МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Национальный исследовательский Томский государственный университет
Томский государственный университет систем управления и радиоэлектроники
Болгарская Академия наук
ООО «Научно исследовательское предприятие «Лазерные технологии»

ИННОВАТИКА-2018

СБОРНИК МАТЕРИАЛОВ

XIV Международной школы-конференции студентов, аспирантов и молодых ученых
26–27 апреля 2018 г.
г. Томск, Россия

Под редакцией А.Н. Солдатова, С.Л. Минькова

Scientific & Technical Translations
STT
ИЗДАТЕЛЬСТВО
Томск – 2018

ОСНОВНЫЕ КОНЦЕПЦИИ ФРЕЙМВОРКА ROS

С.И. Послявский
Национальный исследовательский Томский государственный университет
travoltaj237@gmail.com

BASIC CONCEPTS OF THE ROS FRAMEWORK

S.I. Poslavskiy
National Research Tomsk State University

This article examines the structure and principles of the ROS framework for the development of robot control system architecture. The features of this framework are collated.

Keywords: Robot Operating System, Linux, filesystem level, Point Cloud Library.

Robot Operating System (ROS) – это широко используемый в робототехнике фреймворк. В ROS имеются стандартные возможности операционной системы, такие как, аппаратная абстракция, управление устройствами на низком уровне, реализована часто используемая функциональность, передача сообщений между процессами, и управление библиотеками.

Архитектура ROS основана не графе с централизованной топологией. Обработка происходит в узлах, которые могут принимать или отправлять данные с датчиков, систем контроля состояния и планирования, приводов, и так далее. Библиотека ориентирована на *Unix*-подобные системы (под *Ubuntu Linux* работает отлично, а *Fedora* и *Mac OS X* имеют статус экспериментальных).

Пакет **-ros-pkg* является общим репозиторием для разработки высокоуровневых библиотек. Многие из возможностей часто ассоциируемые с ROS, такие как библиотеки навигации и визуализатор *Rviz*, хранятся в этом репозитории. Эти библиотеки предоставляют мощный набор инструментов (различные визуализаторы, симуляторы, средства отладки) для упрощения работы.

ROS распространяется на условиях лицензии BSD и является ПО с открытым исходным кодом. ROS бесплатна для исследовательских и коммерческих целей. ROS способствует повторному использованию кода, так что разработчики робототехники и ученые могут не изобретать колесо постоянно. Можно получить код из репозитория, изменить его и вновь поделиться улучшенным ПО. Имеется возможность написать драйвер собственного датчика для ROS.

ROS поддерживает параллельные вычисления, имеет хорошую интеграцию с популярными C++ библиотеками, такими как как *OpenCV, Qt*, *Point Cloud Library* и пр., и она может работать на одноплатных компьютерах, таких как *Raspberry Pi* или *BeagleBone Black*, а также с микроконтроллерными платформа-ми, например, *Arduino* [1].

В архитектуре ROS можно выделить три концептуальных уровня:

1) уровень файловой системы (*Filesystem level*);
2) уровень вычислительного графа (*Computation Graph level*);
3) уровень сообщества (*Community level*).

Первый уровень – это уровень файловой системы. На этом уровне расположена внутренняя структура ROS – структура папок, файлы, необходимые для работы.

Второй уровень – это уровень вычислительного графа, на котором происходит взаимодействие между процессами и системами. На этом уровне находятся концепции и модули, которые имеются в ROS для создания систем, обработки всех процессов, коммуникации с более чем одним компьютером и так далее.

Третий уровень – это уровень сообщества. Этот уровень содержит инструменты и концепции для обмена знаниями, алгоритмы и код от любого разработчика. Этот уровень очень важен, поскольку ROS быстро растет при мощной поддержке со стороны сообщества.

Подробнее рассмотрим первый уровень архитектуры ROS.

Как и в случае обычной операционной системы, программы в ROS разделены на папки, в которых содержатся некоторые файлы, описывающие ее функциональность:

1) Пакеты (*Packages*): формируют атомарный уровень ROS. Пакет имеет минимальную структуру и содержимое, чтобы создать программу в ROS. Он может иметь выполняемые процессы (узлы или *node*), файлы конфигурации и так далее;
2) Декларации (*Manifests*): содержится информация о пакетах, лицензионная информация, зависимости, флаги компиляции и прочее. Управление декларациями осуществляется через файл manifests.xml;
3) Стеки (Stacks): когда вы собираете вместе несколько пакетов для получения некоторой функциональности, то получите стек. В ROS, существует много таких стеков для различных целей, например, стек навигации;

4) Декларации стеков (*Stack manifests*): предоставляют данные о стеке, включая его лицензионную информацию и его зависимости от других стеков;
5) Типы сообщений (*Message types, msg*): сообщение является информацией, которую процесс отправляет другим процессам. В ROS имеется множество стандартных типов сообщений. Описание сообщения сохраняется в *my_package/msg/MyMessageType.msg*;
7) Типы сервисов (*Service types, src*): описания сервисов хранятся в *my_package/srv/MyServiceType.srv*. Определяют в ROS структуры данных запросов и ответов для сервисов.

В рамках этой работы были рассмотрены основные концепции и архитектура фреймворка ROS. Философией ROS является создание программного обеспечения, позволяющего работать с различными роботами, лишь внеся небольшие изменения в код. Эта идея позволяет создавать функциональность, которая может быть достаточно просто перенесена для использования различными роботами, избежав постоянного «изобретения колеса» [2].

Работа выполнена при финансовой поддержке Минобрнауки России, уникальный идентификатор проекта RFMEFI57817X0241.

Литература

1. Powering the world's robots [Электронный ресурс]. – URL: http://www.ros.org/ (дата обращения: 08.02.2018).
2. Willow Garage [Электронный ресурс]. – URL: http://www.willowgarage.com/pages/ software/rosplatform (дата обращения: 06.04.2018).