уровень переменной составляющей тока якорной цепи. Так при изменении $\Delta U_{об}$ от 0.5% до 2% и скорости вращения от номинальной до максимальной относительная амплитуда переменной составляющей тока якорной цепи возрастает от 0,1827 до 2,43.

3. Значительное увеличение относительной амплитуды переменной составляющей тока якорной цепи (до предельного значения) при максимальной скорости вращения (за счет ослабления потока возбуждения в два раза) обусловлено увеличением коэффициента усиления регулятора скорости и снижением электромагнитного момента в два раза.

4. Относительные дополнительные потери активной мощности в якорной цепи двигателя в пределах от 0,0018 до 0,326, реактивной мощности - от 0,00025 до 0,226. обусловленные переменной составляющей тока якорной цепи, определяются пульсациями напряжения тахогенератора. Их амплитуда растёт по мере ухудшения качества стыковки валов двигателя и тахогенератора и возрастания частоты вращения прокатного двигателя.

Список литературы

1. Тун А. Я. Системы контроля скорости электропривода. - М.: Энегоатомиздат, 1984 г. -167 с.

2. Мерзляков Ю.В., Толмачев Г.Г., Карандаев А.С., Галкин В.В., Головин В.В., Хлыстов А.И. Исследование условий коммутации двигателей электроприводов широкополосного стана горячей прокатки // Известия ТулГУ. Технические науки. Вып. 3: в 5 ч. Тула: Изд-во ТулГУ, 2010. Ч. 3. С. 89 – 96.

3. Косматов В.И., Мерзляков Ю.В., Толмачев Г.Г., Киселев С.Н. Оценка влияния оборотных пульсаций напряжения тахогенератора на дополнительные потери мощности в электроприводе // Изв. Вузов Электромеханика. 2006. № 4. С. 35 – 37.

УДК 621.382

ПРОГРАММНЫЕ РЕШЕНИЯ ДЛЯ РАЗРАБОТКИ АРХИТЕКТУРЫ СИСТЕМЫ УПРАВЛЕНИЯ РОБОТОМ

магистрант У.В. Михайлова, магистрант Е.А. Михайлов, руководитель – А.С. Сарваров

ФГБОУ ВПО «Магнитогорский государственный технический университет им. Г.И. Носова», Россия, г. Магнитогорск
ylianapost@gmail.com

Аннотация

В данной статье рассмотрены основные программные решения в области разработки архитектуры системы управления роботом. Проведен их краткий анализ и сделаны выводы.

Ключевые слова: робот, система управления, программное обеспечение для управления роботами.

SOFTWARE SOLUTIONS FOR THE DEVELOPMENT OF ROBOT CONTROL SYSTEM ARCHITECTURE

undergraduate U.V. Mikhailova, undergraduate E.A. Mikhailov, supervisor – A.S. Sarvarov

Nosov Magnitogorsk State Technical University
Russia, Magnitogorsk
ylianapost@gmail.com

Abstract

This article describes the major software solutions in the field of robot control system architecture. Held their brief analysis and conclusions are made.

Key words: robot, control system, software to control robots.

Робототехника является научной и практической областью, которая занимается исследованиями, разработкой и производством роботов различного типа и назначения (промышленных, исследовательских, сервисных, домашних и др.)

В процессе разработки архитектуры системы управления роботом необходима большая открытость управляющих систем. Наиболее важным фактором успеха при разработке архитектуры системы управления роботом является использование распространенной элементной базы и программного обеспечения с открытым кодом. Другой важный фактор при создании современной открытой архитектуры управляющей системы робота - это использование программного обеспечения с открытым кодом (ПООК) [1].

В данной статье рассмотрим существующие программные решения для разработки, архитектуры системы управления и программирования роботов.

Первое из рассматриваемых программных решений это программное обеспечение с открытым исходным кодом операционная система для роботов ROS (Robot Operating System). ROS выпускается в соответствии с условиями BSD-лицензии, которая является одной из самых популярных лицензий для свободного программного обеспече-

ния. ROS это фреймворк для программирования роботов, предоставляющий функциональные возможности для распределённой работы. Фреймворк ROS разработан в 2007 году совместными усилиями исследовательской лаборатории Willow Garage и лаборатории искусственного интеллекта Стэнфордского университета.

Проект ROS реализует системный подход управления роботом, а на его основе разработаны прикладные пакеты (ros-pkg): OpenCV - библиотека машинного зрения, система планирования действий; Player - сервер управления и другие технологии.

Основная задача фреймворка ROS – это возможность повторного использования кода в робототехнических исследованиях и разработках. Фреймфорк ROS интегрирует различные драйверы, алгоритмы и популярные открытые робототехнические библиотеки. Фреймфорк ROS предоставляет функционал операционной системы робота: аппаратная абстракция, низкоуровневый контроль оборудования, реализация часто используемого функционала, передача сообщений между процессами, управление пакетами. Фреймфорк ROS не является системой реального времени, но может использовать системы реального времени (например, OROCOS Real-time Toolkit).

Второе из рассматриваемых решений это открытое программное обеспечение для управления роботами - OROCOS (Open Robot Control Software project). OROCOS - это набор библиотек, следовательно, у OROCOS отсутствуют графические инструменты разработки и собственный симулятор.

Проект OROCOS поддерживает направления четырёх библиотек *C++*: The Real-Time Toolkit; The Kinematics and Dynamics Library; The Bayesian Filtering Library; The Orocos Component Library.

Проект OROCOS состоит из следующих библиотек:

1. OCL (The Orocos Components Library) выдает готовые к использованию компоненты управления, а также компоненты для управления и доступа к аппаратным средствам;

2. RTT (The Orocos Real-Time Toolkit) обеспечивает инфраструктуру и функциональность других приложений. Данная библиотека акцентируется на приложениях реального времени, позволяющих интерактивно управлять модулями системы;

3. BFL (Orocos Bayesian Filtering Library) обеспечивает рекурсивную обработку информации в соответствии с алгоритмами оценки, применяемых в Bayes'rule, тем самым обеспечивая независимую структуру (framework) для Dynamic Bayesian Networks;

4. KDL (The Orocos Kinematics and Dynamics Library) является частью программ, разработанных на языке C++, которая, позволяет рассчитывать в реальном времени кинематику.

Еще одно из решений система URBI разработанная французской компанией Gostai. URBI - кросс-платформенная открытая программная платформа написанная на языке C++, используемая при разработки приложений для сложных систем и робототехники. URBI базируется на распределенной компонентной архитектуре UObject. Она включает параллельный и событийный скриптовый язык urbiScript. URBI работает под Windows, Linux, и MacOS.

Набор программных продуктов входящих в систему URBI:

1. Gostai Urbi SDK - комплект средств разработки для создания компонентов Urbi, т.н. UObject - промежуточной слой архитектуры Urbi. Он так же поддерживает работу скриптового языка urbiscript, который используется для программирования управления роботами;

2. Gostai Lab - программа, включающая в себя свойства Gostai Console, позволяющая строить виртуальный пульт дистанционного управления роботом. Она предоставляет различные виджеты для визуализации данных от робота (включая видео и звук), а также позволяет изменять состояние робота. Визуальные виджеты связываются с различными переменными urbiScript. Например, можно связать один виджет с камерой, а другой виджет будет использован для связи численной переменной и слайдера;

3. Gostai Console (urbiConsole) - утилита для win32 с графическим интерфейсом, для подключения к удалённому Urbi-серверу. Так же можно использовать telnet;

4. Gostai Studio - IDE для создания графических программ, определяющих поведение сложных систем или роботов. Программа, включающая в себя все функции Gostai Console. Это высокоуровневая интегрированная среда разработки для Urbi, которая базируется на иерархических конечных автоматов (HFSM - Hierarchical Finite State Machines). Имеется визуальное отслеживание выполнения кода в реальном времени и интуитивный визуальный редактор.

Следующее из решений это программное обеспечение Microsoft Robotics Developer Studio. Microsoft Robotics Developer Studio (Microsoft RDS, MRDS) — Windows-ориентированная среда разработки приложений для роботизированных платформ. Robotics Studio выпущен в 2006 году. Robotics Studio содержит инструменты визуального программирования и трехмерную виртуальную среду для физической симуляции работы роботов - PhysX.

Составляющие компоненты Microsoft Robotics Developer Studio:

1. Runtime environment – окружение с приложением для роботов, в котором происходит отслеживание и взаимодействие с другими приложениями роботов. В основе лежит CLR 2.0, в результате есть возможность писать приложения с использованием любых языков про-

граммирования платформы Microsoft.NET. Окружение Runtime environment состоит из двух элементов: CCR (Concurrency and Coordination Runtime, библиотека параллельных вычислений и координации) — библиотека для работы с параллельными и асинхронными потоками данных; DSS (Decentralized Software Services, децентрализованные программные сервисы) – средство создания распределенных приложений на основе сервисов (для работы и взаимодействия используется протокол Decentralized System Services Protocol (DSSP), который основан на протоколе SOAP - для обмена произвольными сообщениями используется формат XML).

2. VPL (Visual Programming Language) - язык визуального программирования написания приложений для роботов (диаграммы VPL сохраняются в виде XML-схем).

3. Simulation environment - симулятор-окружение для роботов (основана на модели физики в Microsoft Robotics Studio однако она достаточно упрощена, поэтому этот симулятор не подойдет там, где нужны точные расчеты.).

Приложение в Robotics Studio - это композиция параллельно выполняющихся слабосвязанных компонентов. При этом все компоненты это независимо исполняемые сервисы. Пакет RDS позволяет разрабатывать программы для различных аппаратных платформ. Недостаток RDS это зависимость от Windows и закрытые исходники. Кроме того Robotics Developer Studio не имеет встроенных систем компьютерного зрения, навигации и машинного обучения. Протокол SOAP, используемый в Robotics Studio для взаимодействия распределенных сервисов, не предназначен для приложений, работающих в режиме реального времени.

Еще одно решение проект Player (ранее Player/Stage/Gazebo). Представляет собой проект по созданию свободного программного обеспечения для исследования сенсорных систем и робототехники. Проект Player, обеспечивает сетевой интерфейс для разнообразного сенсорного оборудования и роботов. Клиент-серверная модель Player позволяет писать программы управления робота на любом языке программирования и работать на любом компьютере, подключённого к сети вместе с роботом. Проект состоит из 3 основных компонентов: сервер Player; платформа для симуляции роботов – двумерный симулятор Stage; трёхмерный симулятор Gazebo. Компоненты проекта работают на POSIX-совместимых операционных системах, включая Linux, MacOS X, Solaris и BSD; планируется портирование на Microsoft Windows. Проект был основан Brian Gerkey, Richard Vaughan и Andrew Howard в университете Лос-Анджелеса в 2000 году и широко используется для проведения исследований в робототехнике и в про-

цессе обучения. Программное обеспечение распространяется под лицензией GNU General Public License с документацией под GNU Free Documentation License.

Особенности проекта:

1. Независимость от робототехнической платформы.

2. Поддержка нескольких языков программирования, включая C, C++, Java, Tcl и Python.

3. Минимальный и гибкий дизайн.

4. Поддержка нескольких устройств в одном интерфейсе.

5. Конфигурации сервера «на лету».

Проект Player реализован путём создания нескольких уровней абстракции. Он скрывает низкоуровневые аппаратно-зависимые реализации за набором установленных «интерфейсов». Высоко-уровневый контроль взаимодействует только с этими интерфейсами и программа управления может не учитывать что скрывается, за этими интерфейсами.

Следующее решение ERSP от компании Evolution Robotics - это среда для разработки программного обеспечения роботов. Она состоит из трех основных частей:

1. ViPR - модуль визуального распознавания.

2. vSLAM - модуль ориентирования, основанный на данных от одной камеры и оптических энкодеров, позволяющий осуществлять локализацию и построение карты местности с точностью до 10 см.

3. ERSA - операционная система робота, предоставляющая всю инфраструктуру и функционал для управления всеми аппаратными и программными компонентами робота.

Система SLAM и система компьютерного зрения основана на алгоритме SIFT. Для платформы ERSP существует программа визуального программирования на основе различных «поведенческих» блоков. Кроме того, в ERSP возможна разработка программ на языке Питон (Python). ERSP – кросс-платформенная разработка, которая поддерживает следующие платформы: 32-bit(64-bit) Fedora Core 8; 32-bit(64-bit) Debian 4.0;32-bit Windows XP.

Еще одно решение - LabVIEW Robotics. LabVIEW (Laboratory Virtual Instrumentation Engineering Workbench) - это кросс-платформенная графическая среда разработки и платформа для выполнения программ, созданных на графическом языке программирования “G” фирмы National Instruments. В основе LabVIEW лежит парадигма потоков данных. LabVIEW используется в системах сбора и обработки данных, а также для управления технологическими процессами и техническими объектами. LabVIEW поддерживает широкий спектр оборудования различных производителей и содержит много-

численные библиотеки компонентов, в том числе, для управления роботами и системами машинного зрения. Например, на LabView основана среда программирования Lego NXT.

В состав LabVIEW Robotics входит:

1. Pyro – инструментарий управления роботом, написанный на языке Python.

2. OpenRAVE – трёхмерный симулятор.

3. YARP – фреймворк с открытым исходным кодом, предназначенный для работы с оборудованием робота и написанный на языке C++.

4. CLARAty – программная платформа, распространяемая в виде open source-проекта. Реализует широкий спектр алгоритмов.

В результате проведенного анализа существующих решений программного обеспечения для разработки архитектуры системы управления роботом, можно подвести следующие общие итоги:

1. Все решения представлены в виде в промежуточного слоя между обычной ОС и программами/скриптами для управления роботом.

2. Все решения имеют модульную структуру, которая работает поверх базовой прослойки (фреймворка).

3. Все решения имеют распределённую клиент-серверную структуру.

Список литературы

1. ИНЖЕНЕР инфо. Статьи: Контроллеры. Программирование и составление программы PLC на ПЛК. – Режим доступа: http://www.ingener.info/pages-page-27-2.html (дата обращения: 27.11.13).

УДК 621.382

ИСПОЛЬЗОВАНИЕ ФРЕЙМВОРКА ROS ДЛЯ РАЗРАБОТКИ АРХИТЕКТУРЫ СИСТЕМЫ УПРАВЛЕНИЯ РОБОТОМ

магистрант У.В. Михайлова, магистрант Е.А. Михайлов, руководитель – А.С. Сарваров

ФГБОУ ВПО «Магнитогорский государственный технический университет им. Г.И. Носова», Россия, г. Магнитогорск
ylianapost@gmail.com