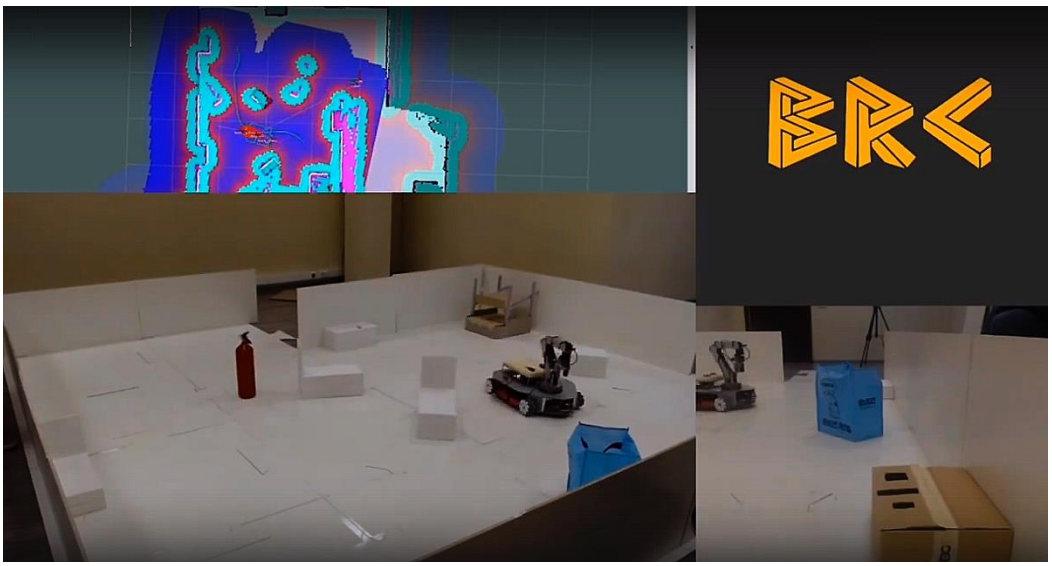
ЛАБОРАТОРНАЯ РАБОТА N 11

«Синтез управляющей модели AGV-робота при движении по заданному полигону»

Автор: Родион Анисимов

Цель работы: получение практических навыков разработки симуляционных моделей мобильных РТК в среде Gazebo фреймворка ROS



1. Теоретическая часть

В лабораторной работе создаются программные средства симуляции движения мобильных самодвижущихся роботов (AGV) в рамках соревнований Autorace Turtlebot.

1.1. Основная задача

Данные соревнования имитируют различные дорожные ситуации, и все сводится к тому, что робот должен проехать по размеченной линии параллельно выполняя различные миссии:

- остановка на светофоре, в самом начале движения (на рис. 1 видно как горит зеленый сигнал светофора);
- парковка, в одной из двух специально отведенных зон (одна из зон будет занята другим роботом);
- прохождение лабиринта (в лабиринте линии нет, поэтому он проходится при помощи лазерного дальномера).

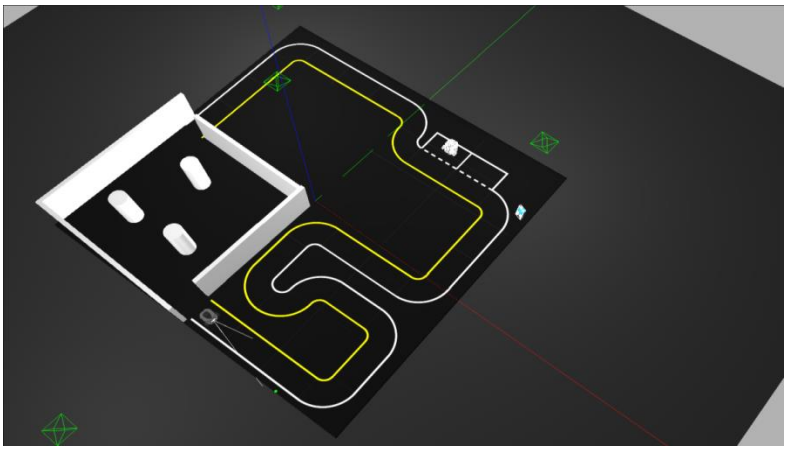


Рис. 1. Рабочее поле

Рассмотрим соревновательное поле и миссии подробнее. Поле представляет собой квадратную поверхность с нанесенными на нее желтой и белой линиями, имитирующими дорожную разметку. По

регламенту соревнований робот не должен выезжать за пределы дорожной полосы, иначе ему будут начислены штрафные баллы. Поэтому очень важно реализовать максимально точное следование линии.

1.2. Миссии

- **Остановка на светофоре**

Первой миссией является *остановка на светофоре*. Изначально на светофоре горит зеленый сигнал. Когда робот подъезжает к нему, зеленый сигнал сменяется красным и робот должен остановиться. Затем робот должен ждать повторного появления зеленого сигнала и только после этого продолжать движение. Пример того, что видит робот при подъезде к светофору, приведен на рис.2:

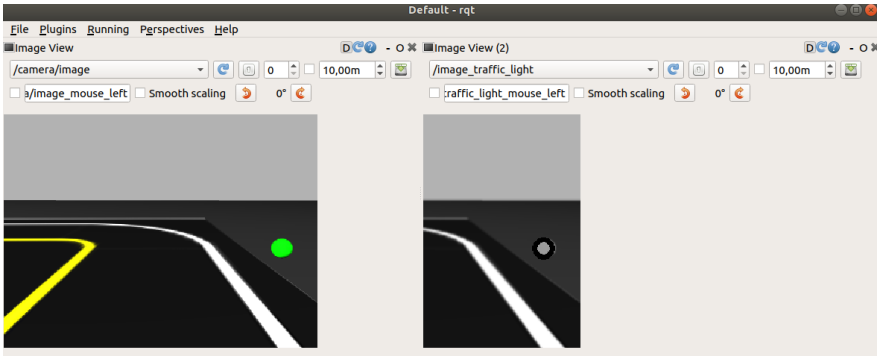


Рис. 2. Приближение к светофору

- **Парковка**

Второй миссией, которую робот должен пройти является *парковка*. В рамках данной миссии, робот должен заехать в одно из двух отведенных мест парковки (другое место занимает второй робот). В качестве ориентира для начала миссии используется знак парковки (рис. 3):

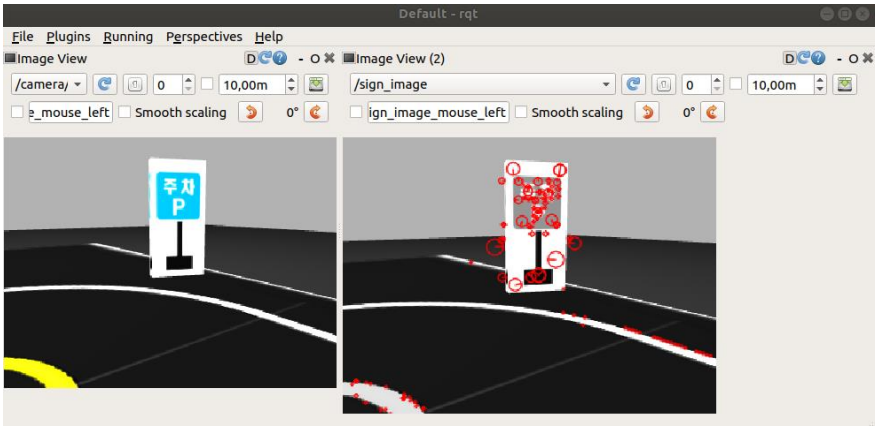


Рис. 3. Парковка

- **Туннель**

Последней миссией для робота является *замкнутый лабиринт*, имеющий один вход и один выход. Для ориентации в тоннеле робот не может использовать камеры, но может использовать остальные имеющиеся датчики. Главная задача - выехать из туннеля через зону выхода. Зона входа же в свою очередь помечена знаком (рис. 4):

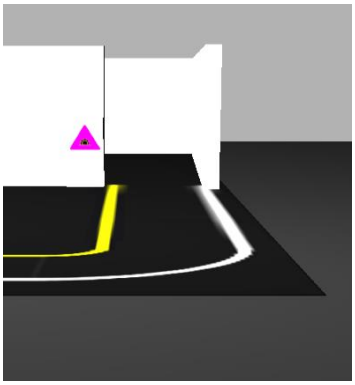


Рис. 4. Вход в туннель

Конечной целью робота является прохождение всех миссий за наименьшее время. Робот оснащен следующим набором сенсоров:

- лазерным дальномером;
- инерциальным датчиком;
- телекамерой tracking_cam3;
- энкодерами (средствами колесной одометрии).

2. Практическая часть

В лабораторной работе необходимо создать комплекс программных средств симулирующих выполнение всех миссий мобильного робота с дифференциальной колесной схемой в среде Gazebo фреймворка ROS.

2.1. Задание на лабораторную работу

1. Запустить симуляционный полигон робота TurtleBot3.
2. Убедиться в правильности работы симуляционного полигона.
3. Визуализировать показания датчиков робота.
4. Запустить программу детектирования дорожной разметки.
5. Реализовать алгоритм движения робота по линии при помощи СТЗ.
6. Реализовать алгоритм движения робота вдоль стены при помощи лазерного дальномера.

2.2. Программные средства

Основным программным средством, использующимся в лабораторной работе, является фреймворк ROS (robot operating system), который позволяет строить архитектуру ПО робота, используя множество готовых модулей. В данном фреймворке управляющие программы - ноды, могут передавать друг другу сообщения с помощью буферов - топиков. Так любая нода может публиковать сообщения в любой топик, а также читать сообщения из него. Это позволяет сделать сделать систему управления максимально вариативной и распределенной. Также каждый топик имеет свое регламентированное сообщение, что позволяет формализировать общение между нодами.

Gazebo 3D, разрабатываемый некоммерческой организацией OSRF (Open Source Robotics Foundation), имеет ряд преимуществ по сравнению с другими робототехническими симуляторами. Во-первых, он бесплатный и имеет открытый код. Во-вторых, он очень популярен среди мирового робототехнического сообщества и является официальным симулятором соревнований DARPA. В-третьих, Gazebo отлично интегрируется с программной платформой ROS (Robot Operating System), а значит разработанную программу управления виртуальным роботом в Gazebo и ROS будет относительно несложно перенести на реального робота.

При симуляции используются 3 пакета:

- пакет для симуляции соревнований autorace с использованием tracking_cam3 (также возможен запуск без tracking_cam3);
- пакет, содержащий стандартные симуляционные пакеты для робота Turtlebot3;
- пакет, содержащий симуляцию манипуляционных роботов Applied Robotics.

2.2.1. Стандартные пакеты Turtlebot3

Стандартные пакеты для Turtlebot3 содержат в себе набор симуляционных полигонов, с помощью которых можно тестировать различные SLAM и навигационные алгоритмы. На выбор представлено 3 поля.

- empty_world – пустой полигон, включающий в себя запуск только робота (рис. 5):

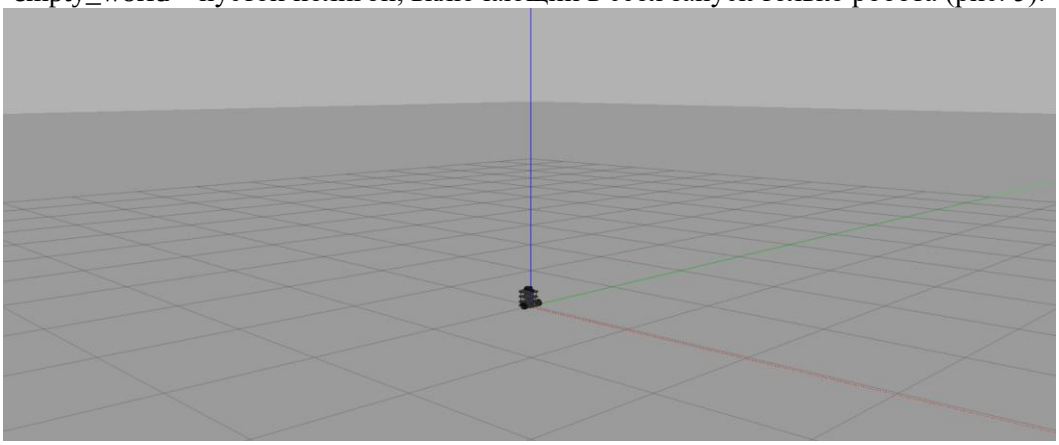


Рис. 5. Пустой полигон

- house – симуляционный полигон, имитирующий жилое помещение (рис. 6):

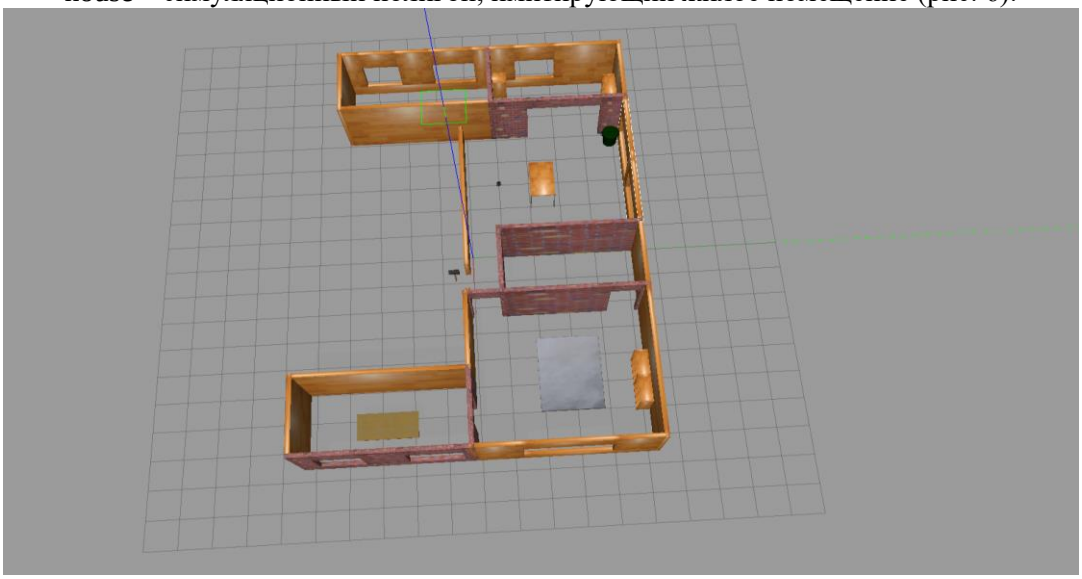


Рис. 6. Жилое помещение

- world – симуляционный полигон, со стандартным полем для Turtlebot3 (рис. 7):

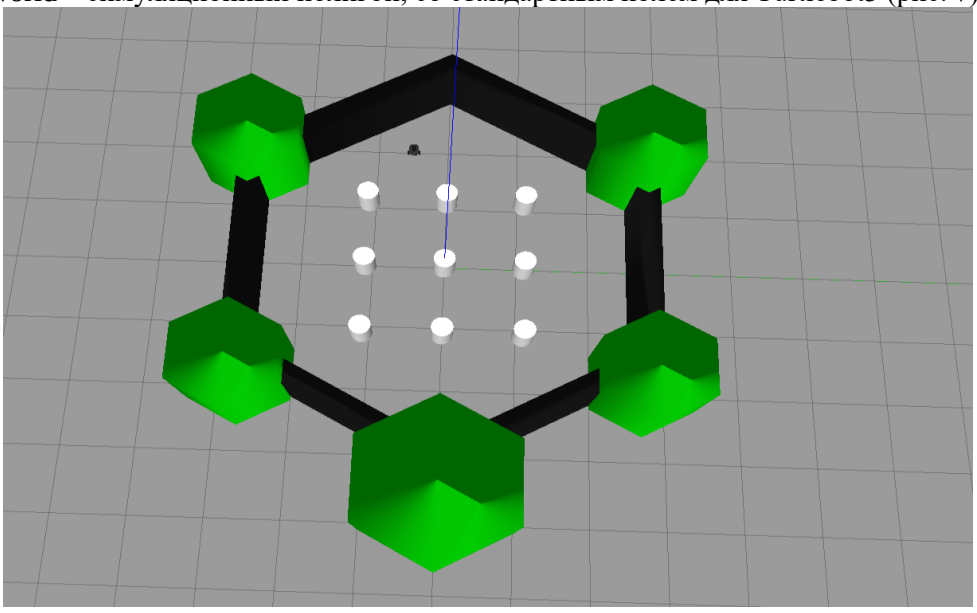


Рис. 7. Стандартное поле Turtlebot3

Для более детального ознакомления обратитесь к: https://emanual.robotis.com/docs/en/platform/turtlebot3/simulation/#turtlebot3-simulation-using-gazebo

Все предустановленные стандартные пакеты Turtlebot и симуляции Turtlebot находятся в папке turtlebot3_original_ws в домашней директории виртуальной машины.

2.2.2. Autorace

Данную симуляцию можно запустить как с применением tracking_cam3 и так и без нее. При этом в интерфейсе tracking_cam3 уже реализованы алгоритм детектирования линии и нахождения отклонения робота от центра линии, а также алгоритм детектирования светофора и дорожных знаков.

- **Запуск с использованием tracking_cam3**

Все пакеты *управления и визуализации* роботов располагаются в папке workspace. На данной виртуальной машине папка workspace называется tb_ws. Там располагаются папки, отвечающие за сборку проекта (devel и build), а также папка src, в которой непосредственно находятся пакеты визуализации и управления. Структура пакетов в данном случае имеет следующий вид (рис. 8):

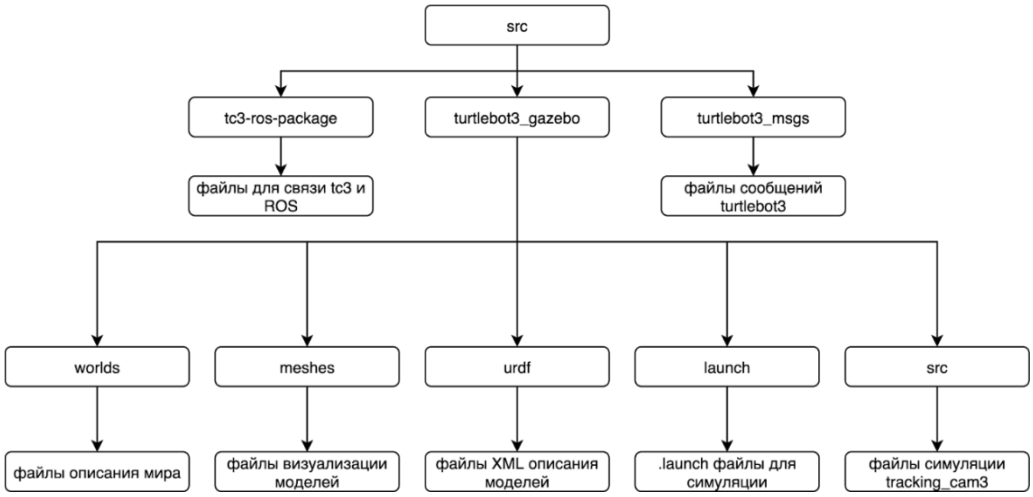


Рис. 8. Структура пакетов управления и визуализации

В первую очередь необходимо запустить виртуальную tracking_cam3 для этого в домашней директории выполняется команда:

./start_tracking_cam.sh

В терминале потребуется ввести пароль: 1234

После этого в новом терминале, перейти в папку tb_ws

cd tb_ws

После чего проинициализировать текущий workspace

source devel/setup.bash

Затем для запуска Gazebo

roslaunch turtlebot3_gazebo turlebot3_autorace.launch tc3:=true

В данном случае аргумент tc3:=true необходим для того, чтобы активировать программы, отвечающие за взаимодействие tracking_cam3 и ROS.

Для того чтобы активировать алгоритм детектирования светофоров и дорожных знаков необходимо зайти в браузер и в адресной строке вписать

localhost

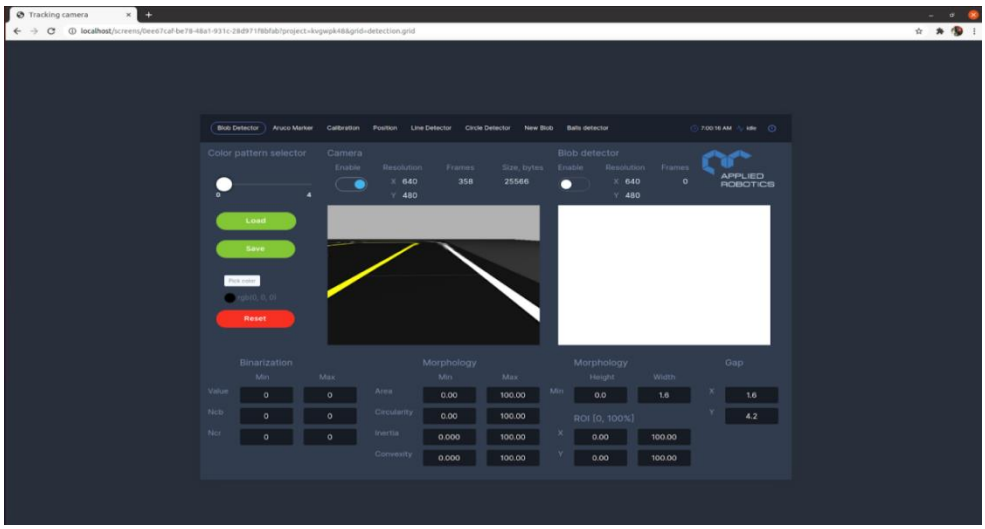


Рис. 9. Интерфейс настройки trackingCam3

После этого необходимо перейти во вкладку New Blob и нажать на переключатель Enable

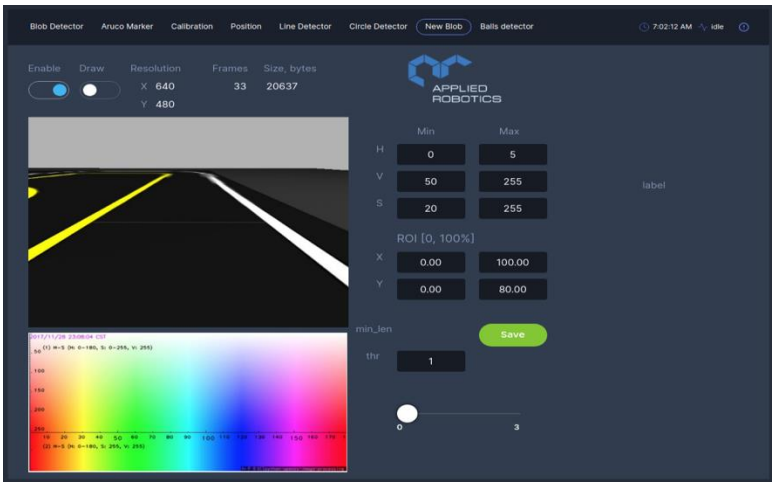


Рис. 10. Настройка алгоритма детектирования цветовых блобов

Для того чтобы активировать миссии (появление светофора, занятие одного из парковочных мест) необходимо перейти в директории tb_ws проинициализировать setup.bash файл и выполнить следующую команду

roslaunch turtlebot3_gazebo turtlebot3_autorace_mission.launch

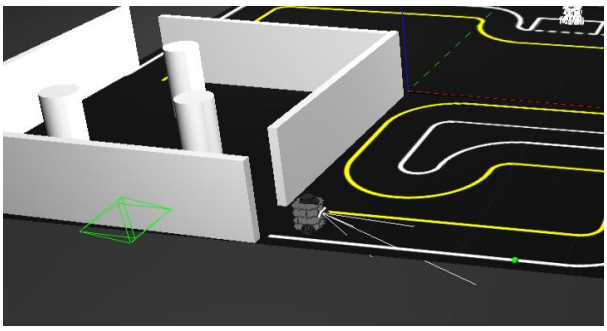


Рис. 11 Поле с запущенными миссиями

В ROS для взаимодействия с системой управления используется система *топиков*, каждый топик имеет свое специфическое сообщение. Рассмотрим топики робота Turtlebot3. Для того чтобы вывести все топики робота можно использовать команду

rostopic list

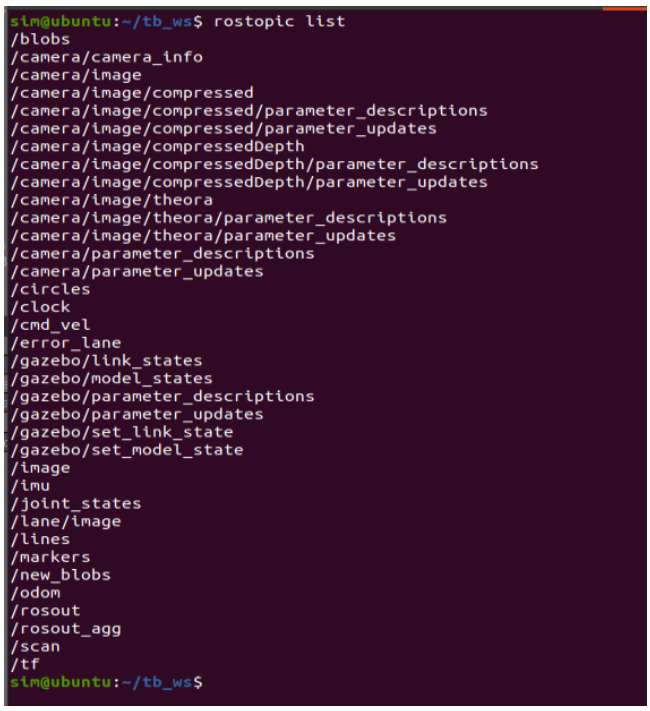


Рис. 12. Топики генерируемые средой симуляции

Рассмотрим один из топиков, генерируемых датчиками робота:
scan – топик лазерного дальномера, использует сообщение

sensor_msgs/LaserScan

Для того чтобы посмотреть какие данные приходят в топик используется команда

rostopic echo /имя топика

Например, rostopic echo /scan выдает следующее (рис. 13):

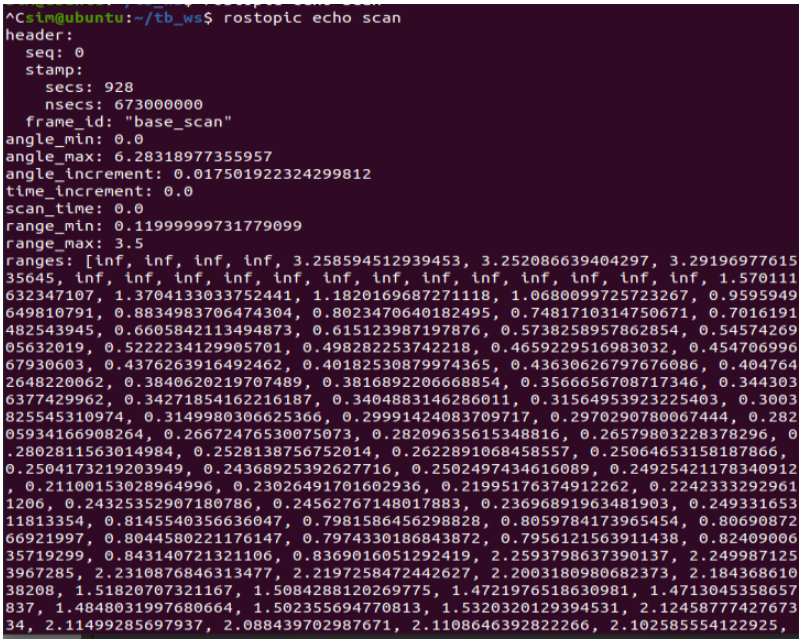


Рис. 13. Показания лазерного дальномера

Для того чтобы узнать структуру конкретного ROS сообщения можно воспользоваться командой

rosmsg show имя сообщения

Например, rosmsg show sensor_msgs/LaserScan

```
sim@ubuntu:~/tb_ws$ rosmsg show sensor_msgs/LaserScan
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
float32 angle_min
float32 angle_max
float32 angle_increment
float32 time_increment
float32 scan_time
float32 range_min
float32 range_max
float32[] ranges
float32[] intensities
```

Рис. 14. Структура сообщения LaserScan

Кроме команд rostopic echo и rosmsg для визуализации данных можно использовать утилиту rqt, а именно topic monitor. Для вызова утилиты используется команда

rqt

Рис. 15. Утилита rqt

Для запуска topic monitor необходимо перейти в Plugins -> Topics -> Topic Monitor.

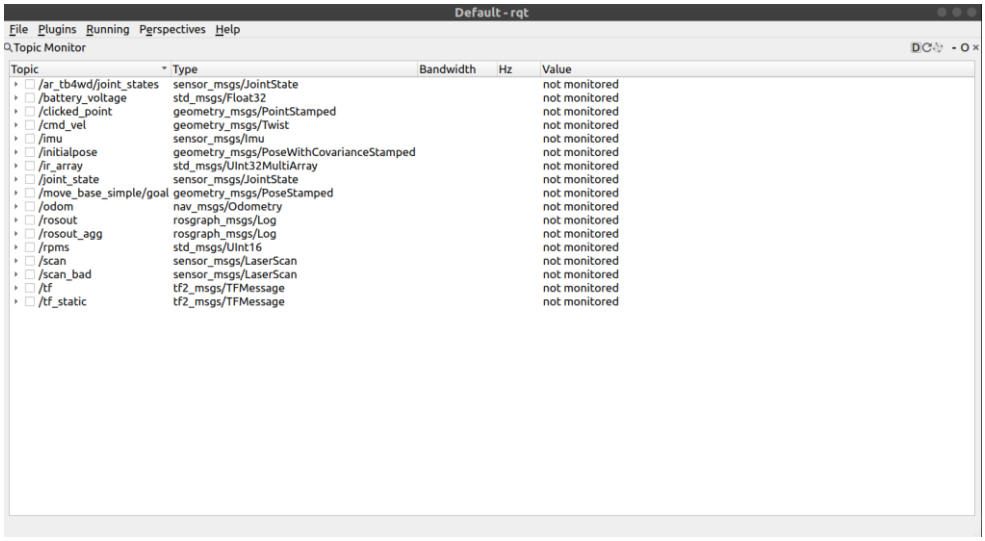


Рис. 16. Активные топики системы визуализированные в rqt

Для более детального отображения информации из топика необходимо поставить галочку для визуализации (рис. 17):

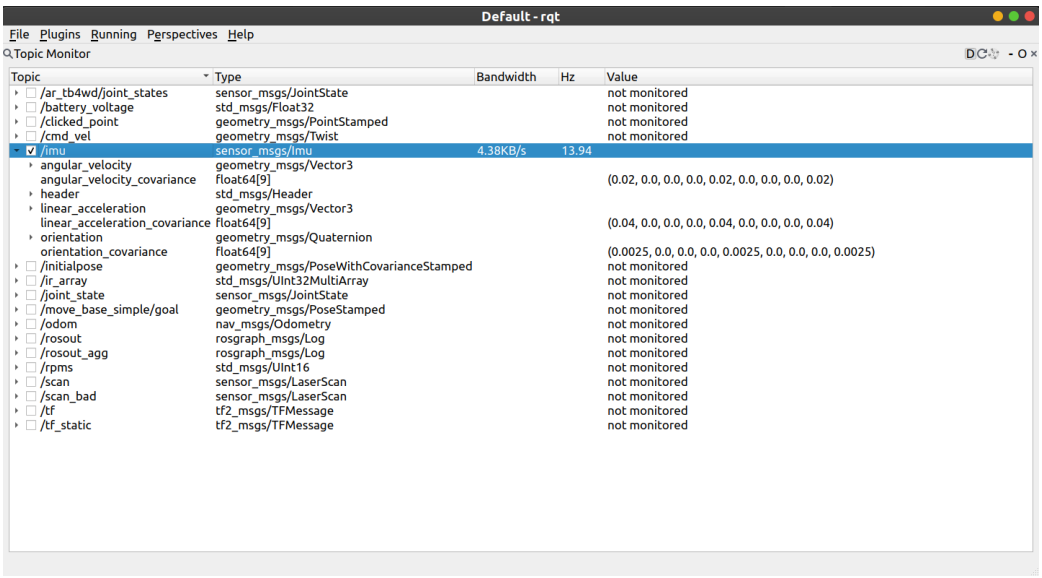


Рис. 17. Раскрытые показания топика IMU (топик инерциального датчика)

- /odom содержит информацию о *колесной одометрии* (рис. 18):

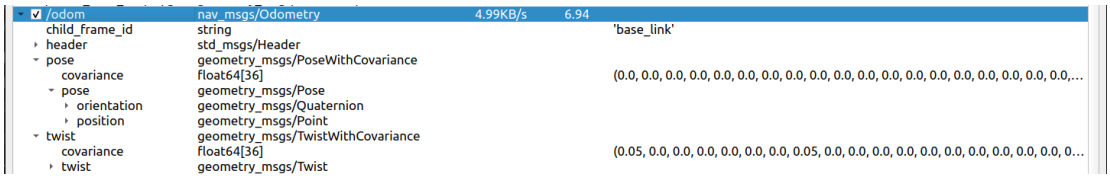


Рис. 18. Показания колесной одометрии

- /scan содержит информацию о показаниях *лазерного дальномера* (рис. 19):

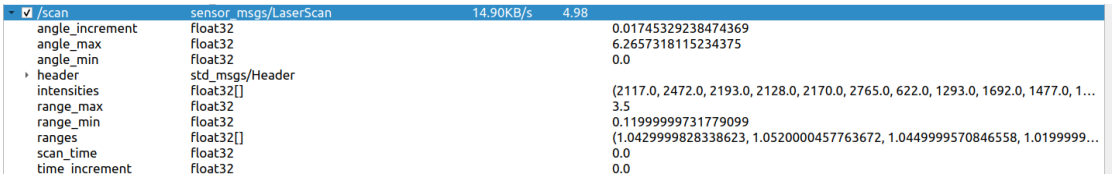


Рис. 19. Показания лазерного дальномера

- /joint_state содержит информацию о реальных *скоростях каждого из моторов* (рис. 20):

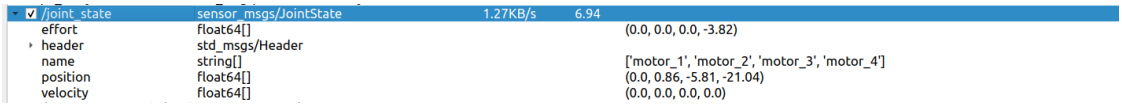


Рис. 20. Показания скоростей моторов

- /marker /blobs /lines /circles /new_blobs содержат *визуальную информацию* о детектируемых объектах с помощью tracking_cam.

Примечание

В данном случае new_blob является уже настроенным алгоритмом, и в топик new_blobs публикуется информация о детектируемых сигналах светофора и дорожных знаках.

Топик new_blobs использует сообщение blobs_msgs/BlobArray, которое является массивом сообщений blob_msgs/Blob. Структура сообщения Blob выглядит следующим образом (рис. 21):

```
^Csim@ubuntu:~/tb_ws/src/tc3-ros-package/tc3-ros-package$ rosmsg show blobs_msgs
/Blob
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
uint32 id
geometry_msgs/Point pose
  float64 x
  float64 y
  float64 z
```

Рис. 21. Структура сообщения Blob

Здесь id – порядковый номер детектируемого блоба. Знаки имеют следующие порядковые номера:

0 – зеленый сигнал светофора;

1 – красный сигнал светофора;

2 – знак парковки;

3 – знак туннеля.

x и **y** – координаты центра блоба (в пикселях камеры).

Например, при детектировании зеленого сигнала светофора в топик публикуются следующие сообщения (рис. 22):

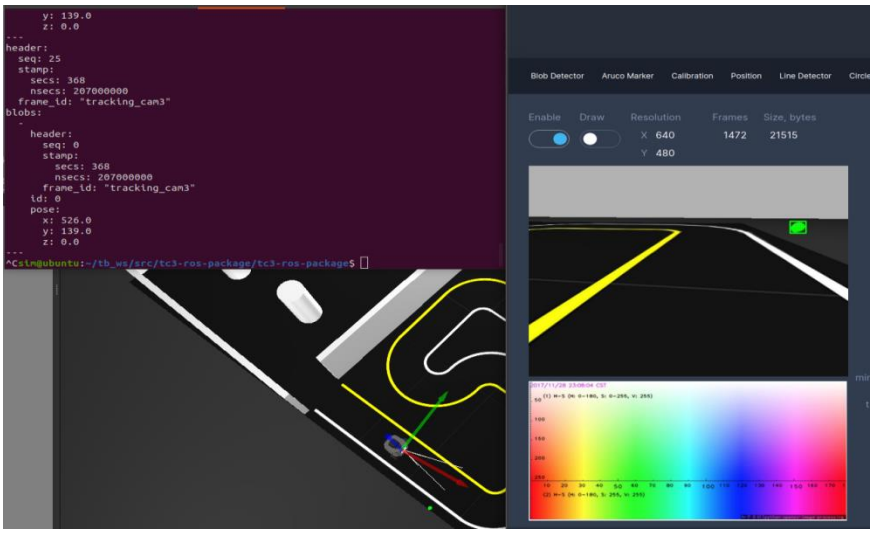


Рис. 22. Сообщения о детектировании зеленого сигнала светофора

Кроме этого, tracking_cam3 реализует алгоритм детектирования дорожной разметки и нахождение отклонения робота от центра линии.

Топик /lane/image содержит в себе *изображение* с результатом детектирования дорожной разметки (рис. 23).

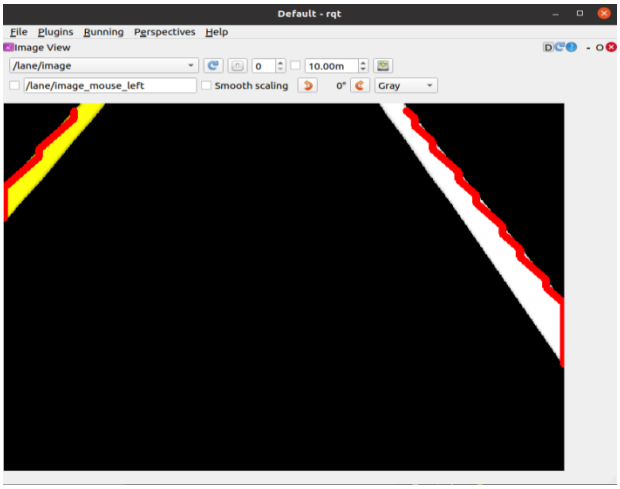


Рис. 23. Детектируемая дорожная разметка

Топик /error_lane содержит информацию о *текущем отклонении* робота от центра линии (в пикселях, рис. 24)

```
sim@ubuntu:~$ rostopic echo /error_lane
data: -80.70566666666666
---
data: -82.65283333333333
---
data: -83.47136666666665
---
data: -84.31256666666667
---
```

Рис. 24. Текущее отклонение робота от центра линии

Управление моделью робота осуществляется при помощи задания необходимой линейной и угловой скорости движения. В топик **cmd_vel**, публикуется сообщение типа geometry_msgs/Twist, которое состоит из двух векторов:

linear (линейная скорость, м/с):

x – по оси **X**;
y – по оси **Y**;
z – по оси **Z**.

angular (угловая скорость, т.е. скорость вращения вокруг своей оси, рад/с):

x – вокруг оси **X**;
y – вокруг оси **Y**;
z – вокруг оси **Z**.

Оси координат модели робота, расположены следующим образом (рис. 25):

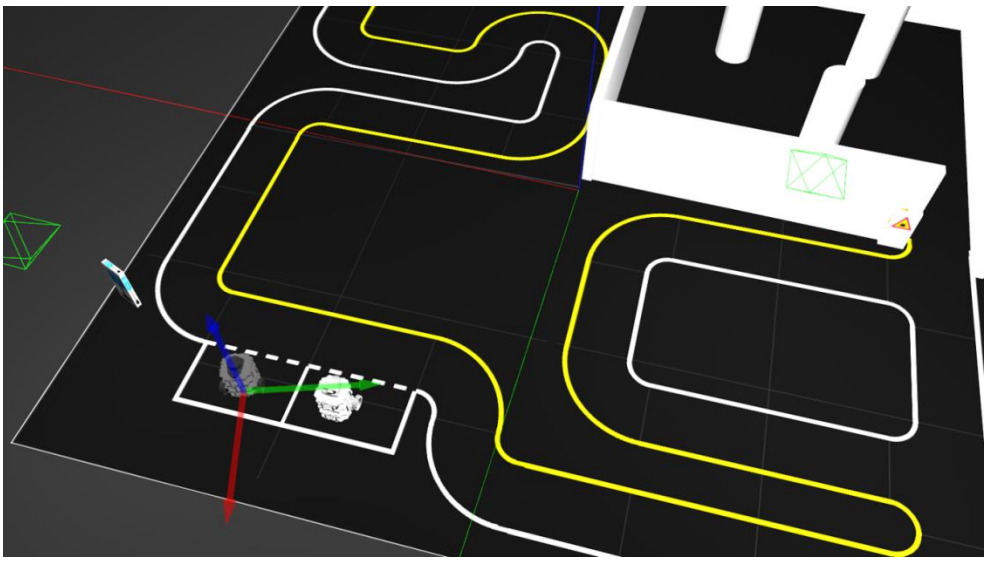


Рис. 25. Локальная СК робота

Здесь синяя ось – ось **Z**, красная – ось **X**, зеленая – ось **Y**.

Из-за того что робот Turtlebot имеет дифференциальную колесную базу, он может перемещаться только вдоль оси **X** и вокруг оси **Z**. Соответственно, для того, чтобы робот поехал вперед, необходимо отправить следующее сообщение:

msg:
linear:
x = 0.1
y = 0
z = 0
angular:
x = 0
y = 0
z = 0

Для того чтобы робот крутился на месте:

msg:
linear:
x = 0
y = 0
z = 0
angular:
x = 0
y = 0
z = 0.1

А для того чтобы он крутился по дуге:

msg:
linear:
x = 0.1

y = 0
z = 0
angular:
x = 0
y = 0
z = 0.1

Для лучшего понимания принципа управления роботом рекомендуется ознакомиться с программой, расположение программы: tb_ws/src/example/src/line_controll.py, в которой реализован простейший алгоритм движения по линии. Для запуска программы необходимо перейти в папку /tb_ws после чего выполнить две команды:

source devel/setup.bash

rosrun example line_controll.py

- **Запуск без tracking_cam3**

В отличие от режима с использованием утилиты tracking_cam3 здесь нет реализованных алгоритмов детектирования дорожной разметки, сигналов светофора и дорожных знаков. Поэтому для выполнения задания потребуется реализовать их самостоятельно

Сначала необходимо перейти в папку tb_ws

cd tb_ws

После чего проинициализировать текущий workspace

source devel/setup.bash

Затем для запуска Gazebo

roslaunch turtlebot3_gazebo turlebot3_autorace.launch

Запустится симулятор, но без сервисов tracking_cam3.

Для доступа к изображению с телекамеры, установленной на роботе, можно воспользоваться топиком /camera/image (рис. 26):

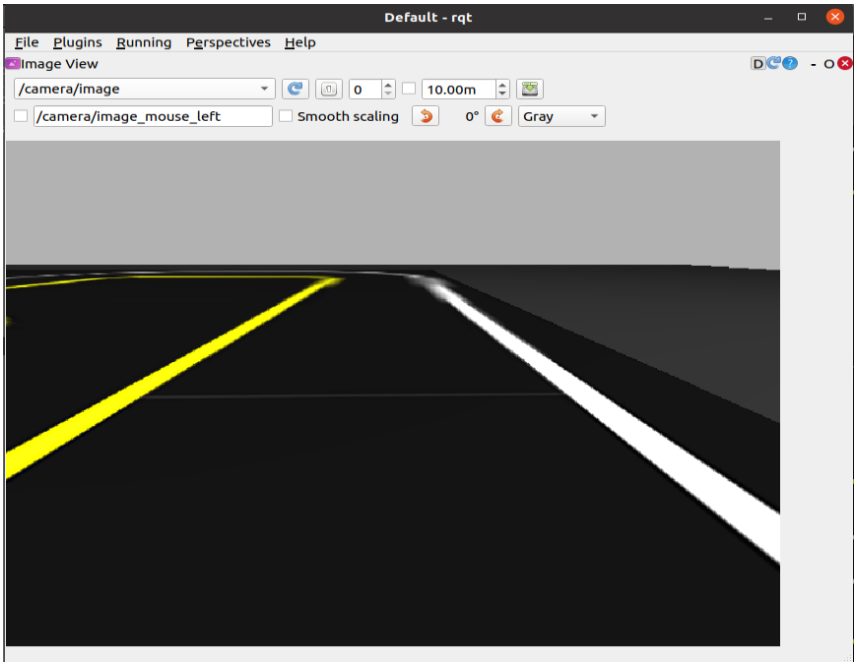


Рис. 26. Изображение получаемое с камеры установленной на робота

Таким образом студент должен ознакомиться с фреймворком ROS и соревнованиями turtlebot3 autorace, а также проверить работу алгоритма движения по линии и на его примере разработать алгоритм прохождения лабиринта на языке python. После чего предоставить отчет о проделанной работе.

2.3. Представление отчета по лабораторной работе

В отчет по лабораторной работе должны входить:

- скриншоты запущенной симуляции и визуализации показаний датчиков;

- описание алгоритма движение робота по линии, листинг программы;
- описание алгоритма движения робота вдоль стены, листинг программы;
- видеозапись работы алгоритмов.

3. ПРИЛОЖЕНИЕ

3.1. Установка и настройка ПО

Для полноценной работы с фреймворком ROS необходим какой либо дистрибутив операционной системы с ядром linux, для этого можно использовать виртуальную машину. То есть, необходимо установить операционную систему внутри операционной системы, для этого понадобится программа virualBox, которую можно установить и скачать по ссылке: https://www.virtualbox.org/.

Также необходимо скачать сам образ операционной системы с предустановленным ROS, который можно будет открыть с помощью программы virtualBox. Для этого необходимо скачать архив по ссылке:

https://drive.google.com/file/d/1FB8gXQvN0vyorsEFLbHAUPr8yaHhQ7Ur/view?usp=sharing

Возможно, потребуется сохранить файл на личный Яндекс диск для скачивания.

После скачивания его нужно разархивировать в удобную вам папку. После разархивирования можно увидеть следующую структуру папок и файлов:

Ubuntu 64-bit
Ubuntu 64-bit
Файлы виртуальной машины

Данные файлы нельзя как-либо модифицировать и изменять во избежание всевозможных поломок ПО. В дальнейшем с помощью программы virtualBox будет открываться файл Ubuntu 64-bit с расширением Virtual Machine Disk Format (.vmdk. рис. 1)

Рис. 1. Структура файлов операционной системы Ubuntu

3.2. Настройка

Для первого запуска виртуальной машины необходимо:

1. Запустить программу virtualBox
2. Нажать кнопку создать, чтобы создать новую виртуальную машину (рис. 2):

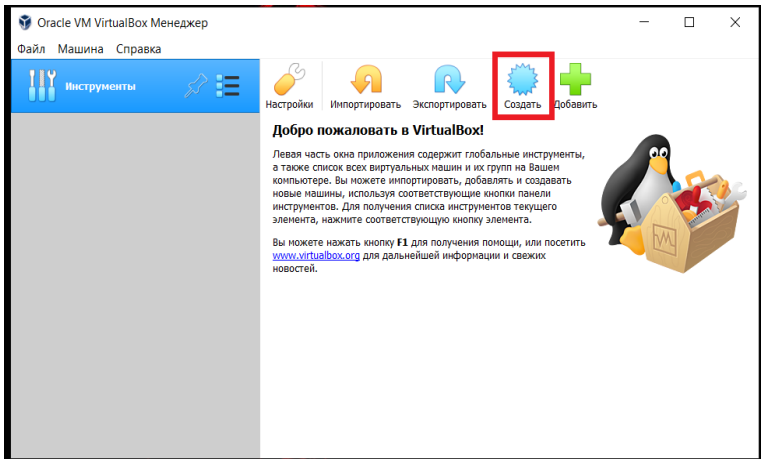


Рис. 2. Создание виртуальной машины

3. В высветившемся окне нажать кнопку экспертный режим (рис. 3)

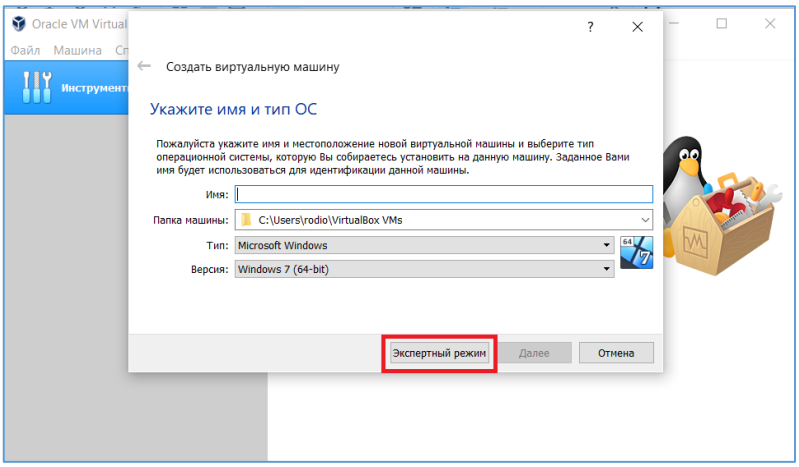
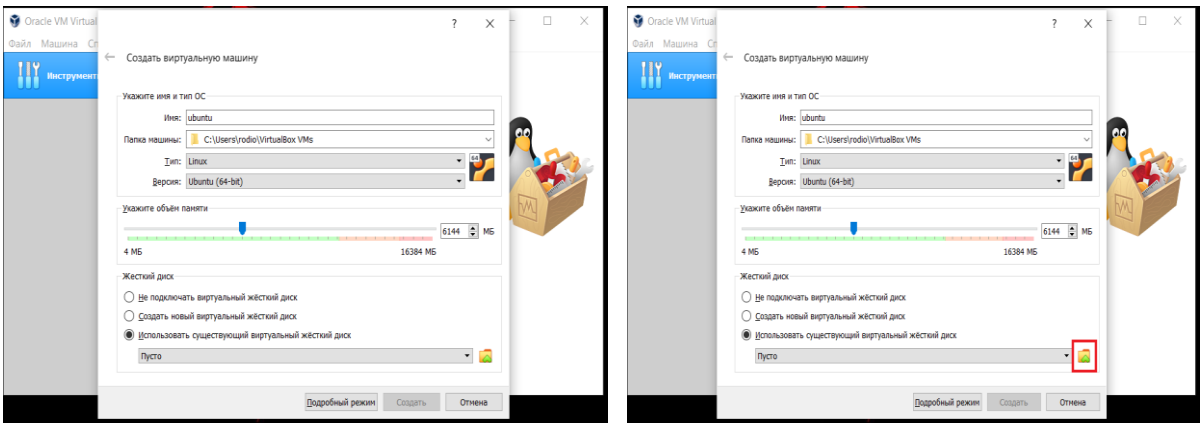


Рис. 3. Задание экспертного режима

4. Настраиваем виртуальную машину:
   4.1. Задать любое имя машины
   4.2. Выбрать папку, в которой будут храниться данные виртуальной машины, предложенную папку можно не изменять
   4.3. Тип машины - Linux, версия Ubuntu (64-bit)
   4.4. Выбрать количество оперативной памяти, которое будет доступно виртуальной машине. ДЛЯ КОРРЕКТНОЙ РАБОТЫ НЕОБХОДИМО НЕ МЕНЕЕ 6 ГБ (рис. 4, *а*)



а *б*

Рис. 4. Установка объема оперативной памяти

   4.5. В графе жесткий диск выбрать пункт **использовать существующий виртуальный диск**, и в качестве виртуального диска выбрать файл **Ubuntu 64-bit.vmdk** описанный в предыдущем пункте (рис. 4, *б*)

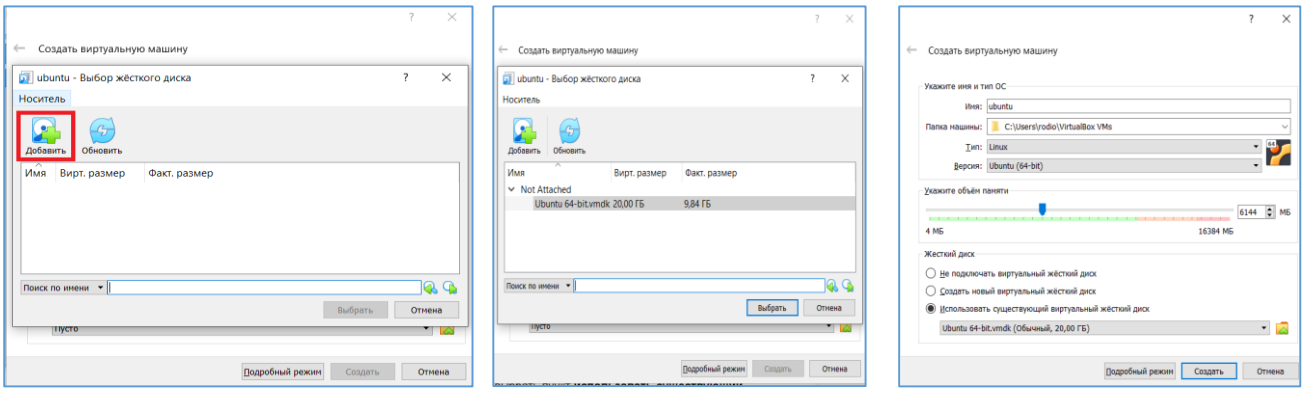
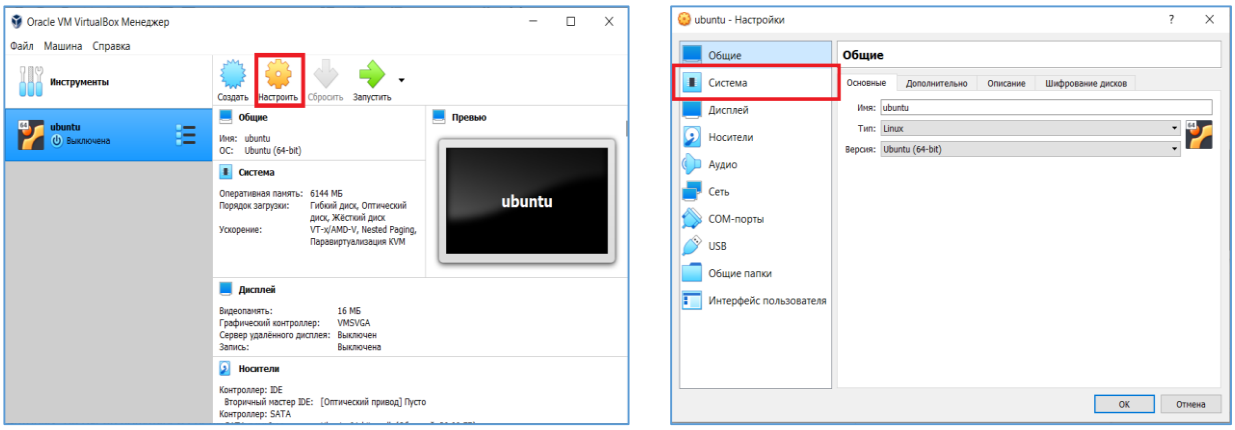


Рис. 5. Продолжение настройки

5. Когда все настройки выставлены в соответствии с пунктом 4, необходимо нажать кнопку «Создать».

6. Перед запуском виртуальной машины, необходимо настроить количество **ядер процессора**, которые она сможет использовать (для комфортной работы рекомендуется не менее 4). Для этого:
    6.1. Необходимо нажать кнопку «Настроить» (рис. 6, *а*):



а *б*

Рис. 6. Настройка виртуальной машины

    6.2. Перейти в графу «Система» (рис. 6, *б*)
    6.3. Перейти во вкладку процессор и разрешить использование 4 или более ядер (количество разрешенных к использованию ядер зависит от общего количества ядер процессора, рис. 7)

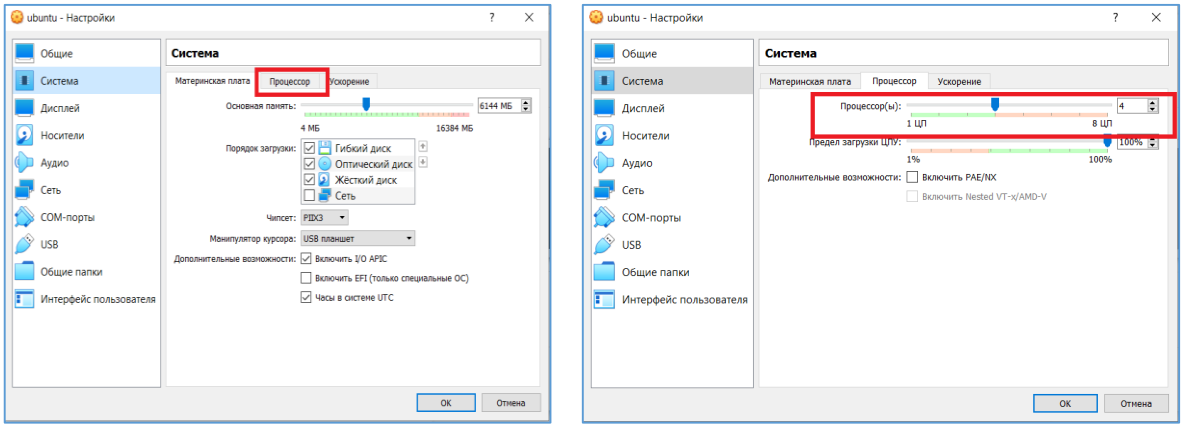


Рис. 7. Настройка процессора

7. После всех настроек можно запустить виртуальную машину, для этого необходимо:
    7.1. Нажать кнопку «Запустить» (рис. 8)

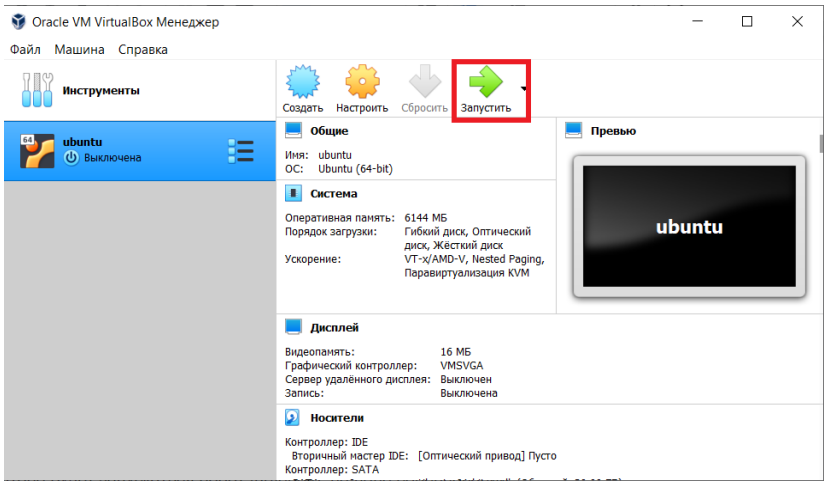


Рис. 8. Запуск виртуальной машины

    7.2. После загрузки системы появится следующее окно (рис. 9):

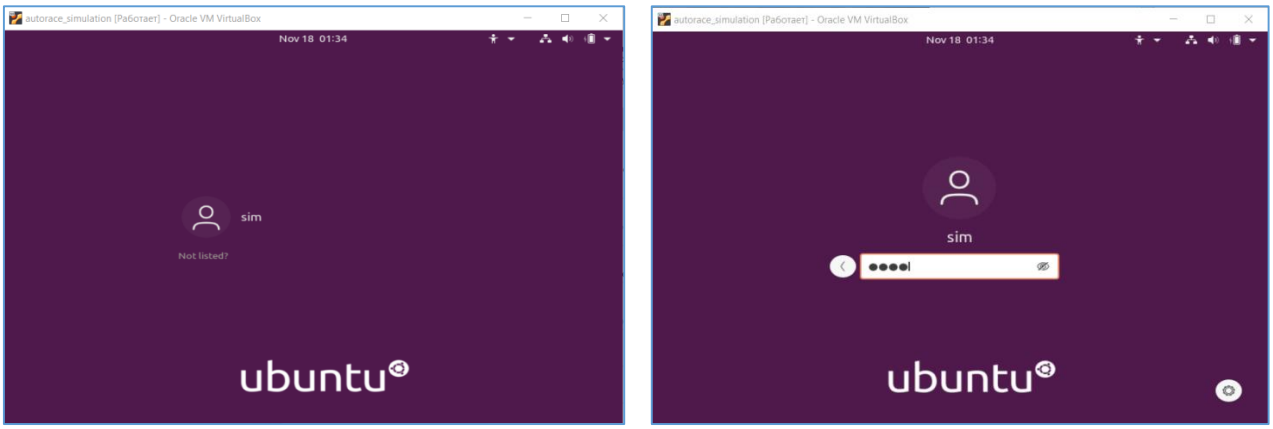
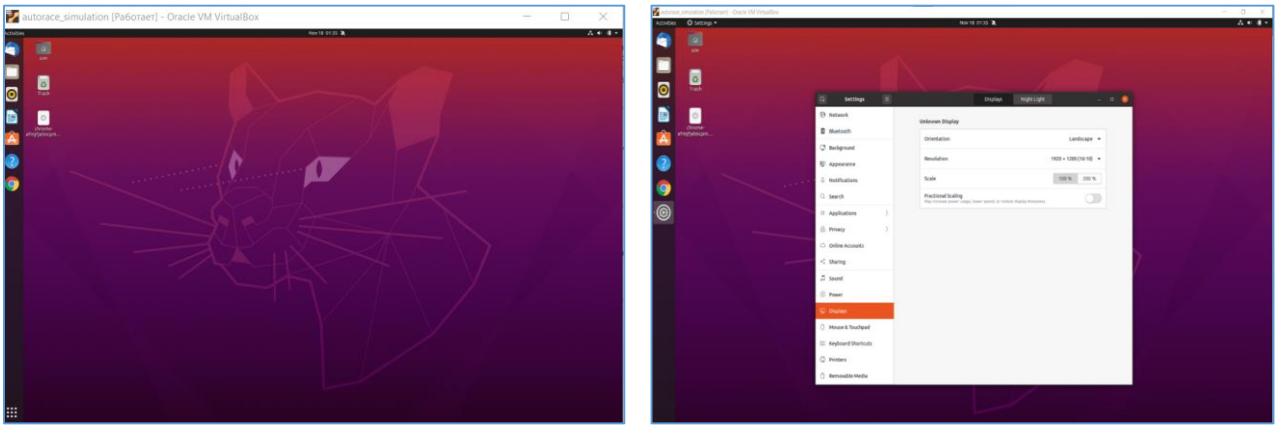


Рис. 9. Запуск операционной системы

7.3. В данном окне нажать на пользователя tb3, после чего высветится окно для ввода пароля:

7.4. Ввести пароль**1234**, после чего откроется окно системы (рис. 10):



а *б*

Рис. 10. Рабочий стол

Примечание

Если окно системы слишком мало для работы, то в настройках Ubuntu необходимо изменить разрешение экрана (рис. 10, *б*).

Теперь внутри данного окна можно вести полноценную работу с операционной системой Ubuntu.

3.2 Дополнительная информация

Описание алгоритмов детектирования и управления:
https://docs.google.com/document/d/1Wm3BlB8vyepEvrPMd2eWsPkqEQTOpcxji4aelQQ3k1c/edit?usp=sharing

Видеоуроки:

- https://youtube.com/playlist?list=PL4d02zZfaa7ZY973JE7MsShmcKt8xYzc4
- https://youtube.com/playlist?list=PL4d02zZfaa7YB6ZDGsNy9v5iWCljpwMxk
- https://youtu.be/G4QjCty0UnI