wd\_mecanum\_gazebo nexus\_4wd\_mecanum\_world.launch

rosrun teleop\_twist\_keyboard teleop\_twist\_keyboard.cpp

rostopic echo /cmd\_vel

Принцип работы представленной системы следующий: при запуске launch-файла, созданного в пункте 2.3, открывается топик /clock, в который Gazebo отправляет время симуляции, запускается пустой мир Gazebo, затем открывается графический интерфейс программы, и запускается модель робота, созданная в пункте 2.2. Модель подключается к топику /cmd\_vel, через который будет происходить ее взаимодействие с нодой управления посредством сообщений типа geometry/Twist. С помощью команды rosrun запускается нода управления в качестве издателя для топика /cmd\_vel (сообщения топика выводятся в новом терминале, в котором было запущено rostopic echo /cmd\_vel). При нажатии клавиши управления, подписчик (модель), постоянно считывающая сообщения из топика, преобразует сообщение в перемещение в среде Gazebo.

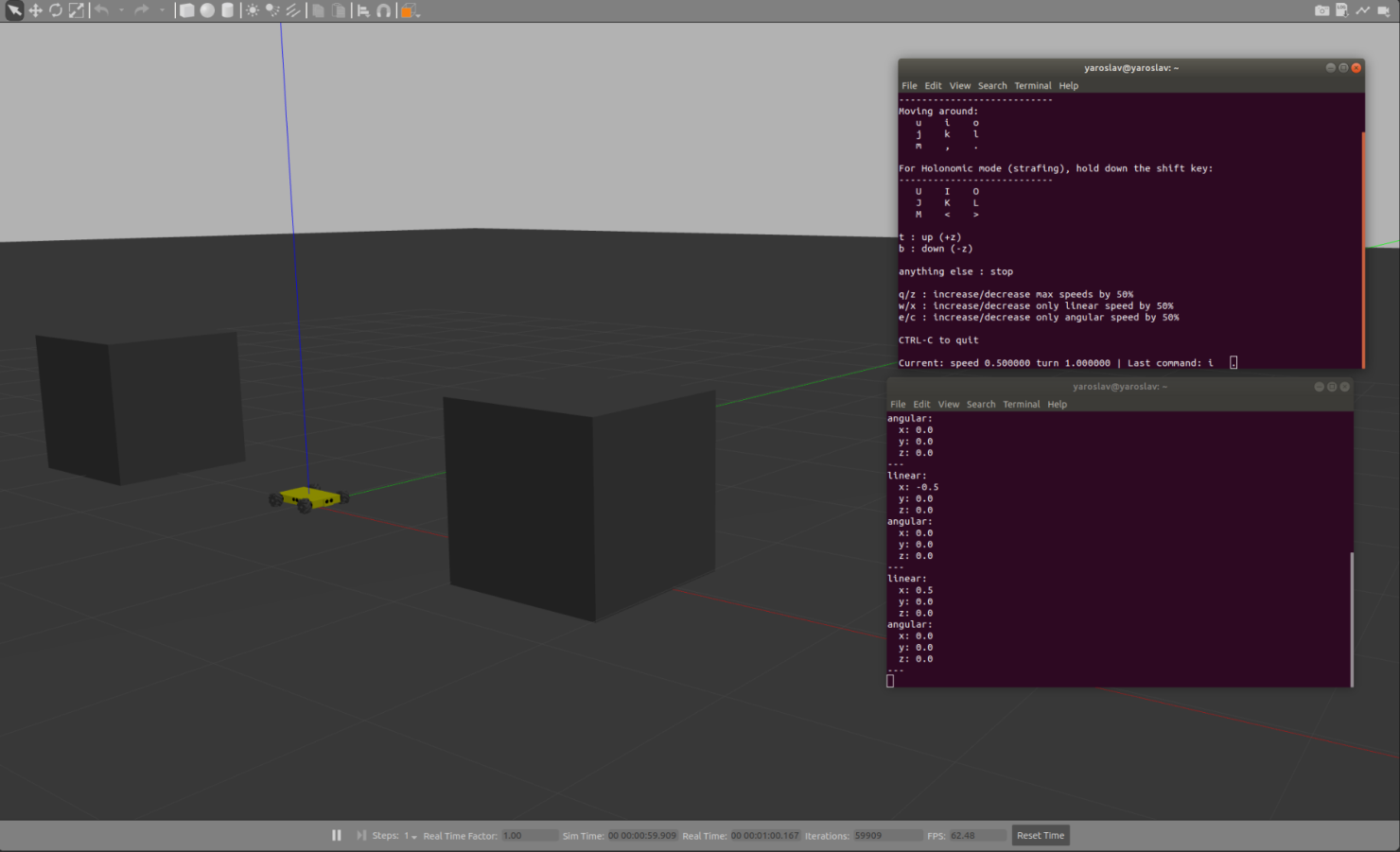


Рисунок 2.5 – Моделирование робота в среде Gazebo, управление с помощью клавиатуры, отслеживание задания

Для демонстрации работоспособности в модели были созданы два куба. При движении вперед робот сталкивается с кубом и не двигается вперед.

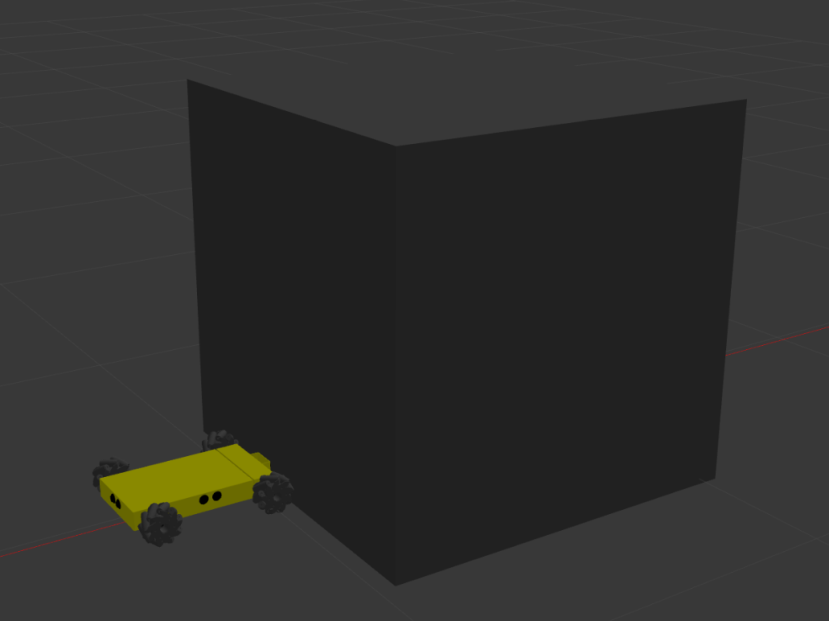


Рисунок 2.6 – Столкновение робота с препятствием

На следующих рисунках демонстрируется движение робота влево и по диагонали:

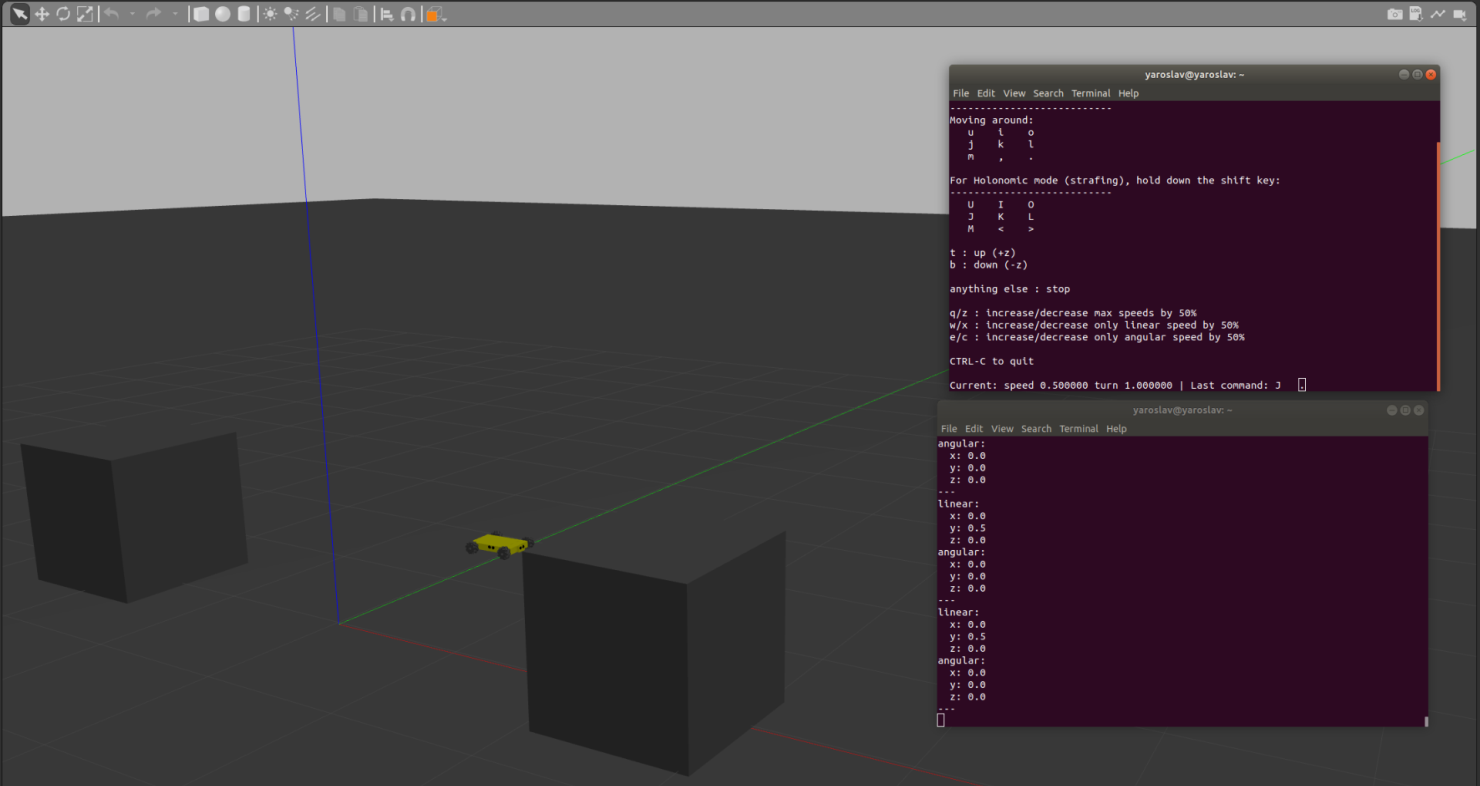


Рисунок 2.7 – Движение робота на Mecanum-колесах влево

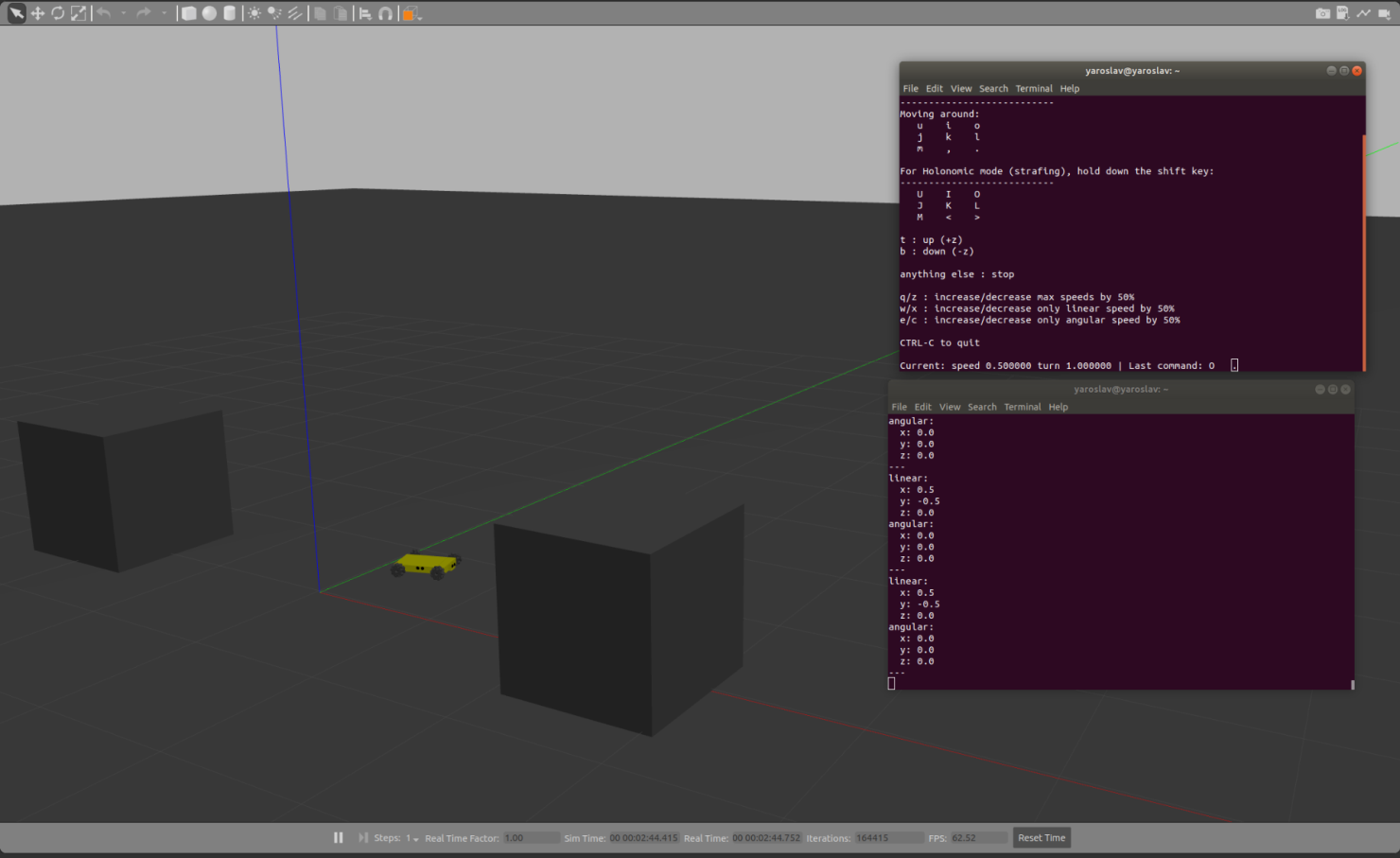


Рисунок 2.8 – Движение робота на Mecanum-колесах по диагонали

ЗАКЛЮЧЕНИЕ

Результатом курсовой работы является созданная модель робота на Mecanum-колесах для пакетов Rviz и Gazebo, а также программа управления этим роботом.

Кроме того, что ROS включает в себе такие удобные программы как Gazebo и Rviz, его использование имеет следующие преимущества: некоторые программы были взяты из интернета, без существенного их изменения, а все переменные, которые отправлялись из ноды управления в среду моделирования, отслеживались с помощью функции ROS – rostopic echo.

Моделирования робота в среде Rviz позволило создать точную модель существующего робота в формате URDF для последующего ее трансформации в формат XACRO и использования в среде моделирования Gazebo.

С помощью Gazebo была создана создана и настроена программа, позволяющая управлять роботом. То есть, для проверки работоспособности программы управления не потребовалось использовать реального робота. С помощью Gazebo также были исследованы физические свойства полученного робота. В частности, при моделировании был рассмотрен случай столкновения робота объектом, превосходящим его массу.

Приведенные пункты позволяют сделать вывод, что использование математического моделирования в инженерной практике позволяет существенно упростить процесс разработки и настройки работоспособности программы, ускорить рабочий процесс, а также не использовать настоящего робота в процессе разработок.

СПИСОК ЛИТЕРАТУРЫ

1. Колесниченко Е.Ю. Кинематическое управление движением шестиколесного меканум-робота [Текст] // Е.Ю. Колесниченко, В.Е. Павловский, Д.А. Грибков. – Москва: Препринты ИПМ им. М.В.Келдыша. 2016. № 127. 26 с.
2. Robot Geeks [Электронный ресурс]: Колеса Mecanum. URL: <https://robotgeeks.ru/collection/mecanum> (дата обращения 20.10.2021).
3. Seeed Studio [Электронный ресурс]: 4WD Mecanum Wheel Robot Kit Series. URL: <https://wiki.seeedstudio.com/4WD_Mecanum_Wheel_Robot_Kit_Series/> (дата обращения 23.10.2021).
4. Килин А.А. Управление тележкой с омниколесами на плоскости [Текст] // A.A. Килин, А. Д. Бобыкин. – Ижевск: Нелинейная динамика,  2014. **-** 10 с.
5. Yahya Tawil. An introduction to Robot Operating System (ROS) [Электронный ресурс]: All about circuits, 2017. URL: https://www.allaboutcircuits.com/technical-articles/an-introduction-to-robot-operating-system-ros/ (дата обращения 10.11.2021).
6. ROS.org [Электронный ресурс]: rviz/UserGuide. URL: <http://wiki.ros.org/rviz/UserGuide> (дата обращения 15.11.2021).
7. Cxem.net [Электронный ресурс]: Gazebo. Мощное программное обеспечение для исследования сенсорных систем робототехники. URL: <https://cxem.net/software/gazebo.php> (дата обращения 30.11.2021).

ПРИЛОЖЕНИЕ А

Перечень замечаний нормоконтролера

студента Ходаковского Я.С., группы ЭАПУ-18

|  |  |  |  |
| --- | --- | --- | --- |
| Обозначение  документа | Документ | Условная  оценка | Содержание  замечания |
|  |  |  |  |

Дата: \_\_\_\_\_\_\_\_\_ Подпись:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ФИО)

ПРИЛОЖЕНИЕ Б

Код основной программы

int main(int argc, char\*\* argv)

{ //Инициализация ноды

ros::init(argc, argv, "teleop\_twist\_keyboard");

ros::NodeHandle nh;

// Инициализация топика /cmd\_vel

ros::Publisher pub = nh.advertise<geometry\_msgs::Twist>("cmd\_vel", 1);

// Создание сообщения типа geometry\_msgs/Twist

geometry\_msgs::Twist twist;

printf("%s", msg);

printf("\rCurrent: speed %f\tturn %f | Awaiting command...\r", speed, turn);

while(true){

// Получение нажатой кнопки

key = getch();

// Если нажата кнопка управления (moveBindings)

if (moveBindings.count(key) == 1)

{

//Установка направлений

x = moveBindings[key][0];

y = moveBindings[key][1];

z = moveBindings[key][2];

th = moveBindings[key][3];

printf("\rCurrent: speed %f\tturn %f | Last command: %c ", speed, turn, key); }

// Если нажата кнопка изменения скорости (speedBindings)

else if (speedBindings.count(key) == 1)

{ speed = speed \* speedBindings[key][0];

turn = turn \* speedBindings[key][1];

printf("\rCurrent: speed %f\tturn %f | Last command: %c ", speed, turn, key); }

// В ином случае – остановиться.

else

{ x = 0; y = 0; z = 0; th = 0;

// При нажатии Ctlr+C – закрыть программу:

if (key == '\x03')

{ break; }

printf("\rCurrent: speed %f\tturn %f | Invalid command! %c", speed, turn, key); }

// Обновление сообщения Twist

twist.linear.x = x \* speed;

twist.linear.y = y \* speed;

twist.linear.z = z \* speed;

twist.angular.x = 0;

twist.angular.y = 0;

twist.angular.z = th \* turn;

// Отправить значения в /cmd\_vel и начать программу заново

pub.publish(twist);

ros::spinOnce(); }

return 0;}