Выполнили студенты группы КРБО-01-19

Минина Полина

Валенков Алексей

Полянский Илья

Отчёт по домашнему заданию №3

**Введение**

Данная работа состоит из двух заданий. Первое задание(А) заключается в ознакомлении с созданием пакетов в ROS и ознакомлением Publisher and Subscriber. Второе задание(Б) заключалась в создании 2 узлов. Один узел нужен для симуляции робота с дифференциальным приводом, а другой нужен от рисовки траектории по показаниям энкодеров.

**Задание А**

В данной части мы познакомились с 3 туториалами. Каждый из них познакомил нас с Ros.

Первый туториал состоял из создания пакета в Ros используя catkin. После ввода команды *$ catkin\_create\_pkg beginner\_tutorials std\_msgs rospy roscpp*

у нас создается пакет с названием beginner\_tutorials с зависимостями std\_msgs, roscpp, and rospy. Далее нам нужно ввести команду *$ cd ~/catkin\_ws && catkin\_make* для того, чтобы создать пакет в рабочей области catkin.

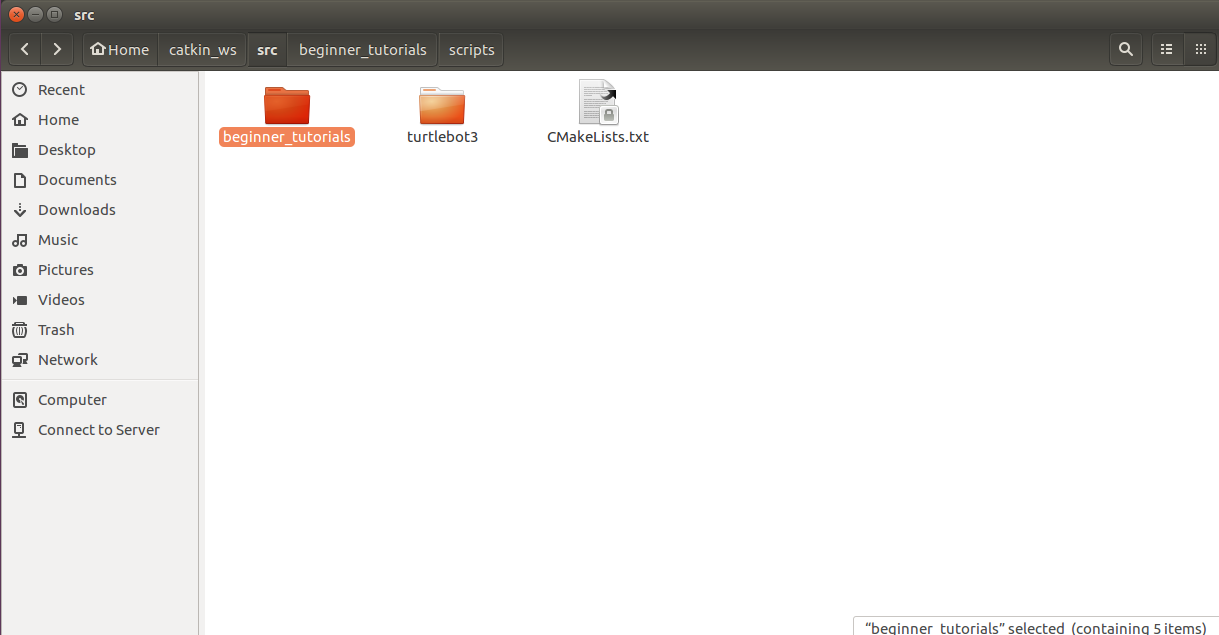


Рисунок 1.Создание пакета beginner\_tutorials

Второй туториал нужен был нам для того чтобы узнать, как написать узел Publisher and Subscriber на питоне. Для этого перейдем в *beginner\_tutorials* с помощью команды *$ roscd beginner\_tutorials* и создадим там папку scripts с помощью команды *$ mkdir scripts.* Далее мы переходим в папку scripts и загружаем с githab скрипт talker.py с помощью команды *$wget https://raw.github.com/ros/ros\_tutorials/kinetic-devel/rospy\_tutorials/001\_talker\_listener/talker.py.* Потом нам нужно сделать его исполняемым с помощью команды *$ chmod +x talker.py*. Этот скрипт будет отправлять сообщение. И нам нужно это сообщение принимать. Для этого нам понадобиться второй скрипт, который мы скачаем той же командой. Нам нужно будет перейти обратно в папку scripts с помощью команды *$ roscd beginner\_tutorials/scripts/* и после этого прописываем команду *$wget https://raw.github.com/ros/ros\_tutorials/kinetic-devel/rospy\_tutorials/001\_talker\_listener/listener.py*. После этого произойдет загрузка скрипта listener.py, который будет принимать сообщение от talker.py. Также нужно не забыть сделать его исполняемым с помощью команды *$ chmod +x listener.py*.

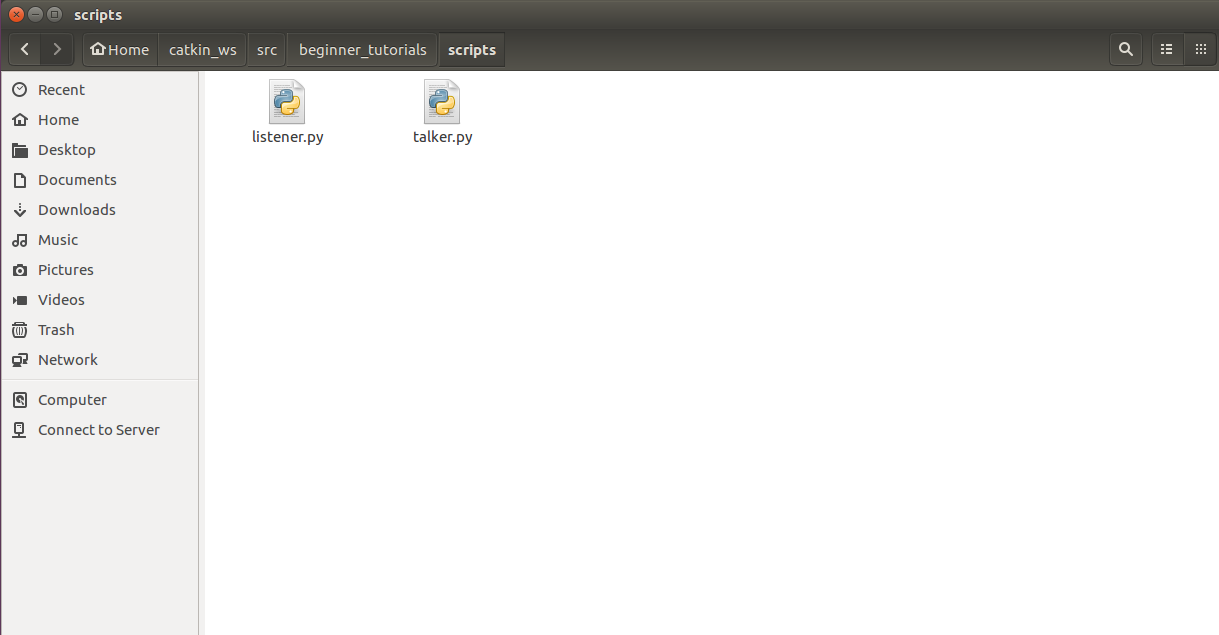


Рисунок 2. Результат второго туториала

Третий туториал показывает, что будет если запустить два скрипта. С помощью команды *$ roscore* мы запускаем ядро Ros. Далее нам нужно написать *$ rosrun beginner\_tutorials talker.py* для запуска скрипта talker.py.

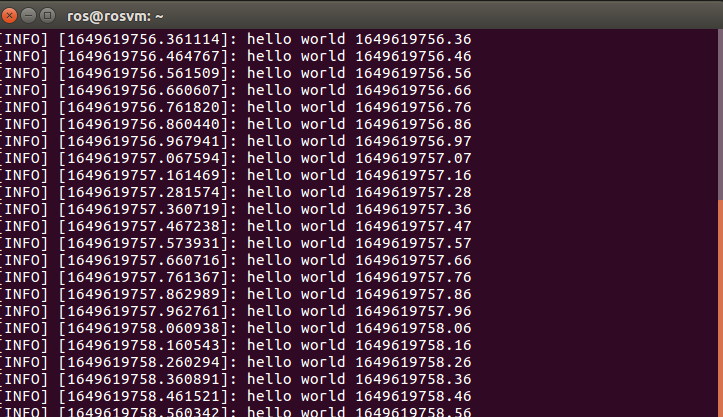


Рисунок 3. Результат запуска talker.py

Эти сообщения выводит talker.py и мы и должны принять с помощью listening.py. Запускаем listening.py параллельно с talker.py с помощью команды *$ rosrun beginner\_tutorials listener.py*.

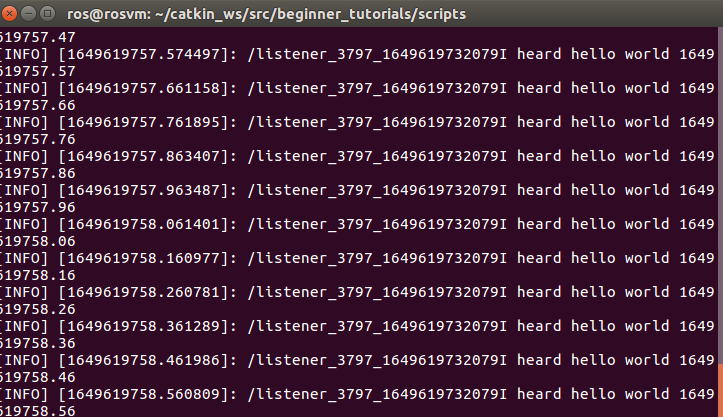


Рисунок 4.Результат запуска скрипта listening.py

Задание Б

Управление симулятором робота осуществляем с помощью пакета turtlebot3\_teleop. Для этого репозиторий с githab клонировали его в *~/catkin\_ws/src*.

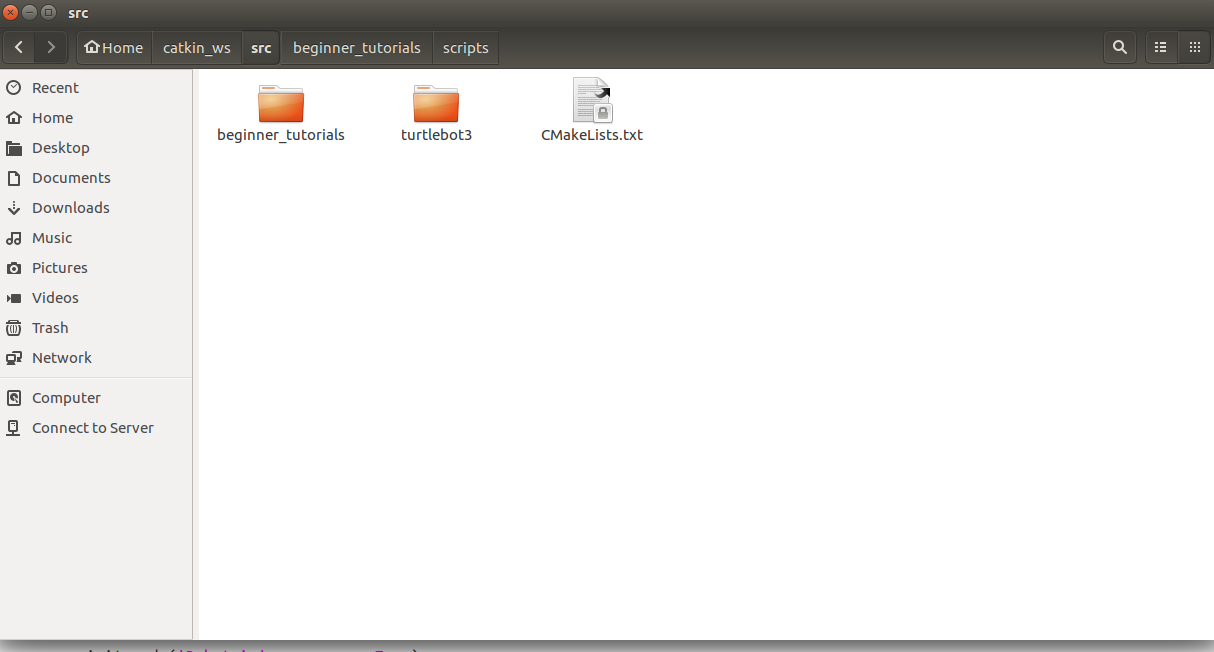


Рисунок 5.Результат клонирования репозитория

Для его запуска в окно терминала пишем *$ export TURTLEBOT3\_MODEL=waffle*; *roslaunch turtlebot3\_teleop turtlebot3\_teleop\_key.launch*.

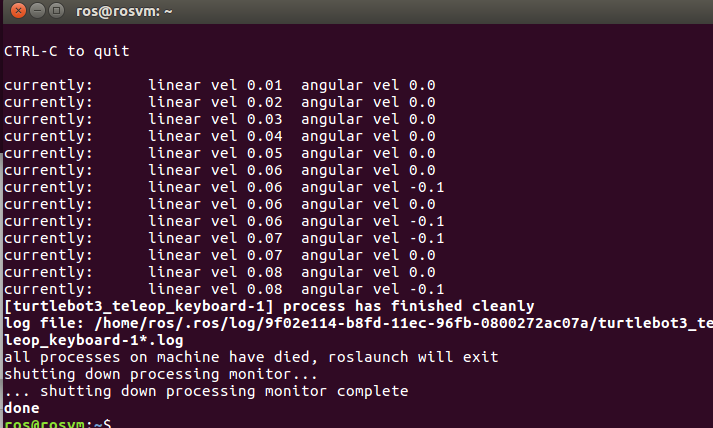


Рисунок 6. Запуск turtelbot3

Из рисунка 6 видно, что, turtlebot3 просто создает эти скорости, чтобы их было удобно вводить и далее передает эти скорости в listeting.py для того чтобы этот скрипт преобразовал их в показания энкодеров. Робот движется по двум осям x и z. X отвечает за линейную скорость вперед и назад, а z отвечает за поворот робота.

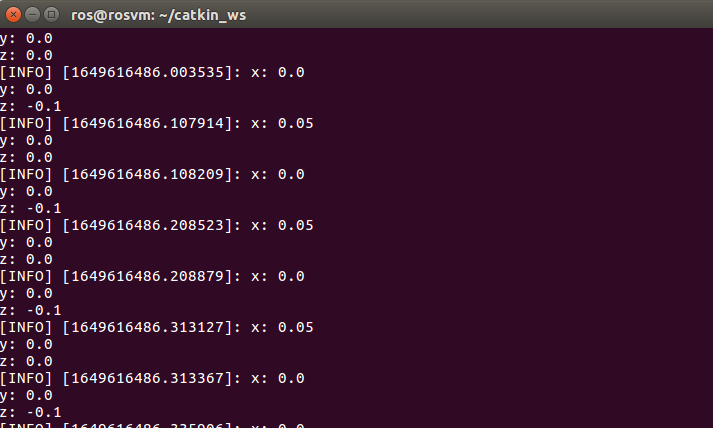


Рисунок 7. Запуск скрипта listening.py

Далее мы хотели передавать показания энкодоров и нынешнее время в Nodule2.

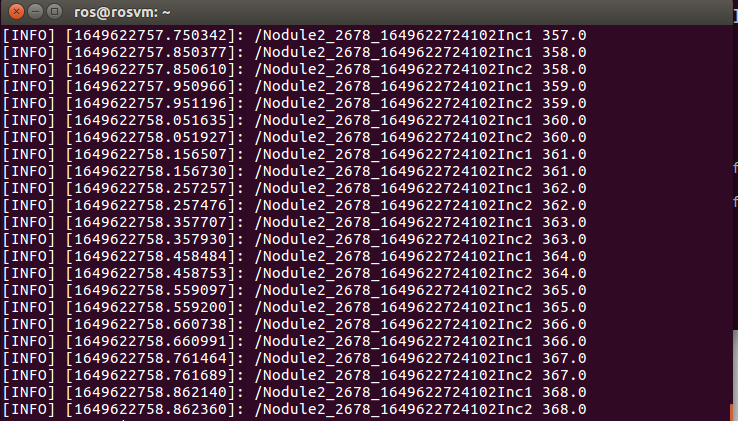


Рисунок 8. Запуск Nodule2

Inc1 и Inc2 это инкриминирование. Оно сделано для того чтобы показать, что связь между listening.py и Nodule2.py есть. К сожалению, это все что мы смогли.

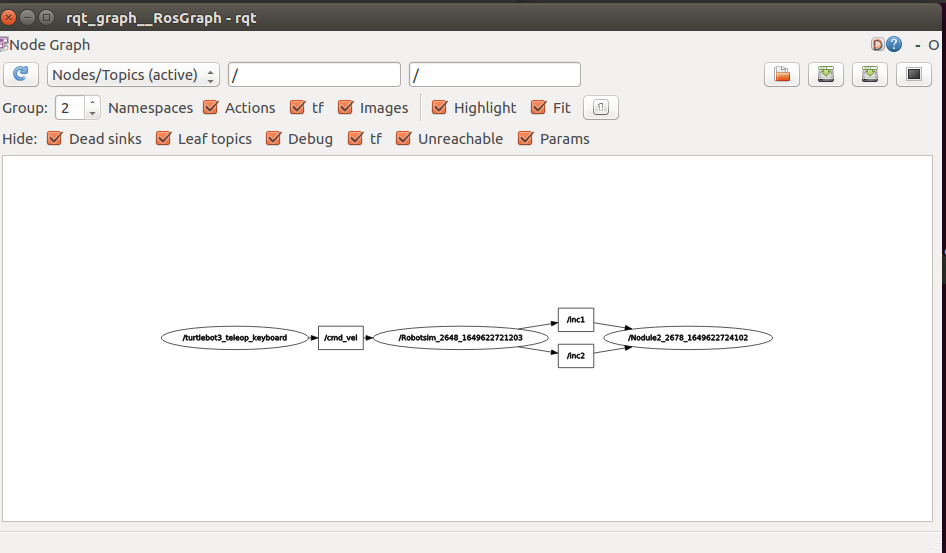


Рисунок 9.Граф сети.