Петрозаводский государственный университет
Институт математики и информационных технологий
Кафедра информатики и математического обеспечения

Направление подготовки бакалавриата

09.03.04 – Программная инженерия

Профиль направления подготовки бакалавриата

Системное и прикладное программное обеспечение

Отчет о прохождении производственной практики

Преимущества и недостатки ROS и ROS 2

Выполнила:
студентка 2 курса группы 22207

Е. Ф. Волкова _____
подпись

Место прохождения практики:
кафедра ИМО, ЦИИ

Период прохождения практики:

_____
период

Руководитель:
О. Ю. Богоявленская

_____
подпись

Итоговая оценка:

_____
оценка

Петрозаводск — 2022

Содержание

Введение

В последнее время в области робототехники особое внимание уделяется платформам. Понятие платформа обычно резделяют на программную платформу и аппаратную платформу. Программная платформа для роботов включает в себя набор инструментов, которые используются для разработки программного обеспечения роботов. Аппаратные платформы, включают в себя готовые исследовательские и образовательные устройства, а также готовые промышленные системы.

В настоящий момент для целей изучения и погружения в робототехнику однозначно рекомендуется ROS. Важными критериями на этапе погружения в робототехнику, являются: активность сообщества, наличие различных библиотек, расширяемость и простота использования. По этим критериям в данный момент равных ROS нет.

В данной работе была изучена система ROS, а также система следующего этапа развития ROS — система ROS 2.

Так, целью практики является сравнение систем ROS и ROS 2.

Задачи практики:

1. изучить обе системы;
2. найти достоинства и недостатки у данных систем;
3. сравнить системы.

1 ROS

1.1 Общие сведения

Операционная система для роботов (ROS) — это набор программных библиотек и инструментов, помогающих создавать приложения для роботов. ROS обеспечивает аппаратную абстракцию, предлагает драйверы устройств, библиотеки, визуализаторы, обмен сообщениями, менеджеры пакетов и многое другое. ROS выпускается в соответствии с условиями BSD лицензии и с открытым исходным кодом.

ROS была первоначально разработана в 2007 году под названием Switchyard Стэнфордской лабораторией искусственного интеллекта в рамках проекта Stanford AI Robot STAIR (STanford AI Robot). С 2008 по 2013 годы разработка велась в основном в Willow Garage, научно-исследовательском институте / инкубаторе робототехники. Тем временем исследователи из более чем двадцати организаций сотрудничали с инженерами Willow Garage в рамках федеративной модели разработки.

Версии ROS могут быть несовместимы друг с другом и часто обозначаются кодовым именем, а не номером версии. На сегодняшний день основные версии: Melodic Morenia, Noetic Ninjemys.

Рис. 1 – Активные версии ROS

1.2 Установка ROS

Для установки ROS noetic на ubuntu необходима версия 20.04, более поздняя версия не поддерживается.

Порядок установки:

1. Настройка компьютера для получения программного обеспечения из packeges.ros.org.
2. Настройка ключей (для этого понадобилось установить curl - утилиту для скачивания файлов).
3. Установка пакетов (была выбрана версия ROS-Base).
4. Настройка среды для работы с ROS при помощи функции source (необходимо запускать каждый раз при работе из терминала, но можно также добавить в автоматический скрипт).
5. Установка заыисимых пакетов (python3-rosdep python3-rosinstall python3-rosinstall-generator python3-wstool build-essential).
6. Установка python3-rosdep - необходима для упрощение установки системных зависимостей для ресурсов, которые будут компилироваться а также для некоторых компоненотов ROS.

При необходимости можно устанавливать дополнительные пакеты.

1.3 Создание и сборка ROS пакета

Требования к пакету:

1. Пакет должен содержать catkin-совместимый packege.xml файл, который содержит метаинформацию о пакете (catkin - система сборки от ROS).
2. Пакет должен содержать CMakeList.txt, который использует catkin.
3. Каждый пакет должен иметь собственную папку.

Создание и сборка пакета:

1. Для начала необходимо создать директорию с названием рабочей среды, которая будет содержать внутри себя директорию *src/*.

2. Внутри созданной директории *src/* происходит инициализация рабочей среды с помощью команды *catkin_init_workspace*, затем сгенерируется файл CMakeList.txt.

3. Затем выполняется команда *catkin_create_pkg*, на вход которой даётся название для нового пакета, а также прописываются зависимости *rospy* и *std_msg*. После этого внутри директории *src/* появится директрия с названием пакета, содержащая файлы CMakeList.txt и package.xml, а также директорию *src/*.

4. Приступаем к сборке. Для этого в исходной директории запускаем команду *catkin_make*. После сборки пакетов в директории появятся папки *build* и *devel*.

5. Пакеты собраны, теперь с помощью функции *ros run* можно осуществлять необходимые действия над пакетами.

2 ROS 2

2.1 Общие сведения

Операционная система робота 2 (ROS 2) является второй версией ROS, который является коммуникационным интерфейсом, который позволяет различным частям системы робота обнаружить, отправить и получить данные.

Первая официальная стабильная версия ROS2 была выпущена в декабре 2017 года. Дистрибутив назвали «Ardent Apalone». Далее был релиз в июне 2018, с именем «Bouncy Bolson». «Crystal Clemmys» появился в декабре 2018. «Dashing Diademata» в мае 2019 г. Как видно, дистрибутивы обновляются достаточно быстро. Так, перечисленные дистрибутивы уже вышли из эксплуатации. Взамен пришли другие: «Foxy Fitzroy», «Galactic Geochelone», «Humble Hawksbill». В разработке «Rolling Ridley».

Рис. 2 – Активные версии ROS 2

ROS 2 поддерживает широкий спектр приложений робототехники, от образования и исследований до разработки и внедрения продуктов. Он включает в себя большой набор взаимосвязанных программных компонентов, которые обычно используются для разработки приложений робототехники.

2.2 Установка ROS 2

В качестве операционной системы для установки ROS 2 была выбрана Windows (поддерживается только 10 версия).

1. Сначала требовалось установить необходимые компоненты, а именно:
   - Chocolatey — менеджер пакетов для Windows;
   - Python version 3.8.3;
   - Visual Studio с распространяемыми компонентами;
   - OpenSSL v1.1.1o;
   - Qt5;
   - Graphviz;
2. Из активных дистрибутивов для установки был выбран «Humble Hawksbill». Необходимо было загрузить релизную версию и прилагающие бинарные файлы.
3. Также понадобилось скачать CMake и некоторые зависимости, недоступные в базе данных пакетов Chocolatey.
4. Для взаимодействия в режиме «real-time» потребовалось установить RTI Connext DDS.
5. Следующим этапом была настройка среды с помощью команды *call*.
6. Когда рабочая область настроена, можно приступать к выполнению различных примеров для подтверждения корректной работы ROS 2. Например, C++ talker и Python listener.

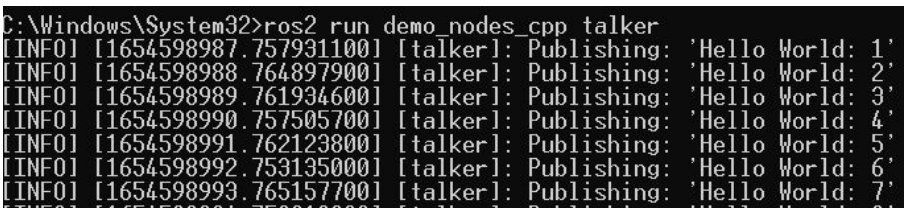

Рис. 3 – C++ talker

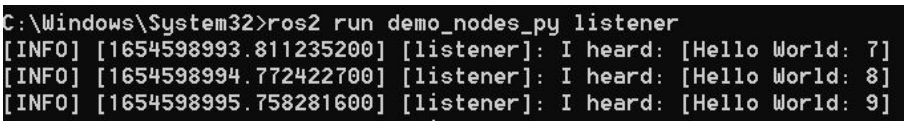

Рис. 4 – Python listener

2.3 Создание и сборка пакета в ROS 2

Каждый пакет ROS 2 Python и CMake имеет собственное минимально необходимое содержимое:

- package.xml – файл, содержащий метаинформацию о пакете;
- setup.py содержит инструкции по установке пакета;
- setup.cfg требуется, когда в пакете содержатся исполняемые файлы;
- /<package_name>- каталог с тем же именем, что и у вашего пакета, используемый инструментами ROS 2 для поиска вашего пакета, содержит__init__.py

Шаги создания и сборки пакета:

1. Для начала необходимо создать папку, в которой будет создаваться пакет.
2. Далее с помощью команды *pkg create* и аргумента $--node-name$ создаем пакет с исполняемым файлом типа Hello World.
3. После запуска команды ваш терминал вернет сообщение с информацией о создании пакета, сгенерированных файлов и т.д. (см. Рис.5).
4. Для того, чтобы собрать пакет, необходимо воспользоваться командой *colcon build* $--merge-install$.

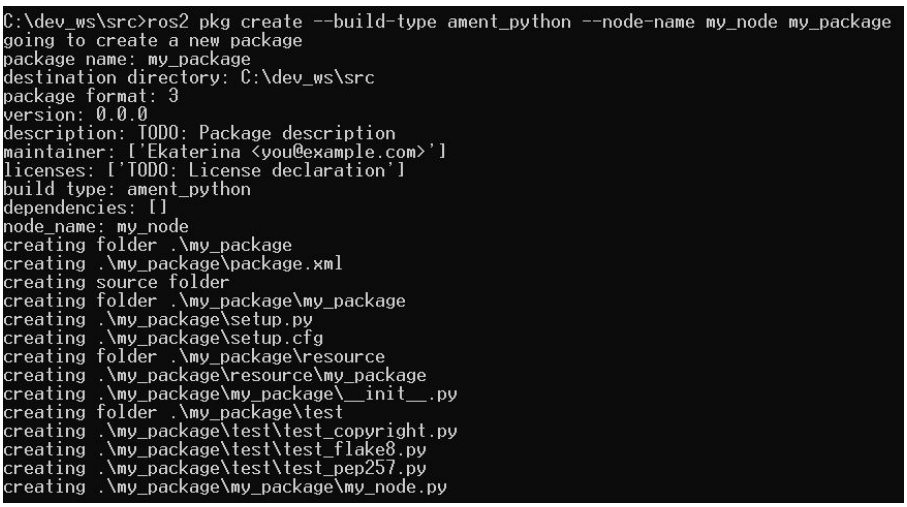

Рис. 5 – Создание пакета

5. Чтобы запустить исполняемый файл, который вы создали с помощью $--node-name$ аргумента при создании пакета, введите команду: $ros2\ run\ my\_package\ my\_node$. После терминал должен вывести сообщение:

Рис. 6 – Запуск пакета

3 Сравнение систем

3.1 Преимущества и недостатки ROS

Преимущества:

– В данный момент версия Python2 не поддерживается, но ROS вплоть до финальной версии поддерживает разработку на этой версии Python;

– ROS поддерживает разработку на cpp 11/14, будучи изначально нацеленной на cpp 98;

– Добавлена функциональность $Nodelets$, с помощью которой можно записывать несколько узлов в один и тот же исполняемый файл с обменом данными внутри процесса.

Недостатки:

– В 2025 году планируется выпустить окончательную версию ROS, после чего больше не планируется разработка новых дистрибутивов;

– Для cpp и python используются разные, не связанные друг с другом библиотеки, что затруняет разработку на обоих языках;

– Использование XML для написания файлов запуска;

– Отсутствие специфичной структуры, которая сказала бы вам, как следует прописывать функционал узла;

– Также можно добавлять функции обратного вызова в любом месте программы и использовать ООП, но каждая такая реализация должна быть уникальной.

3.2 Преимущества и недостатки ROS 2

К преимуществам можно отнести следующие особенности системы ROS 2:

– ROS 2 позволяет устанавливать безопасные соединения между компонентами системы (нодами);

– взаимодействие осуществляется в режиме «real-time» через концепцию DDS (Data Distributed Services);

– соединение нескольких роботов в одну сеть упрощено;

– реализуется уровень ROS-архитектуры непосредственно на аппаратном обеспечении;

– используются новейшие пакеты обновлений (ROS 1 обновляется не так часто);

Помимо преимуществ, существуют и некоторые недостатки:

– Использование XML для написания файлов запуска;

– ROS 2 адаптирован не для всех продуктов DDS;

3.3 Сравнительная таблица

На заключительном этапе работы сравним системы ROS и ROS 2 в общих чертах, для наглядности заполним таблицу.

Критерий	ROS	ROS2
Поддерживаемые ОС	Linux Ubuntu, OS X	Linux, Windows 10, OS X
Поддерживаемые языки программирования	Cpp до 14, Python3, Python2	Python3, Cpp 11, 14, 17, C
Возможнось добавления новых модификаций	Затруднена, так как модификации требуют большого количества изменений, делающих систему нестабильной	ROS2 была разработана с нуля, поэтому никаких затруднений нет, модификации предусмотрены
Поддержка дистрибутивов	Финальная версия системы будет выпущена в 2025 году, после чего система не будет обновляться	Новые ROS2 версии реализуются каждый год, обеспечивая долгосрочную поддержку
API	Система поддерживает две независимые библиотеки для Cpp и Python	Существует только одна базовая библиотека с именем rcl на C, содержащая все основные функции ROS2. Возможно использование клиентских библиотек поверх rcl
ООП	Отсутствие специфичной структуры, которая бы говорила, как следует прописывать функционал узла	Для написания узла, необходимо создать класс, который наследуется от объекта Node (экономия времени)
Несколько вершин в одном исполняемом файле	Наличие функциональности *Nodelets*	В ROS2 *Nodelets* была непосредственно включена в ядро ROS2 и теперь называется «компонентами»
Файлы запуска	Используется XML	Используется Python, XML

Заключение

Таким образом, были изучены системы ROS и ROS 2, выявлены у обеих систем преимущества и недостатки, а также выполнен сравнительный анализ между ними в виде таблицы.

Итак, можно сказать, что поставленная цель практики достигнута.

Список литературы

1. Macenski S. Robot Operating System 2: Design, architecture, and uses in the wild [Электронный ресурс]. — URL: https://www.science.org/doi/10.1126/scirobotics.abm6074 (дата обращения : 07.06.2022).

2. ROS Documentation [Электронный ресурс]. — URL: http://wiki.ros.org/ru (дата обращения : 07.06.2022).

3. ROS 2 Documentation [Электронный ресурс]. — URL: http://docs.ros.org/en/humble/index.html (дата обращения : 07.06.2022).