КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ
ВЫСШАЯ ШКОЛА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И
ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ
Кафедра интеллектуальной робототехники

Р.О. ЛАВРЕНОВ, Е.А. МАГИД

ОСНОВЫ РОБОТОТЕХНИЧЕСКОЙ ОПЕРАЦИОННОЙ СИСТЕМЫ (ROS)

учебно-методическое пособие

КАЗАНЬ
2019

УДК 621.865.8(075.8)
ББК 32.816я7
Л13

Рецензент
кандидат технических наук, заведующий лабораторией
нейроморфных вычислений и нейросимуляции
Высшей школы ИТИС КФУ **М.О. Таланов**

Лавренов Р.О.

Л13 **Основы Робототехнической операционной системы (ROS):** учебно-методическое пособие / Р.О. Лавренов, Е.А. Магид. – Казань: Изд-во Казан. ун-та, 2019. – 36 с.

Учебно-методические пособие предназначено для студентов, обучающихся по направлению «Программная инженерия» и профилю подготовки «Интеллектуальная робототехника». Материалы, представленные в пособии, также могут быть полезны преподавателям робототехники. С помощью пособия читатель сможет самостоятельно освоить основные понятия робототехнической операционной системы, изучать сторонние программные библиотеки ROS и связи между ними. Кроме того, в пособии описаны средства исследования ROS-процессов как из командной строки, так и с помощью средств с графическим интерфейсом.

Пособие будет полезно для начинающих изучение Робототехнической операционной системы и может служить шпаргалкой для тех, кто уже знаком с ROS.

УДК 621.865.8(075.8)
ББК 32.816я7

© Лавренов Р.О., Магид Е.А., 2019
© Издательство Казанского университета, 2019

Содержание

Введение

Программное обеспечение является сердцем любого робота. Для обеспечения функциональных возможностей для связи с исполнительными механизмами и датчиками робота чаще всего используются операционные системы. Операционная система на основе Linux может обеспечить большую гибкость взаимодействия с низкоуровневым оборудованием и обеспечить настройку операционной системы в соответствии с приложением робота. Преимуществами Ubuntu в этом контексте являются его отзывчивость, легкость и отличную поддержку сообщества. Ubuntu также имеет релизы долгосрочной поддержки (LTS), которые обеспечивают поддержку пользователей на срок до пяти лет. Эти факторы заставили разработчиков ROS придерживаться Ubuntu, и это единственная операционная система, которая полностью поддерживается ROS. Комбинация Ubuntu-ROS является идеальным выбором для программирования роботов.

Таким образом первое требование к начинающим изучать ROS — наличие базовых умений работы с linux-системами. В частности, знакомство читателя с терминалом (командной строкой) linux, методами установки программ в linux-системах.

Начнем наше изучение Робототехнической Операционной Системы!

1. Программирование роботов

1.1 Что такое программирование роботов?

Как вы знаете, робот — это машина с датчиками, исполнительными механизмами (двигателями или, как их еще называют, актуаторами) и вычислительным блоком, который работает на основе пользовательских команд или может принимать свои собственные решения на основе данных с датчиков. Можно сказать, что мозг робота — это вычислительная единица. Это может быть микроконтроллер или ПК. Принятие решений и действия робота полностью зависят от программы, управляющей мозгом робота. Эта программа может быть встроенным программным обеспечением, работающим на микроконтроллере, или кодом C/C ++ или Python, работающим на ПК или одноплатном компьютере, например Raspberry Pi. Программирование робота — это процесс написания программы, контролирующей работу робота. На рисунке 1.1 демонстрируется общая блок-схема робототехнических устройств.

Основными компонентами любого робота являются исполнительные механизмы и датчики. Приводы перемещают суставы робота, обеспечивая вращательное или линейное движение. Редукторы с сервоприводом, шаговые двигатели и двигатели постоянного тока являются видами исполнительных механизмов. Датчики обеспечивают данные о положении робота и его частей и информацию об окружающей среде. Примеры датчиков робота: энкодеры колес, ультразвуковые датчики, лазерные дальномеры и камеры.

Актуаторы (или двигатели) управляются специальными контроллерами и взаимодействуют с микроконтроллером/ПК. Некоторые приводы напрямую управляются через USB-порт ПК. Датчики также взаимодействуют с микроконтроллером или ПК. Ультразвуковые датчики и инфракрасный датчик соединяются с использованием специальных контроллером.

Высококачественные датчики, такие как камеры и лазерные сканеры, могут напрямую взаимодействовать с ПК. Для питания всех роботизированных компонентов обязательно наличия блока питания или аккумулятора. Часто есть кнопка аварийного останова для остановки/сброса

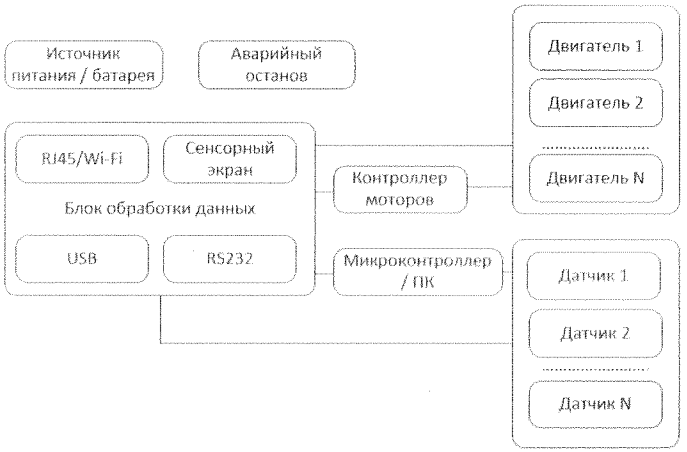

Рис. 1.1: Общая блок-схема роботов

работы робота. Две основные части для программирования внутри робота — это ПК и микроконтроллер/ПЛК (программируемый логический контроллер). ПЛК в основном используются в промышленных роботах.

Короче говоря, мы можем сказать, что программирование робота — это программирование ПК/микроконтроллера/ПЛК внутри робота для выполнения определенного действия с использованием исполнительных механизмов и обратной связи от различных датчиков.

Для программирования роботов с ПК могут использоваться различные языки программирования: C/C++, Python, Java, C# и т.д. Для микроконтроллеров используют Embedded C, язык Wiring (на основе C++), который используется в Arduino, и Mbed (https://os.mbed.com). В приложениях для промышленных роботов используются SCADA или собственные языки программирования, разработанные производителями, такие как ABB, или KUKA.

Программирование робототехнических программ создает элементы искусственного интеллекта роботов для самостоятельного принятия решений, автоматизации повторяющихся задач и роботизированного зрения.

1.2 Почему программирование роботов так различается?

Программирование роботов является подмножеством компьютерного программирования. У большинства роботов есть «мозг», который может

принимать решения. Это может быть микроконтроллер или ПК. Различия между программированием робота и обычным программированием заключаются в устройствах ввода и вывода. Устройства ввода включают в себя датчики роботов, обучающие подвески и сенсорные экраны, а устройства вывода включают ЖК-дисплеи и исполнительные механизмы. Любой из языков программирования может использоваться для роботов, но производительность и время создания прототипов программ делают C++ и Python наиболее часто используемыми.

Ниже приведены некоторые функции, необходимые для программирования робота.

- *Многопоточность:* Как видно на блок-схеме (рисунок 1.1), в роботе могут быть несколько датчиков и исполнительных механизмов. Следовательно, понадобиться многопоточный язык программирования для работы с разными датчиками и сенсорами в разных потоках. Каждый поток может общаться друг с другом для обмена данными.

- *Высокоуровневое объектно-ориентированное программирование:* Как вы уже знаете, объектно-ориентированные языки программирования являются модульными, и код можно легко использовать повторно. Поддержка кода также гораздо легче по сравнению с не объектно-ориентированными языками программирования.

- *Низкоуровневое управление устройствами:* Языки программирования высокого уровня могут также получать доступ к устройствам низкого уровня, таким как пины GPIO (универсальные порты ввода/вывода), последовательные порты, параллельные порты, USB, SPI и I2C. Языки программирования, такие как C/C ++ и Python, могут работать с устройствами низкого уровня, поэтому одноплатные компьютеры, такие как Raspberry Pi и Odroid предпочитают эти языки.

- *Простое прототипирование:* Простота создания прототипа алгоритма робота, безусловно, является аргументом при выборе языка программирования. Python — хороший выбор для быстрого прототипирования алгоритмов роботов.

- *Межпроцессное взаимодействие:* У робота много датчиков и исполнительных механизмов. Мы можем использовать многопоточную архитектуру или написать независимую программу для выполнения каждой задачи; например, одна программа берет изображения с камеры и обнаруживает лицо, другая программа отправляет данные на встроенную плату. Эти две программы могут общаться друг с другом для обмена данными. Многопоточная система более сложна, чем параллельная работа нескольких программ.

- *Производительность:* Если мы работаем с датчиками с высокой пропускной способностью, такими как камеры глубины и лазерные сканеры, вычислительные ресурсы, необходимые для обработки данных, очевидно, высоки. Хороший язык программирования может позволяет контролировать работу с памятью — выделять и загружать необходимые объемы. Язык C++ — хороший выбор для обработки подобных сценариев.

- *Поддержка сообщества:* При выборе любого языка программирования для программирования роботов убедитесь, что для этого языка достаточно поддержки сообщества, включая форумы и блоги.

- *Наличие сторонних библиотек:* Наличие сторонних библиотек может облегчить нашу разработку; например, если мы хотим выполнить обработку изображений, мы можем использовать библиотеки, такие как OpenCV. Если ваш язык программирования имеет поддержку OpenCV, вам будет проще создавать приложения для обработки изображений.

- *Поддержка существующего программного обеспечения для робототехники:* Существуют существующие программные среды для программирования роботов, такие как ROS. Если ваш язык программирования имеет поддержку ROS, проще создать прототип приложения для робота.

2. Робототехническая операционная система (ROS)

2.1 Что такое ROS?

ROS — это свободно распространяемый фреймворк с открытым исходным кодом для робототехники, который используется как в коммерческих, так и в исследовательских приложениях. Платформа ROS предоставляет следующие возможности программирования роботов.

- *Интерфейс передачи сообщений между процессами.* ROS обеспечивает интерфейс передачи сообщений для связи между двумя программами или процессами. Как уже упоминалось, это одна из функций, необходимых для программирования робота.

- *Элементы операционной системы.* Как следует из названия, ROS не является реальной операционной системой. Это мета операционная система, которая обеспечивает некоторые функциональные возможности операционной системы. Эти функции включают многопоточность, низкоуровневое управление устройством, управление пакетами и аппаратная абстракция. Управление пакетами помогает пользователям организовывать программное обеспечение в единицах, называемых пакетами. Каждый пакет имеет исходный код, файлы конфигурации или файлы данных для конкретной задачи. Эти пакеты могут быть распространены и установлены на других компьютерах.

- *Поддержка языков программирования высокого уровня.* Преимущество ROS состоит в том, что он поддерживает популярные языки программирования, используемые в программировании роботов, включая C, Python и Lisp. Существует экспериментальная поддержка таких языков, как C#, Java, Node.js и т. д. Полный список находится по адресу http://wiki.ros.org/. ROS предоставляет клиентские библиотеки для этих языков. Это означает, что программист может получить функциональные возможности ROS на

упомянутых языках. Например, если пользователь хочет реализовать приложение Android, использующее функциональность ROS, можно использовать клиентскую библиотеку rosjava. ROS также предоставляет инструменты сборки пакетов.

- *Наличие сторонних библиотек.* Платформа ROS интегрирована с большинством популярных сторонних библиотек; например, библиотека OpenCV (https://opencv.org) интегрирована для машинного зрения, а PCL (http://pointclouds.org) интегрирована для реконструкции и манипуляции с 3D окружением робота.

- *Готовые алгоритмы.* В ROS реализованы популярные алгоритмы робототехники, такие как PID (wiki.ros.org/pid); SLAM (одновременная локализация и картографирование) (wiki.ros.org/gmapping); и планировщики пути, такие как A, Dijkstra (wiki.ros.org/global_planner) и локализация AMCL (адаптивная локализация Монте-Карло) (wiki.ros.org/amcl). Готовые алгоритмы сокращают время разработки робототехнических программ.

- *Поддержка сообщества.* Разработчики ROS есть по всему миру. Они активно разрабатывают и поддерживают пакеты ROS. Большая поддержка сообщества включает разработчиков, задающих вопросы, связанные с ROS. ROS Answers — это платформа для вопросов, связанных с ROS (answers.ros.org/questions/). ROS Discourse — это онлайн-форум, на котором пользователи ROS обсуждают различные темы и публикуют новости, связанные с ROS (discourse.ros.org).

- *Утилиты и среды моделирования.* ROS состоит из множества инструментов, работающих как из командной строки, так и с графическим интерфейсом, для отладки, визуализации и моделирования робототехнических приложений. Например, инструмент Rviz (wiki.ros.org/rviz) используется для визуализации данных с камер, лазерных сканеров, инерциальных датчиков, камер глубины и т. д. Для моделирования окружающей среды для робота используют симуляторы, такие как Gazebo (gazebosim.org).

До проекта ROS велась активная разработка в области робототехники, но не было единой платформы и сообщества для разработки приложений для робототехники. Каждый разработчик создавал программное обеспечение для своего собственного робота, которое в большинстве случаев не могло быть использовано для любого другого робота. После возникновения ROS все изменилось. Сейчас существует общая платформа для разработки робототехнических приложений. Это бесплатный и открытый исходный код для коммерческих и исследовательских целей. ROS изменил и создал новые принципы программирования робототехники.

2.2 История возникновения и развития ROS

Ниже приведены исторические вехи проекта ROS.

- Проект ROS был запущен в Стэнфордском университете в 2007 году под руководством Моргана Куигли. Сначала это был набор программного обеспечения, разработанный для роботов в Стэнфорде.

- Позже, в 2007 году, стартап по исследованию робототехники под названием Willow Garage взял на себя проект и придумал название ROS, что означает «Робототехническая операционная система».

- В 2010 году была выпущена ROS 1.0. Многие из его функций все еще используются.

- В 2012 году ROS переходит к OSRF (Open Source Robotics Foundation).

- В 2014 году был выпущен ROS Indigo Igloo (восьмой релиз), это был первый выпуск с долгосрочной поддержкой (LTS).

- В 2016 году была выпущена ROS Kinetic Kame. Это вторая LTS версия ROS.

- В мае 2018 года была выпущена двенадцатая версия ROS, Melodic Morenia.

Версии ROS и более подробную историю можно найти на сайте www.ros.org/history/. Каждая версия ROS называется дистрибутивом ROS. Если вы ищете новейшие функции ROS, вы можете выбрать новые дистрибутивы, а если вы ищете стабильные пакеты, вы можете выбрать LTS. В данном пособии примеры используют версию Kinetic Kame. В настоящее время ROS разрабатывается и поддерживается Open Robotics, ранее известной как Open Source Robotics Foundation.

ROS продолжает развиваться. Проект под названием ROS 2.0 разрабатывает гораздо лучшую версию существующей ROS с точки зрения безопасности и работы в реальном времени // (github.com/ros2/ros2/wiki). ROS 2.0 может стать хорошим выбором для робототехники в будущем.

2.3 Установка ROS

Это важный шаг в развитии ROS. Установить ROS на ПК несложно. Перед установкой вы должны знать о различных платформах, которые поддерживают ROS. На рисунке 2.1 показаны различные операционные системы, в которых вы можете установить ROS. Как уже говорилось, ROS не является операционной системой, но для работы ей нужна операционная система.

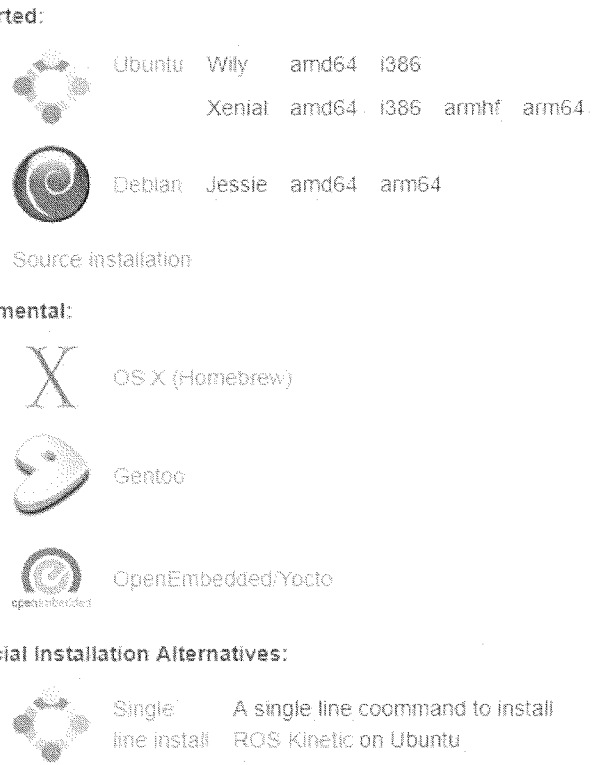

Рис. 2.1: Операционные системы, которые поддерживают ROS.

Ubuntu Linux является наиболее предпочтительной ОС для установки ROS. Как вы можете видеть на рис. 2.1, ROS поддерживает 32-битную, 64-битную Ubuntu, 32-битную ARM и 64-битную ARM. Это означает, что ROS может работать на ПК и на одноплатных компьютерах, таких как Raspberry, Odroid и NVIDIA TX1/TX2. Debian Linux также поддерживает ROS. В OS-X и других операционных системах ROS все еще находится в экспериментальной фазе, что означает, что функциональные возможности ROS еще не доступны.

Давайте перейдем к установке. Если вы используете ПК или плату ARM, на которой работает Ubuntu armhf или arm64, вы можете выполнить процедуры со страницы: http://wiki.ros.org/ROS/Installation. Когда вы заходите в эту вики, она спрашивает, какую версию ROS вам нужно установить. В зависимости от установленной версии Ubuntu, доступны

для установки разные версии ROS.

Мы можем установить ROS двумя способами: с помощью готовых бинарных файлов или путем компиляции исходного кода. В этом пособии мы устанавливаем ROS Kinetic на Ubuntu 16 LTS.

Ниже описаны этапы установки.

1. Настройка файла *sources.list*. Это важный шаг в установке ROS. Он добавляет в ОС информацию о хранилище ROS, где хранятся двоичные файлы. Для этого выполните следующую команду в терминале.

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

   Эта команда создает новый файл /etc/apt/sources.list.d/ros-latest.list и добавляет к нему следующую строку.
   "deb http://packages.ros.org/ros/ubuntu xenial main". Обратите внимание, что если вы выполните

   lsb_release -sc

   в терминале, вы получите вывод 'xenial'.

2. Добавление ключей. В Ubuntu, если мы хотим загрузить двоичный файл или пакет, мы должны добавить безопасный ключ в нашу систему, чтобы аутентифицировать процесс загрузки. Пакет, который аутентифицируется с использованием этих ключей, является доверенным. Ниже приведена команда для добавления ключей.

   sudo apt-key adv –keyserver hkp://ha.pool.sks-keyservers.net:80 –recv-key 421C365BD9FF1F717815A3895523BAEEB01FA116

3. Обновление списка досупных пакетов Ubuntu. Делается это с помощью команды *update* программы *apt*, которая контролирует пакеты в Ubuntu.

   sudo apt update

4. Устанавливаем пакеты ROS Kinetic, используя следующую команду.

   sudo apt-get install ros-kinetic-desktop-full

   Эта команда устанавливает все необходимые пакеты в ROS, включая необходимые для работы библиотеки, симуляторы и основные алгоритмы для роботов. Для загрузки и установки всех этих пакетов требуется время.

5. После установки всех пакетов нам нужно установить инструмент под названием *rosdep*, который полезен для установки зависимых пакетов пакета ROS. Например, какой-нибудь пакет ROS может иметь несколько зависимых пакетов для правильной работы. rosdep проверяет, доступны ли зависимые пакеты, и если нет, автоматически устанавливает их. Следующие две команды устанавливают rosdep и обновляют его конфигурацию.

```
sudo rosdep init
rosdep update
```

6. Настройка окружения ROS. Как обсуждалось ранее, ROS поставляется с необходимыми утилитами и библиотеками. Чтобы получить доступ к этим инструментам и пакетам командной строки, нам нужно настроить среду ROS. Следующая команда добавляет строку в файл *.bashrc* в вашей домашней папке, которая подключает среду ROS в каждом новом окне терминала.

```
echo "source /opt/ros/kinetic/setup.bash»>  /.bashrc
```

Затем введите следующую команду, чтобы добавить среду ROS в текущее окно терминала.

```
source  /.bashrc
```

Установка завершена.

7. Установка дополнительных пакетов, полезных для сборки остальных пакетов.

```
sudo apt-get install python-rosinstall python-rosinstall-generator
python-wstool build-essential
```

Поздравляем, вы закончили с установкой. Вы можете проверить правильность установки, используя следующую команду.

```
rosversion -d
```

Если в качестве вывода будет слово *'kinetic'* то и установка и настройка прошли успешно.

3. Архитектура ROS

Робот может иметь множество датчиков и исполнительных механизмов. Как мы можем управлять несколькими приводами и обрабатывать данные с большого количества датчиков? Лучший выход, написать небольшие независимые программы для обработки данных датчиков и управления приводами, и обмен данными между этими программами. Это именно та ситуация, когда нужно использовать ROS.

По сути, ROS является основой для взаимодействия между двумя или более программами или процессами. Например, если программа A хочет отправить данные в программу B, а B хочет отправить данные в программу A, мы можем легко реализовать это с помощью ROS. Давайте посмотрим, как происходит связь между двумя программами в ROS. Рисунок 3.1 иллюстрирует базовую блок-схему ROS.

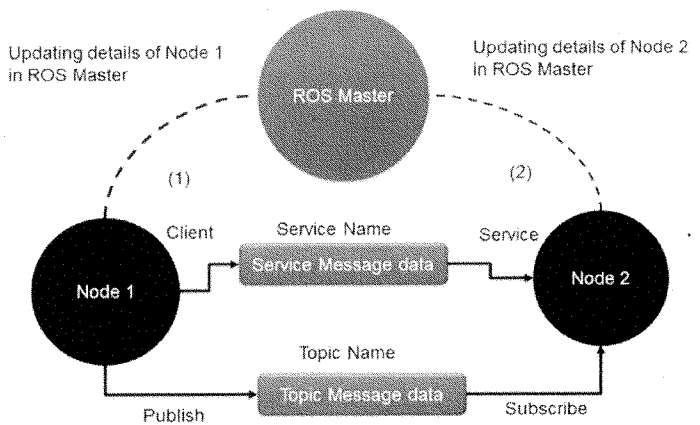

Рис. 3.1: Базовая блок-схема ROS.

На рисунке 3.1 показаны две программы, помеченные как Нода 1 и Нода 2. Нода (node — с англ. узел) — название программы в ROS. Когда запускается любая из программ, нода связывается с программой ROS,

называемой ROS-Мастер (ROS Master). Нода отправляет всю свою информацию ROS-Мастеру, включая типы данных, которые она отправляет или получает. Ноды, которые отправляют данные, называются публикующими (publish — с англ. издавать, выпускать) , а ноды, которые принимают данные, называются подписчиками (subscribe — с англ. подписаться). ROS-Мастер имеет всю информацию о запущенных нодах, что они публикуют и на что подписаны. Если нода 1 отправляет конкретные данные, называемые «A», иноде 2 требуются те же данные, то ROS-Мастер отправляет информацию нодам, чтобы они могли обмениваться данными друг с другом.

Ноды ROS могут отправлять друг другу разные типы и структуры данных, которые включают в себя примитивные типы данных, такие как целое число, число с плавающей запятой, строки и т.д. Различные типы отправляемых данных называются сообщениями (messages — с англ. сообщения) ROS. С помощью сообщений ROS мы можем отправлять данные одного типа или данные одной конкретной структуры. Эти сообщения отправляются через поток сообщений, называемый топиком ROS (topic — с англ. тема). У каждого топика есть имя. Например, в топике с именем «cmd_vel» отправляются команды скорости на автономное устройство.

На рисунке 3.1 нода 1 публикует в топик, а нода 2 подписана на него. За взаимообмен данными между нодами отвечает ROS-Мастер. Далее давайте рассмотрим некоторые важные понятия и термины, которые используются при работе с ROS. Их можно классифицировать как три категории: файловая система ROS, концепции ROS и ROS-сообщество.

3.1 Файловая система ROS

Файловая система ROS включает пакеты, метапакеты, файлы описания пакетов, репозитории, типы сообщений и типы сервисов.

Пакеты ROS — это отдельные единицы или атомные единицы программного обеспечения ROS. Весь исходный код, файлы данных, файлы сборки, зависимости и другие файлы организованы в пакеты. Метапакет ROS группирует набор похожих пакетов для конкретного приложения. В метапакете ROS нет исходных файлов или файлов данных. Он имеет зависимости от аналогичных пакетов. Файл описания пакета — это файл XML, размещенный внутри пакета ROS. Он содержит всю основную информацию о пакете ROS, включая имя пакета, описание, автора, зависимости и так далее. Типичный *package.xml* показан ниже.

```
<?xml version="1.0"?>
<package>
<name>test_pkg</name>
<version>0.0.1</version>
```



```
<description>The test package</description>
<maintainer email="lirs_itis_kfu@gmail.com">robot</maintainer>
<license>BSD</license>
<buildtool_depend>catkin</buildtool_depend>
.................. ............... .
<run_depend>catkin</run_depend>
....... . ............ .
</package>
```

ROS репозиторий — это набор пакетов ROS, которые используют общую систему контроля версий. Внутри каждого пакета обычно находится несколько директорий. В директории *src* хранятся файлы исходного кода, в *include* — заголовочные файлы. В папке *msg* пользователи могут написать собственный тип сообщений, который будет передаваться в топиках. А в директории *srv* — собственный сервис. Кроме того, пользователь может создать папку *launch* в которой будут лежать файлы запуска нод (.launch файлы) с определенными, прописанными параметрами. В файле *CMakeLists.txt*, обязательно имеющемся в каждом пакете, находятся настройки сборки пакета, а также указываются все необходимые для сборки зависимости. На рисунке 3.2 демонстрируется пример содержимого пакета ROS.

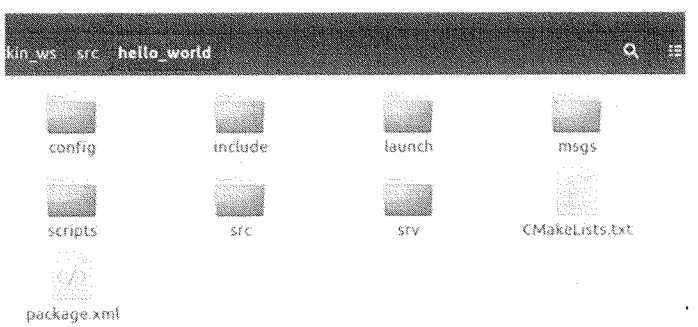

Рис. 3.2: Структура пакета ROS.

3.2 Основные понятия ROS

Термины, которые используются в Робототехнической Операционной Системе.

- *Nodes – ноды.* Программа ROS, выполняющая какие-либо действия (обработка данных с датчиков, отправка данных в другие ноды, и т.д.), в которой используется API ROS, собирается из одного или

нескольких файлов с исходным кодом. Является базовым элементом ROS. Могут запускаться независимо друг от друга.

- *ROS Master – ROS-Мастер.* Первоначально запускаемая программа, которая соединяет ноды ROS, для межпроцессорного взаимодействия между ними.

- *Parameter server – сервер параметров.* Программа, которая запускается вместе с ROS-Мастером. Пользователь может хранить различные параметры или значения на этом сервере, и все ноды могут получить к нему доступ. Пользователь также может установить конфиденциальность параметров. Если это публичный (public) параметр, все ноды имеют к нему доступ; если это частный (private), то только конкретная нода будет иметь доступ к параметру.

- *Topic – топик или тема.* Именованный поток данных, в которых передаются данные определенного типа (или структуры). Ноды ROS могут публиковать сообщения в топики или подписываться на них для получения данных. Ноды могут публиковать или подписываться на любое количество топиков.

- *Message – сообщение.* Сообщения передаются через топики. Они могут быть примитивными типами данных или сложными структурами данных. Пользователи могут формировать собственные структуры для отправки их в качестве сообщений.

- *Service – сервис.* Реализация в ROS механизма "запрос/ответ" (request/responce). Нода, которая реализует работу сервиса, называется сервером, а нода, которая вызывает сервис, называется клиентом. Выполнение сервисной функции осуществляется только по запросу от клиентской ноды.

- *ROS bags – багс или мешки.* Полезный метод для сохранения и воспроизведения топиков ROS. Бывает необходимо для сохранения определенного потока данных через топики для последующей обработки.

3.3 Сообщество ROS

Ниже приведены ресурсы, используемые для обмена программным обеспечением и утилитами ROS.

- В ROS wiki есть учебники по настройке и программированию ROS.

- Ответы ROS (https://answers.ros.org/questions/) содержат вопросы и решения по проблемам ROS, аналогично ресурсу Stack Overflow.

- Дискурс ROS (https://discourse.ros.org) — это форум, на котором разработчики могут делиться новостями и задавать вопросы, связанные с ROS.

Если вы хотите узнать больше о новинках и передовых разработках ROS, посетите сайт wiki.ros.org/ROS/Concepts.

4. Базовое использование ROS

4.1 Инструменты командной строки

В этом разделе рассматриваются инструменты командной строки ROS. Существуют различные инструменты ROS, которые мы можем использовать для изучения различных аспектов ROS. Мы можем реализовать практически все возможности ROS с помощью этих инструментов. Инструменты командной строки выполняются в терминале Linux.

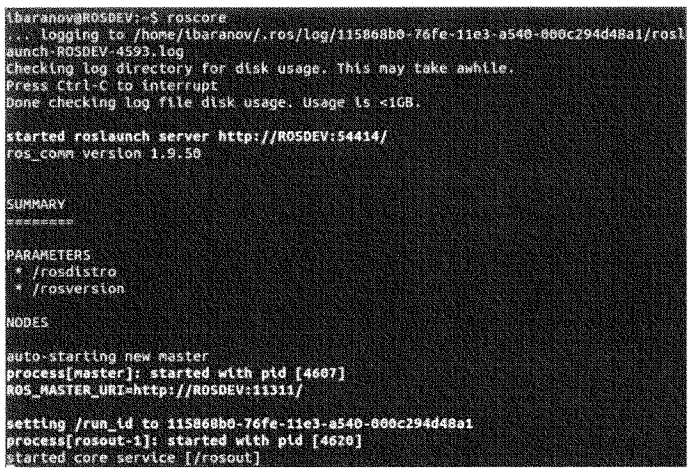

Рис. 4.1: Окно терминала после вызова *roscore*.

1. Команда ***roscore*** — очень важный инструмент в ROS. Когда мы запускаем эту команду в терминале, она запускает ROS-Мастер, сервер параметров и ноду ведения записей (логирования — logging). Мы можем запустить любую другую ноду ROS после выполнения этой команды. Если вы запустите *roscore* в окне терминала, вы

можете получить сообщения, подобные тем, которые показаны на рисунке 4.1. В терминале вы можете видеть сообщения о запуске ROS-Мастера и сервера параметров. Также можно увидеть основной адрес ROS для удаленного подключения.

2. С помощью команды ***rosnode*** можно исследовать все свойства и настройки ноды ROS. Обратите внимание, что при выполнении этой и последующих команд, в отдельном окне должен быть запущен ROS-Мастер (*roscore*).

   rosnode list – вывод списка нод ROS, запущенных на текущий момент.

   rosnode info /NodeName – получение информации о ноде: на какой топик подписана нода и в какой публикует, какие у нее есть сервисы и с какими нодами она связана. Ниже, на рисунке 4.2 приведен пример получения информации о ноде.

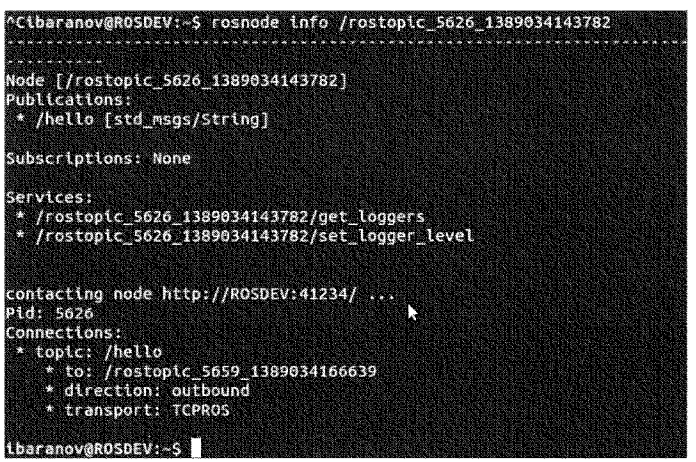

Рис. 4.2: Окно терминала после вызова *rosnode info*.

   rosnode ping /NodeName – проверить соединение с нодой *NodeName*.

   rosnode kill /NodeName – закрытие ноды *NodeName* и очистка всей памяти выделенной на неё.

3. Команда ***rostopic*** предоставляет информацию о топиках, которые публикуются или на которые подписаны ноды в текущий момент. Эта команда очень полезна для анализа существующих топиков, просмотра данных в них, и публикации вручную данных в топики.

   rostopic list – вывод списка всех существующих в системе топиков.

rostopic echo /TopicName – вывод в консоль данных с топика *TopicName*.

rostopic type /TopicName – вывод в консоль информацию о типе данных, передаваемого по топику *TopicName*.

rostopic find /TypeName – поиск топиков в которые публикуются сообщения типа *TypeName*.

rostopic pub TopicName msg_type Data – публикация данных (*Data*), имеющих тип *msg_type* в топик *TopicName*.

4. Инструменты ***rosmsg*** и ***rossrv*** предоставляют пользователю информацию о сообщениях и сервисах, соответственно. Набор команд у них одинаковый и приведен далее на примере *rosmsg*:

   rosmsg show MsgName – вывод информации о типе с названием *MsgName*, если это структура, то выведется содержимое.

   rosmsg package PckgName – вывод информации о всех используемых типах в пакете с названием *PckgName*.

5. Кроме того, для исследования сервисов есть специальный инструмент ***rosservice***. Примеры его использования на рисунке 4.3.

   rosservice list – информация о всех, доступных для вызова сервисах.

   rosservice node /SrvName – вывод названия ноды, которая осесчивает работу сервиса с названием *SrvName*.

   rosservice call /SrvName ... – вызов сервиса с аргументами.

   rosservice find /TypeName ... – поиск сервиса выдающего в качестве ответа сообщение типа *TypeName*.

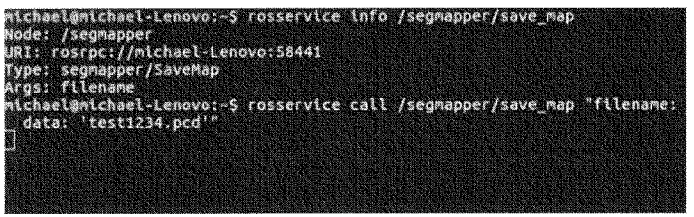

Рис. 4.3: Примеры использования *rosservice*.

6. Инструмент ***rosparam*** содержит команды для получения и установки параметров ROS в сервере параметров с использованием файлов в кодировке YAML (YAML-файлов).

   rosparam list /namespace – получить список параметров из пространства имен *namespace*.

rosparam get /PrmName – получить значение параметра *PrmName*.

rosparam set /PrmName ... – задание значения параметру *PrmName*.

rosparam load – получить значение параметра из файла.

rosparam dump – записать значение параметра в файл.

rosparam delete – удалить параметр из файла.

7. Инструмент ***rosrun*** позволяет запускать ноду из заданного пакета. После написания команды *rosrun* через пробел в окне терминала пишется название пакета, а еще через пробел — название ноды из этого пакета. Пример на рисунке 4.4.

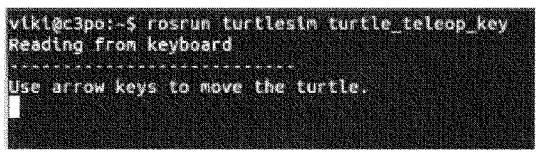

Рис. 4.4: Пример использования *rosrun*.

8. Инструмент ***roslaunch*** позволяет не запускать для каждой ноды отдельное окно терминала. Данный инструмент позволяет воспользоваться .launch файлами, которые имеют структуру XML. В данных файлах можно записать вызов нескольких нод и параметров, с которыми они будут вызваны. Ноды будут вызваны в порядке упоминания в .launch файле. Кроме того, при использовании *roslaunch*, перед запуском первой ноды, будет автоматически запущен *roscore*. Таким образом используя *roslaunch* можно обойтись одним окном терминала, и запускать одновременно несколько нод, отвечающих за разные процессы робототехнической системы и симуляции.

   roslaunch package filename.launch – пример запуска файла *filename.launch* из пакета *package*.

4.2 Программа "Hello world"в ROS

Разберем базовый пример, который прилагается к установленному ROS из пакета *roscpp_tutorials*.

Есть две ноды: *talker* и *listener*. Нода *talker* публикует строковое сообщение. Нода *listener* подписывается на него. В рассматриваемом примере *talker* публикует сообщение Hello World, а *listener* получает его и выводит на экран. Рисунок 4.5 демонстрирует схему взаимодействия двух нод. Как обсуждалось ранее, обе ноды вначале соединяются с ROS-мастером,

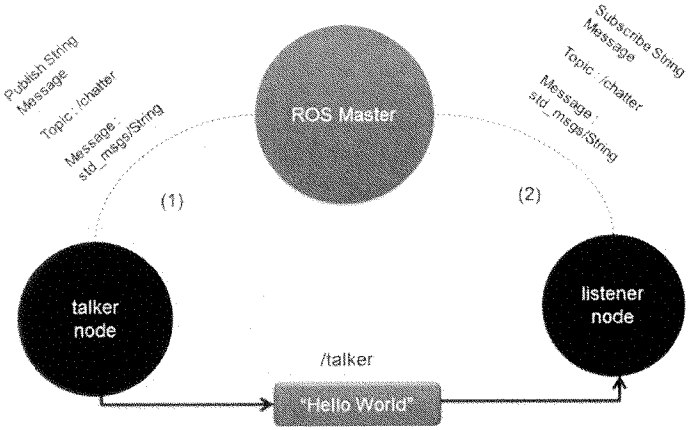

Рис. 4.5: Связь между нодами *talker* и *listener*.

а уже он соединяет их между собой создавая поток данных — топик */talker*.

Вначале запустим наш пример с помощью запуска нод по отдельности. Каждую команду следует запускать в новом окне терминала. Вначале нужно запустить ROS-Мастер с помощью команды:

roscore

Запустим ноду *talker* с помощью команды *rosrun*.

rosrun roscpp_tutorials talker

После запуска ноды вы увидите что она с частотой 1 секунда выводит на экран на экран сообщение "hello world"и номер сообщения. В новых окнах терминала вы можете поэкспериментировать с различными инструментами ROS. Например, если вы выполните команду

rostopic list,

то в выводе увидите топик */chatter*. Именно в этот топик отправляет сообщения нода *talker*. Теперь запустите ноду, слушающую топик, используя следующую команду.

rosrun roscpp_tutorials listener

Нода *listener* начнет выводить сообщения, которые получает от ноды *talker* (рисунок 4.6).

Если вы хотите запустить две ноды в одном окне терминала, используйте команду roslaunch. Так как в пакете *roscpp_tutorials* уже есть готовый .launch файл, запускающий обе ноды, то можно закрыть все открытые окна терминала и в новом окне терминала запустить:

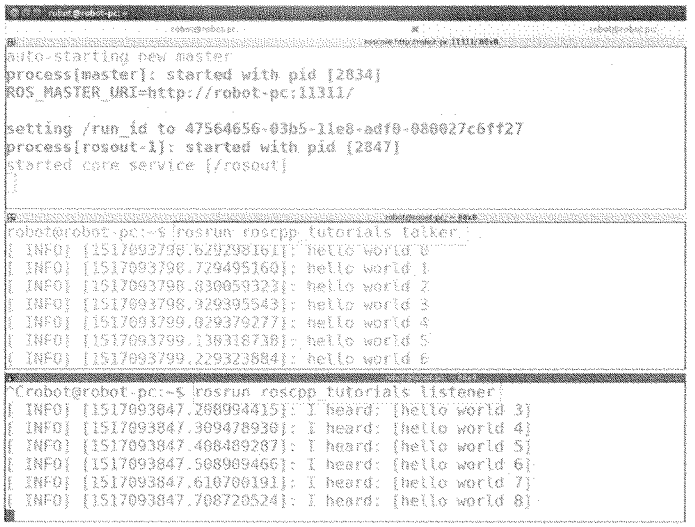

Рис. 4.6: Окна терминала при запуске нод *talker* и *listener*.

roslaunch roscpp_tutorials talker_listener.launch

4.3 Программа "Turtlesim"в ROS

В этом разделе приводится пример интересного приложения для демонстрации концепции ROS. Приложение называется turtlesim, это 2D симулятор черепашки в нем. Вы можете перемещать черепаху, получать текущее положение черепахи, посылать на неё угловые и линейные скорости. При этом используются топики, сервисы и параметры ROS. Работая с turtlesim, вы получите хорошее представление о том, как управлять роботом с помощью ROS. Черепашка используется только для обучения, но, фактически, моделирует любого всенаправленного робота, типа роботов-пылесоса.

Пакет turtlesim предустановлен в ROS. Для запуска, используйте следующие команды:

roscore
rosrun turtlesim turtlesim_node

Вы должны увидеть экран с расположенной в центре черепашкой (рисунок 4.7).

Далее, поэкспериментируйте с командами ROS, выводя различную информацию. Например можно вывести список всех топиков или сервисов (Рисунок 4.8.

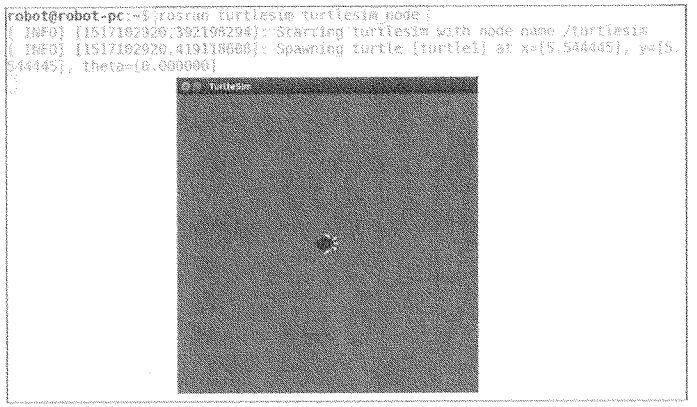

Рис. 4.7: Нода Turtlesim.

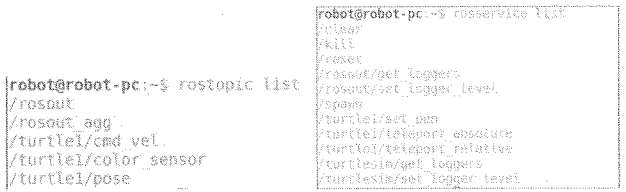

Рис. 4.8: Список запущенных топиков и сервисов Turtlesim.

Далее, начнем подвигаем черепаху. Для этого запустим ноду ROS с помощью следующей команды. Эта команда должна запускаться в другом окне терминала.

rosrun turtlesim turtle_teleop_key

После этого, вы сможете управлять роботом с помощью клавиш со стрелками на клавиатуре.

Таким образом, мы запустили новую ноду которая считывает нажатия клавиш, в зависимости от нажатых клавиш формирует сообщения типа *geometry_msgs/Twist*, хранящее шесть значений типа double. Из них три отвечают за линейные скорости по осям x, y, z, и три за скорости поворота вокруг этих осей. Далее, сформированные сообщения публикуются в топик */turtle1/cmd_vel*. Нода *turtlesim_node* подписана на этот топик, и после получения сообщений обрабатывает их и перемещает с заданной скоростью черепашку-робота, рисуя траекторию пройденного пути. Схема взаимодействия между нодами представлена на рисунке 4.9.

Вы можете поэкспериментировать, отправляя значения скорости вруч-

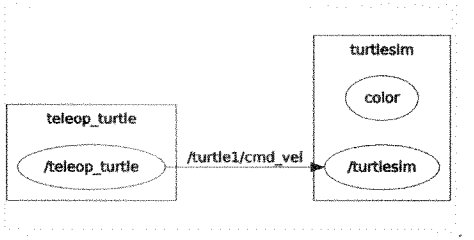

Рис. 4.9: Список запущенных топиков и сервисов Turtlesim.

ную с нового окна терминала. Для этого нужно сформировать сообщение типа *geometry_msgs/Twist* и отправить его в топик */turtle1/cmd_vel* с помощью упомянутой выше функции *rostopic pub*:

rostopic pub -1 /turtle1/cmd_vel geometry_msgs/Twist – '[3.0, 0.0, 0.0]' '[0.0, 0.0, 2]'

Если мы хотим очистить поле черепашки от нарисованных линий, мы можем вызвать службу под названием /reset.

rosservice call /reset

5. Графические средства ROS

5.1 Инструменты фреймворка rqt

rqt — это программный фреймворк внутри ROS, который реализует различные аспекты ROS в удобном для пользователей виде, с использованием графического интерфейса. Любые готовые плагины с графическим интерфейсом можно запустить (при работающем ROS-Мастере) командой:

rosrun rqt_gui rqt_gui

После этого в появившемся интерфейсе можно выбрать любой доступный в системе плагин. Также, пользователи могут создавать свои собственные плагины для rqt с помощью Python или C++. На 2018 существовало более 20 плагинов. rqt заменяет прежние графические инструменты ROS rxtools, которые, начиная с версии ROS Groovy считаются устаревшими.

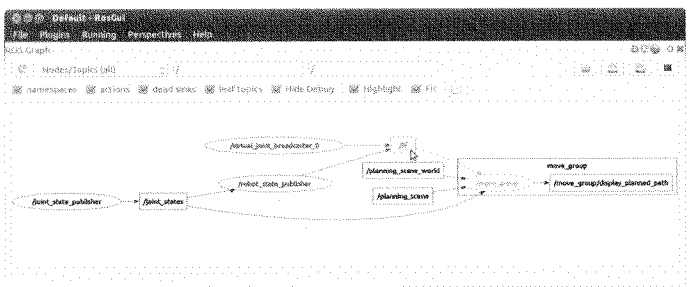

Рис. 5.1: Запущенный плагин rqt_graph, отображающий информацию о нодах и топиках.

Далее, перечислим несколько наиболее часто используемых плагинов, которые нужны для визуализации процессов ROS.

1. ***rqt_graph***. Отображает ноды и взаимодействие между ними в виде графа. Пример на рисунке 5.1. Является интерактивным графом. То есть наводя курсором на ноды или на связывающие их топики можно получить различную дополнительную информацию. Запускается командой:

rosrun rqt_graph rqt_graph

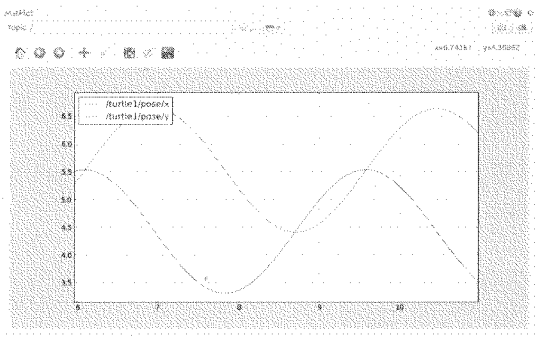

Рис. 5.2: Запущенный плагин rqt_plot, отображающий изменение координат робота-черепашки, выполняющей круговые вращения.

2. ***rqt_plot*** предоставляет плагин ROS, визуализирующий числовые значения в 2D-графике, используя различные шаблоны рисования графиков. Например мы можем отобразить координаты робота-черепашки в виде графика (рисунок 5.2), набрав команду:

rqt_plot /turtle1/pose/x /turtle1/pose/y

3. ***rqt_image_view*** позволяет отображать изображение или видео с нескольких топиков-источников. Например, в первом окне можно отобразить цветное видео с камеры (например, с топика */image_RGB*, а во втором — черно-белое, после его обработки в ноде (например, с топика */image_Grey*.

4. ***rqt_reconfigure*** предоставляет способ просмотра и редактирования параметров нод. Когда вы используете, к примеру, ноды планирования пути или локализации для мобильных роботов, в них имеется большое количество параметров, влияющих на результаты алгоритмов. В зависимости от поставленной задачи, любой из подобных алгоритмов будет нуждаться в дополнительной настройке, что и можно сделать, регулируя параметры с помощью *rqt_reconfigure*. Пример окна представлен на рисунке 5.3.

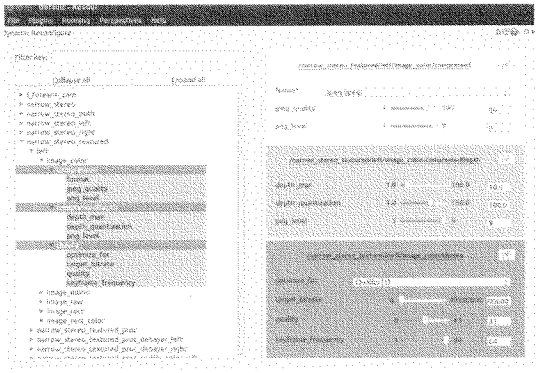

Рис. 5.3: Запущенный плагин rqt_reconfigure, в котором можно настраивать параметры нод.

Существует еще много готовых плагинов, упрощающих разработку собственных нод ROS и исследования существующих. Плагин ***rqt_launch*** упрощает работу с .launch файлами, анализируя ноды в нем и позволяя запускать и останавливать их работу по отдельности. Плагин ***rqt_topic*** отображает информацию о топиках, кто на них подписан и кто в них публикует. ***rqt_msg*** может визуально представить структуру передаваемых сообщений и в каких топиках они передаются. Это еще не весь список плагинов, полезных при исследовании и создании сложных, многопроцессорных алгоритмов. Плагины rqt поддерживаются во всех выходящих дистрибутивах ROS и для разработчиков открыт шаблон по созданию собственных плагинов для ROS.

5.2 Симулятор RViz

Наряду с инструментами командной строки, ROS имеет программы с графическим интерфейсом для визуализации данных с датчиков. ***RViz*** — это программа визуализации, которая используется в ROS. В ней используется 3D-среда, которая позволяет пользователям видеть мир с точки зрения робота. Используя 3D мы можем визуализировать данные с топиков: это может быть изображение с камеры, 3D облако точек, данные с лазерного дальномера робота, двух- и трехмерные карты, получаемые в результате картографирования. Также можно получить информацию о положении суставов (джоинтов — англ. joints) робота, о системах координат каждого сочленения робота. Запустить RViz (если уже запущен ROS-Мастер) можно командой:

rosrun rviz rviz

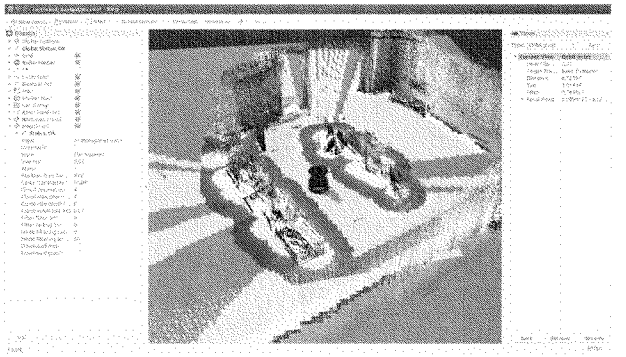

Рис. 5.4: Пример запущенного симулятора RViz. Отображены 2D-карта, получаемое 3D-облако точек и внешний вид робота.

Пример запущенного симулятора RViz можно увидеть на рисунке 5.4. В главном окне 3D-визуализации отображаются данные с датчиков, различные системы координат (мировые, базы роботов и их сочленений) и другие виды информации. Слева находится панель "Displays". В ней находится информация о том, что выбрано для отображения и с какого топика идет чтение данных. Например, на рисунке 5.4 видно, что выбрано отображение облака точек (MapCloud) и чтение данных идет с топика (/rtabmap/mapData). Внизу панели "Displays"находятся кнопки редактирования списка данных. Кнопка "Add"для добавления новых источников данных, "Remove— для удаления из списка и т.д. Сверху находится панель инструментов. В ней мы можем изменять положение камеры, отдалять и приближать изображение, а также устанавливать цели для робота, если среди запущенных нод имеются планировщики пути.

Итак, симулятор RViz необходим, чтобы отображать различные данные с сенсоров робота и другие данные. Но как создать виртуальную среду для тестирования роботов. Окружение робота можно сформировать с помощью симулятора Gazebo.

5.3 Симулятор Gazebo

Gazebo 3D, также как и ROS, разрабатывается некоммерческой организацией OSRF (Open Source Robotics Foundation). Он бесплатный и имеет открытый код. Кроме того, он очень популярен среди мирового робототехнического сообщества и является официальным симулятором соревнований DARPA. Gazebo отлично интегрируется с программной платформой ROS (Robot Operating System), а значит разработанную вами программу управления виртуальным роботом в Gazebo и ROS будет

относительно несложно перенести на реального робота.

Gazebo позволяет моделировать динамику и кинематику механизмов роботов (включая моменты взаимодействия с внешней средой) и формировать физически правдоподобные показания виртуальных датчиков.

Симулятор Gazebo имеет свой собственный редактор, позволяющий без программирования создавать трехмерные сцены и включающий огромную библиотеку моделей. Программа также предоставляет следующие возможности:

1. Использование популярных общеизвестных моделей роботов, таких как: iRobot Create, PR2, TurtleBot, Pioneer 2 DX, Segway RMP, Pioneer 2 AT. Помимо заранее созданных разработчиками моделей есть возможность самостоятельного проектирования необходимых устройств (сенсоров и роботов), загрузки их в мир и дальнейшей симуляции. Однако устройства, модели которых уже есть в программе, эмулируются с гораздо более высокой точностью.

2. Поддержка и симулирование работы множества различных сенсоров, в том числе сонара, лазерного дальномера, датчиков семейства IMU, моно- и стереокамер, кинект-сенсоров, прибора для чтения RFID-меток и других.

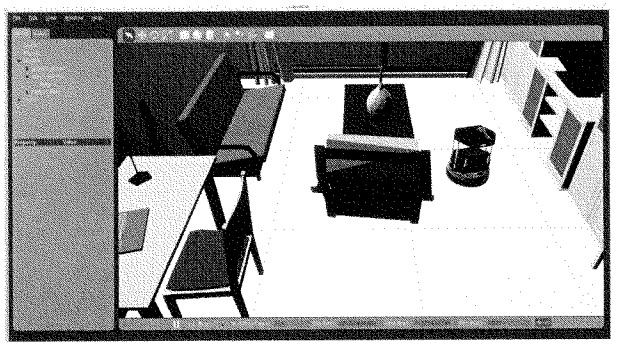

Рис. 5.5: Окно Gazebo. Создана среда, имитирующая жилую комнату, в ней находится робот Turtlebot.

Симулятор состоит из сервера (***gzserver***), который занимается просчетом физики, столкновений симуляцией, сенсоров. В качестве физического движка там используется в том числе библиотека Bullet3, которая может ускорять вычисления с помощью OpenCL на GPU. К серверу могут подсоединятся клиенты — ***gzclient*** — один из них. Именно он занимается отображением графического интерфейса. Для сенсоров и моделей

можно писать свои плагины — это позволяет управлять самой моделью и симулировать входные данные от датчиков. Сервер и клиент Gazebo запускаются последовательно после ввода в терминале команды:

gazebo

После запуска этой команды вы увидите пустую 3D-среду, куда можно добавлять стандартные объекты из верхней панели 5.5. Однако чаще всего вызов среды Gazebo с размещенном в какой либо среде робота происходит в .launch файлах. Пример созданной среды и робота в ней можно увидеть на рисунке 5.5.

Создаваемое окружение описывается в формате SDF (Simulation Description Format) и обычно имеет расширение .world. В нем могут описываться все аспекты окружающего мира: освещение, ветер, гравитация, магнитное поле, и т.д. Кроме того, в нем же описаны добавленные в среду статичные (дома, мебель) и динамические (мобильные роботы) объекты и их первоначальное положение. Пример такого файла на рисунке 5.6.

```
<?xml version="1.0" ?>
<sdf version="1.4">
  <world name="default">
    <!-- A global light source -->
    <include>
      <uri>model://sun</uri>
    </include>
    <!-- A ground plane -->
    <include>
      <uri>model://ground_plane</uri>
    </include>

    <model name='my_mesh'>
      <pose>10 0 0 0 0 0</pose>
      <static>true</static>
      <link name='body'>
        <visual name='parallel_wall'>
            <geometry>
                <mesh><uri>file://maze_parallel_wall.dae</uri></mesh>
            </geometry>
        </visual>
      </link>
    </model>
  </world>
</sdf>
```

Рис. 5.6: Пример .world файла с добавленным освещением, поверхностью и 3D-моделью лабиринта.

Заключение

В данном пособии описано что такое Робототехническая Операционная Система, объяснены основные понятия ROS. Были приведены примеры использования ROS перечислено какие средства командной строки и программы с графическим интерфейсом можно использовать для анализа и создания собственных ROS-систем. Кроме того, проведен обзор симуляторов, которые как отображают данные с сенсоров (RViz) так и создают физическую симуляцию для генерации этих данных (Gazebo)

В следующем пособии будут разобраны примеры программирования собственных алгоритмов для ROS и создание собственных роботов.

Все предложения и замечания просьба высылать по адресу *lavrenov@it.kfu.ru.*

Список литературы

1. *Quigley M.* Programming Robots with ROS: a practical introduction to the Robot Operating System / Quigley M., Gerkey B., Smart W. D. — "O'Reilly Media, Inc. 2015. — 448 с.

2. *Koubâa A.* Robot Operating System (ROS) / Quigley M. – Verlag : Springer, 2017. — 728 с.

3. *Martinez A.* Learning ROS for robotics programming/ Martinez A., Fernández E. — Packt Publishing Ltd, 2013. — 332 с.

4. ROS Tutorials. — URL:http://wiki.ros.org/ROS/Tutorials (дата обращения 15.05.2019)

Учебное издание

Лавренов Роман Олегович
Магид Евгений Аркадьевич

ОСНОВЫ РОБОТОТЕХНИЧЕСКОЙ ОПЕРАЦИОННОЙ СИСТЕМЫ (ROS)

Учебно-методическое пособие

Подписано в печать 24.05.2019.
Бумага офсетная. Печать цифровая.
Формат 60x84 1/16. Гарнитура «Book Antiqua». Усл. печ. л. 2,09.
Уч.-изд. л. 1,13. Тираж 50 экз. Заказ 217/5

Отпечатано с готового оригинал-макета
в типографии Издательства Казанского университета

420008, г. Казань, ул. Профессора Нужина, 1/37
тел. (843) 233-73-59, 233-73-28