электронный научно-технический журнал
ИНЖЕНЕРНЫЙ ВЕСТНИК
издатель ФГБОУ ВПО «Московский государственный технический университет им. Н.Э. Баумана»

Использование robot operating system (ROS) для создания полунатурного моделирующего комплекса манипуляционных роботов

77-48211/496529

 # 11, ноябрь 2012

Калеватых И. А., Лесков А. Г.

УДК 621.865:004.896

Россия, Дмитровский филиал МГТУ им. Н.Э. Баумана
kalevatykh@gmail.com
agleskov@rambler.ru

ВВЕДЕНИЕ

Полунатурные моделирующие комплексы (ПМК) являются эффективным средством, позволяющим проводить исследования новых робототехнических систем (РТС) без необходимости изготовления опытных образцов, а также исследования РТС, функционирование которых в наземных условиях невозможно (космические манипуляционные роботы).

Как ясно из названия, ПМК позволяют отрабатывать в лаборатории элементы, подсистемы и операции РТС посредством компьютерного моделирования и отработки операций на физическом уровне. В состав ПМК может входить различное моделирующее и управляющее программное обеспечение, датчики внешней среды (камеры, датчики силомоментного очувствления (СМД), лазерные дальномеры и прочее), а также исполнительные и захватные устройства. Таким образом, ПМК сам по себе является сложной распределенной РТС. Кроме того, ПМК должен

обеспечивать перестройку на исследование новой системы с минимальными затратами.

На рисунке 1 представлен состав ПМК манипуляционных роботов Дмитровского филиала МГТУ им. Баумана [1,2]. Комплекс включает в себя промышленные роботы (ПР), СМД, захватные устройства манипулятора (ЗУМ), системы технического зрения (СТЗ) и многое другое. При очередной модернизации комплекса встал вопрос о том, как обеспечить прозрачное управление всем этим оборудованием параллельно с компьютерным моделированием, а также возможность изменения/расширения комплекса с минимальными затратами, потому что существующее программное обеспечение не разрабатывалось для таких задач.

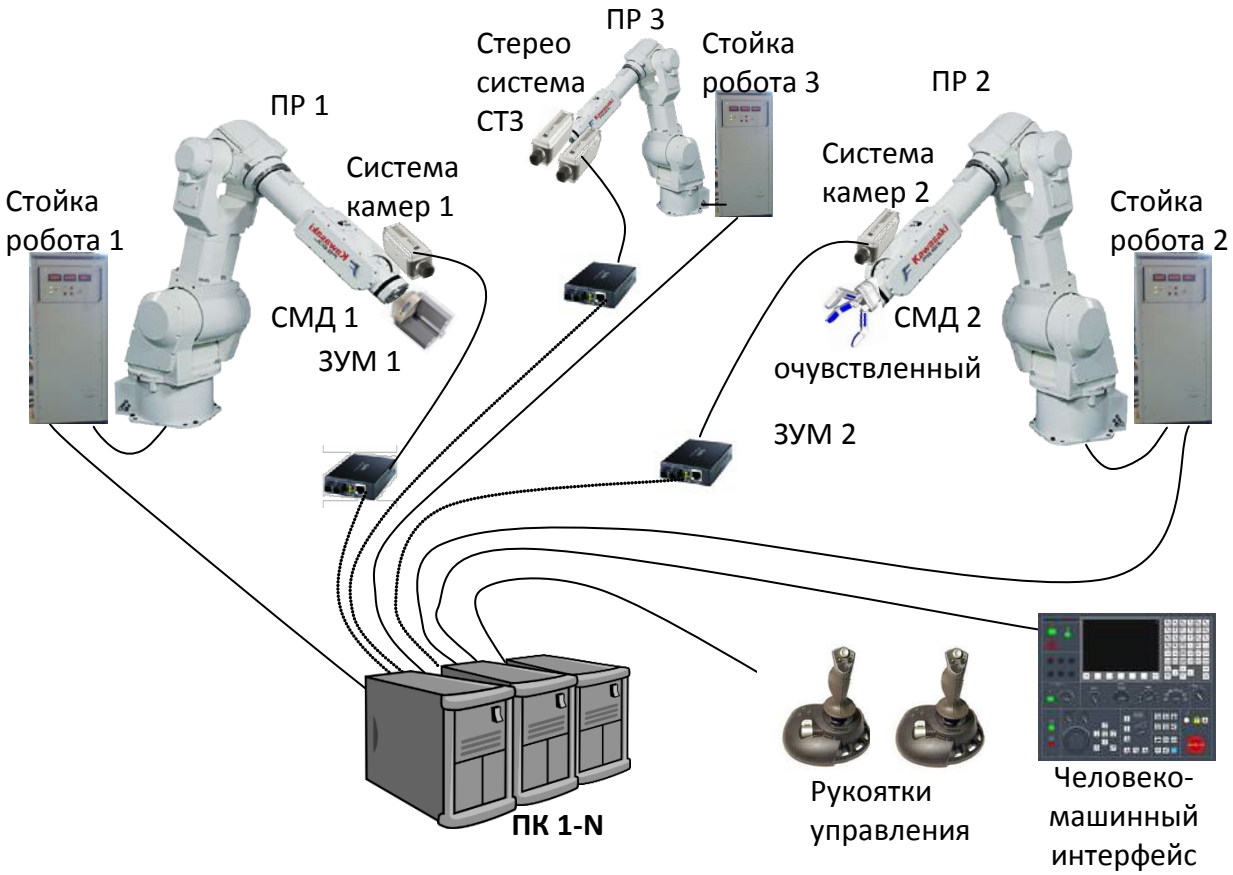

Рисунок 1. Состав ПМК манипуляционных роботов ДФ МГТУ

Разработка программного обеспечения для такого комплекса является трудоемким процессом, сопряженным с появлением множества ошибок. Сложность разработки обусловлена несколькими причинами. Неоднородность оборудования ПМК. Неоднородность программного обеспечения: ПМК должен интегрировать различные функции, в том числе планирование траекторий, машинное зрение, интерпретация данных сенсоров, команд пользователя, и интеллектуальное управление. Для каждой из этих задач разработаны свои собственные подходы, вычислительные методы, структуры данных и алгоритмы. Интеграция и обеспечение распределенной скоординированной параллельной работы всех компонентов. В мире в последнее десятилетие наблюдается растущий интерес к оптимизации и стандартизации процесса разработки приложений для роботов [3]. Поэтому перед началом разработки был проведен обзор состояния робототехнического программного обеспечения в мире с целью использования уже готовых разработок и опыта, он приведен ниже.

1. ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДЛЯ РОБОТОТЕХНИКИ

В настоящее время не существует какого-либо общепринятого стандарта, какой-либо одной платформы для создания программного обеспечения для роботов. Однако существует несколько решений, которые широко используются в академических кругах и “робототехническом сообществе” и могут лечь в основу такого стандарта. Далее рассмотрены некоторые из них, сознательно выбраны только универсальные кросс-платформенные решения с открытым исходным кодом.

1.1 URBI

Urbi [4,5] представляет собой полнофункциональную кросс-платформенную (Windows, Linux, Mac OS) среду с открытым исходным кодом (лицензия GNU AGPL v3) для организации взаимодействия компонентов в сложных системах. Urbi изначально разрабатывалась для

робототехники: она обеспечивает все необходимые функции для координации работы различных независимых параллельных компонентов (приводы, датчики, программное обеспечение устройств, которые обеспечивают такие функции, как преобразование текста в речь, распознавание лиц и так далее).

Архитектура Urbi представлена на рисунке 1. Среда Urbi позволяет использовать компоненты, написанные на языке C++, для взаимодействия с оборудованием там, где нужна эффективность и доступ к низкоуровневым функциям. Компоненты должны использовать API библиотеки UObject C++ для интеграции в Urbi (для Java и Matlab также существует свой API). Среда также предоставляет инфраструктуру промежуточного слоя, который обеспечивает одновременное выполнение компонентов, синхронные и асинхронные запросы к ним и так далее. Она также имеет urbiscript - язык сценариев для описания поведения высокого уровня, похожий на Python или LUA, но со встроенной семантикой для описания параллельных и событийно-управляемых программ. Компоненты с интерфейсом UObject естественно поддерживаются языком программирования urbiscript. Можно взаимодействовать с этими компонентами посылая запросы, изменяя их на лету, наблюдая их состояние, подписываясь на события различного типа и так далее, что ускоряет процесс разработки.

Среда Urbi хорошо документирована, имеет широкое сообщество пользователей. Для Urbi доступно большое количество уже готовых драйверов оборудования и алгоритмов управления, также доступны платные моделирующая среда и визуальная интегрированная среда разработки. Последняя версия среды имеет возможность интеграции с ROS.

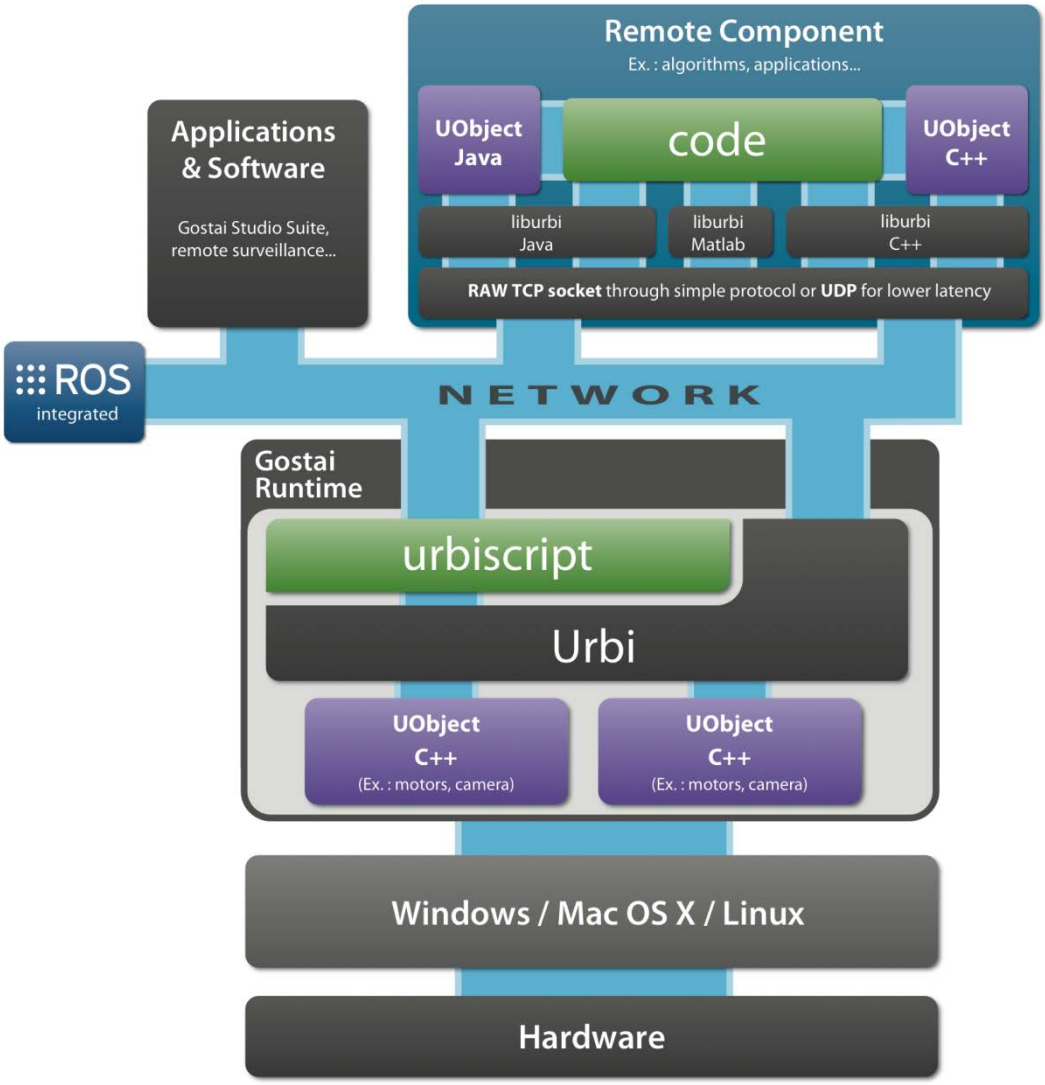

Рисунок 1. Архитектура Urbi

1.2 Player

Player [6, 7] - кросс-платформенное (Linux, Solaris, BSD, Mac OSX) программное обеспечение для исследования робототехнических систем с открытым исходным кодом (лицензия GNU GPL). Оно состоит из трех частей. Собственно Player – сервер, который обеспечивает сетевой интерфейс для взаимодействия между модулями-драйверами оборудования и управляющими программами. Модель клиент/сервер позволяет писать модули на любом языке программирования (клиентские библиотеки есть для C, C++, Python, Ruby, Java) и запускать на любом компьютере с сетевым

подключением к оборудованию. Stage – двумерный симулятор мобильных роботов. Gazebo – симулятор роботов / тренажер в трехмерном пространстве. Он включает в себя графическую подсистему и подсистему моделирования взаимодействия твердых тел и позволяет имитировать кинематику и динамику исполнительных механизмов роботов, в том числе при взаимодействии с объектами внешней среды, и формировать физически правдоподобные показания виртуальных датчиков среды (камеры, дальномеры, силомоментные датчики).

Сервер Player взаимодействует с конкретными аппаратными устройствами с помощью драйверов, а своим клиентам предоставляет стандартный абстрактный интерфейс устройств. Это позволяет управляющим программам прозрачно взаимодействовать с новым оборудованием. Новые драйверы могут быть добавлены любым разработчиком. Драйверы хранятся в виде подключаемых модулей, порядок их загрузки и параметры описываются в конфигурационном файле. Симулятор Gazebo также позволяет взаимодействовать с виртуальным оборудованием через стандартный интерфейс Player. То есть управляющая программа, написанная для Gazebo и там отлаженная, может быть без изменений перенесена на реальное оборудование и наоборот.

Проект Player/Stage/Gazebo имеет обширное сообщество пользователей и хорошую документацию. Как утверждают разработчики, Player является самым широко используемым программным обеспечением в академическом сообществе для робототехнических исследований. На текущий момент все компоненты Player/Stage/Gazebo интегрированы в ROS.

1.3 OROCOS

OROCOS [8,9] (Open Robot Control Software) — кросс-платформенное (Linux/RTLinux, Windows, Mac OS) свободное программное обеспечение (лицензия GNU LGPL) для управления робототехническими системами. OROCOS содержит несколько независимых частей. В том числе Orocos

toolchain - основной инструментарий для создания распределенных робототехнических приложений реального времени. С его помощью можно создавать модули на C++, которые взаимодействуют с остальной системой через предоставляемый Real-Time Toolkit API. Библиотека Orocos Component Model, входящая в Orocos toolchain обеспечивает:

- Безблокировочную, потокобезопасную, межкомпонентную связь в рамках одного процесса.
- Потокобезопасное межпроцессное взаимодействие (в том числе между распределенными процессами).
- Связь между компонентами, работающими в жестком реальном времени и не в реальном времени.
- Детерминированное время выполнения в процессе обмена для потоков с более высоким приоритетом.
- Синхронные и асинхронные связи между компонентами.
- Настройку модулей во время выполнения.

Отличительная особенность системы OROCOS в том, что она специально разработана для создания приложений реального времени. Система OROCOS также интегрирована в ROS [4].

1.4 MCA2

MCA2 (Modular Controller Architecture 2.0) [10,11] – кроссплатформенное (Linux/RTLinux, Win32, Mac OS/X) свободное (лицензия GNU GPL) программное обеспечение для управления роботами. MCA2 представляет собой каркас для создания программного обеспечения на основе модулей со стандартными интерфейсами. Он предоставляет клиентскую библиотеку для создания новых модулей и инфраструктуру, обеспечивающую их работу, синхронизацию и взаимодействие в процессе выполнения, в том числе в режиме реального времени. MCA2 не содержит инструментов автоматической генерации кода и визуальных средств программирования.

В MCA2 все модули реализованы на C++ и предоставляют стандартный интерфейс для взаимодействия. Данные между модулями передаются в виде массива значений с плавающей точкой, каждый модуль должен знать, как их интерпретировать. Модули могут быть включены/отключены/заменены прямо во время работы программы без необходимости перекомпиляции. Они могут объединяться в группы, которые действуют так же, как и любой другой модуль, но с более сложным поведением.

Изначально архитектура MCA разрабатывалась специально для автономных роботов серии ARMAR [12], созданных в лаборатории Гуманоидных и Интеллектуальных систем Института Технологий Карлсруэ (Humanoids and Intelligence Systems Laboratory of Karlsruhe Institute for Technology). Рассмотрим для примера архитектуру программного обеспечения робота-гуманоида ARMAR-III, которое построено с использованием MCA2. Оно состоит из трех слоев (см. рисунок 2). На низшем уровне расположены сигнальные микропроцессоры исполнительной системы управления и другие аппаратные средства, такие как микрофоны, громкоговорители, камеры и так далее. Они связаны с компьютерами среднего уровня либо непосредственно, либо через CAN-шину. На среднем уровне осуществляется управление более высокого уровня: расчет прямой и инверсной кинематики, планирование безопасных траекторий, обработка речи и так далее. Состав модулей на первых двух уровнях остается неизменным. Программирование робота осуществляется только на верхнем уровне. Здесь программный интерфейс робота обеспечивает удобный доступ к датчикам и исполнительным устройствам с помощью C++ переменных и методов. В дополнение к возможности прямого доступа к датчикам и приводам, определены два уровня абстракции: задачи и навыки. Навыки реализуют атомарные возможности, такие как навигация, визуальный поиск объекта, захват/перемещение/установка объекта, открытие/закрытие дверей и

так далее. Задачи работают на более высоком уровне и состоят из нескольких навыков, например, найти и принести чашку со стола.

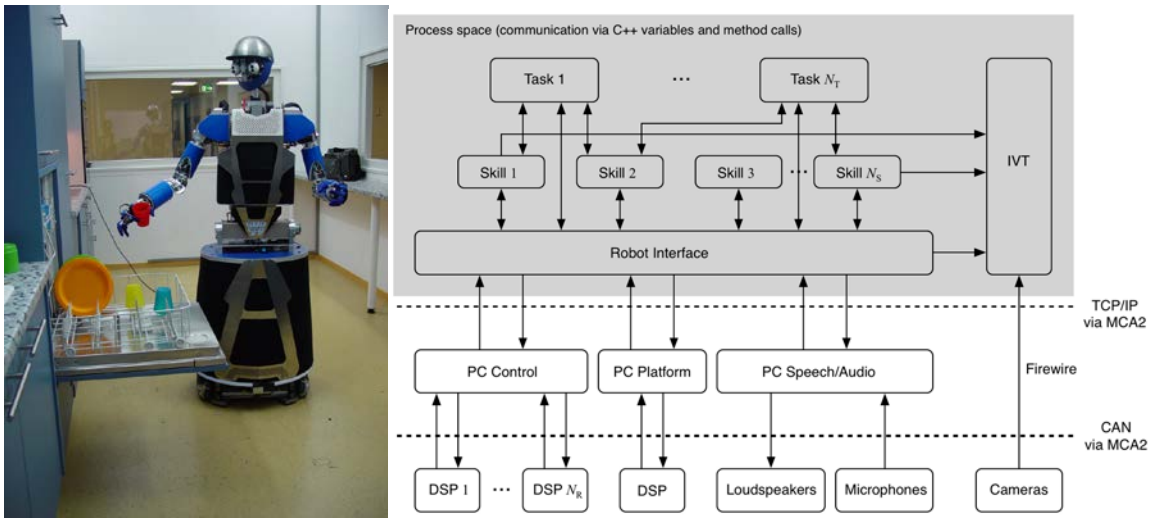

Рисунок 2. Гуманоидный робот ARMAR III и архитектура его системы управления

1.5 ROS

ROS (Robot Operating System) [13,14] – мета-операционная система для роботов с открытым исходным кодом (лицензия BSD). Она не заменяет операционную систему, она расширяет ее набором модулей, необходимых для управления РТС. Изначально ROS разработана для UNIX-совместимых операционных систем, но ведутся работы по ее переводу и на Windows. Основной целью ROS является поддержка повторного использования кода в робототехнических исследованиях и разработках.

ROS состоит из двух частей: ядро ROS – минимально необходимый для работы набор инструментов, модулей и библиотек; и набор развиваемых пользователями пакетов, которые реализуют различные функции робототехники: работа с оборудованием, планирование траекторий, обработка информации, моделирование и многое другое.

Ядро ROS разработано так, чтобы как можно меньше зависеть от архитектуры конечной системы. Граф вычислений ROS представляет собой одноранговую сеть процессов (узлов в терминологии ROS), которые обрабатывают данные совместно. Узлы могут обмениваться данными несколькими способами: через темы, сервисы или сервер параметров. Структура данных и логика обмена может быть любой. При старте узлы сообщают свои регистрационные данные мастер-узлу, который хранит регистрационную информацию. От него узлы также могут получить информацию о других зарегистрированных узлах и наладить с ними связь напрямую через согласованный протокол обмена (наиболее распространен в ROS протокол TCPROS, в основе которого лежит TCP/IP). Мастер также информирует узлы об изменениях в системе, что позволяет динамически создавать соединения при старте новых узлов. Узлы могут быть распределены по разным компьютерам, а топология системы может быть легко изменена в процессе работы без необходимости перекомпиляции. В отличие от других решений, код ROS максимально тонкий [15], он не накладывает ограничений на тип приложений, содержащих узлы или их структуру и цикл работы. ROS не зависит от языка программирования, клиентские библиотеки уже реализованы для Python, C++, Lisp, существуют экспериментальные библиотеки для Java и Lua. Такой подход позволяет ROS легко интегрироваться с различными архитектурами и строить системы различной сложности поверх нее. ROS также легко интегрируется с другим программным обеспечением для роботов, например, Urbi, Player, OROCOS и позволяет использовать лучшие наработки из всех этих систем.

На уровне файловой системы основным блоком для организации программного обеспечения в ROS является пакет. Пакет ROS может содержать исходные коды и исполняемые файлы узлов, библиотеки, описание сообщений и сервисов, базы данных, файлы конфигурации, и другие ресурсы, которые логично организовать вместе. Каждый пакет должен содержать файл манифеста, который предоставляет метаданные о

пакете, включая сведения о лицензии и зависимостях, а также флаги компилятора и так далее. ROS предоставляет инструменты для управления пакетами: загрузка/создание, сборка, установка, настройка, навигация. Пакеты, предоставляющие совместную функциональность, могут объединяться в стеки, например, «стек навигации».

На уровне сообщества ROS предоставляет ресурсы, которые позволяют разным организациям или разработчикам обмениваться программным обеспечением и знаниями. Эти ресурсы включают в себя дистрибутивы – коллекции версий стеков для совместной установки (аналог дистрибутивов Linux); сеть федеративных хранилищ кода, где пользователи могут размещать свои собственные компоненты программного обеспечения робота; ROS Wiki – основное хранилище документации ROS, где любой член сообщества может делиться своей документацией; форум; система трекинга ошибок.

2. ПРИМЕНЕНИЕ ROS ДЛЯ ПМК ДФ МГТУ

После анализа текущего состояния робототехнического программного обеспечения было принято решение выбрать ROS в качестве основы для построения ПМК Дмитровского филиала МГТУ им. Баумана. По следующим причинам: удобство разработки и использования, универсальность, модульность, гибкость, возможность легкого масштабирования, интеграция с другими системами робототехнического программного обеспечения, открытый программный код, большое количество готовых драйверов, алгоритмов, решений для стандартных задач робототехники, возможность независимо разрабатывать и добавлять в систему собственные модули, легкость отладки, большое активное сообщество пользователей. Также немаловажным является то, что разработка под ROS может вестись на разных языках программирования и в любой среде разработки, например, бесплатной Eclipse. Для ROS доступны мощные инструменты отладки, сбора информации в процессе работы системы и ее последующего анализа.

Для запуска системы управления ПМК под ROS были разработаны отсутствующие в ROS драйверы для промышленных роботов Kawasaki, силомоментных датчиков и захватного устройства Schunk; адаптированные для системы симуляции Gazebo модели оборудования; пакеты навигации для ПР Kawasaki в «одноруком» и «двуруком» исполнении; управляющее программное обеспечение, например, различные версии управляющих программ для податливого движения двурукого робота. На рисунке 3 представлен внешний вид комплекса с трехмерным графическим интерфейсом rviz. На рисунке 4 для примера приведена вычислительная сеть ROS при управлении одним из роботов, которая включает узлы осуществляющие взаимодействие с пользователем, планирование и фильтрацию траекторий с учетом объектов рабочей среды, систему безопасности, обмен командами и информацией с промышленным контроллером манипулятора. Переход на исследование двурукого робота может быть осуществлен буквально одной командой, без необходимости длительной настройки и тем более перекомпиляции.

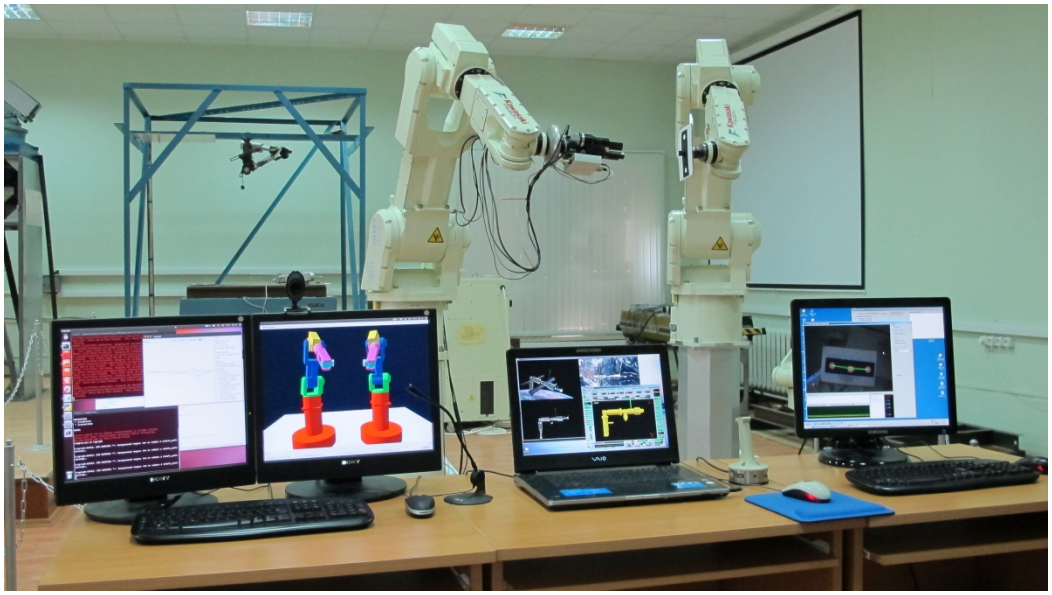

Рисунок 3. Внешний вид комплекса под управлением ROS

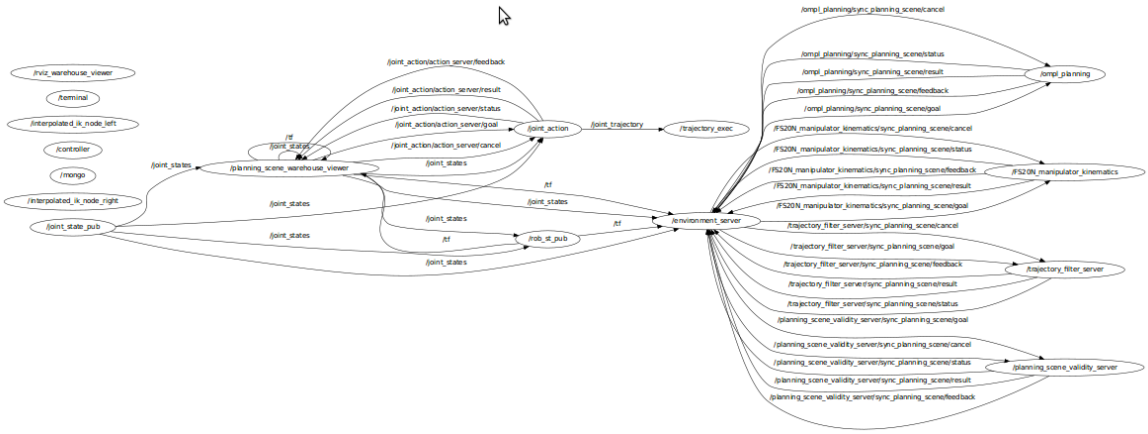

Рисунок 4. Вычислительная сеть ROS одной из конфигураций ПМК ДФ МГТУ им. Баумана

ЗАКЛЮЧЕНИЕ

Разработанное программное обеспечение в совокупности с уже доступными в ROS пакетами позволит решать с помощью ПМК новые задачи с минимальными затратами, концентрируясь лишь на разработке новой функциональности и использовать в работе последние мировые достижения в области робототехники. Кроме того, применение широко используемого программного обеспечения позволяет говорить с заказчиками «на одном языке», а благодаря гибкости и архитектуре ROS модули, отлаженные на ПМК, можно переносить на конечную систему практически без изменений.

Список литературы

1. Illarionov V.V., Korshunov S.V., Leskov A.G., Leskova S.M., Shumov A.V., Zimin A.M. Using Integrated assembly of Virtual and Real Robot System with Remote Access for Pracical Training. // Innovations 2009: World Innovations in Engineering Education and Research / Editors W. Aung et al. - INEER, Arlington, VA 22205, USA, 2009. – pp. 99 - 108.

2. Лесков А.Г., Илларионов В.В. Математическое и полунатурное моделирование операций космических манипуляционных роботов. – Тезисы докладов 8-й Международной научно-практической конференции «Пилотируемые полёты в космос». – Звёздный городок, 2009. с.70-71.

3. A. Shakhimardanov, J. Paulus, N. Hochgeschwender, M. Reckhaus, G. Kraetzschmar, “Best Practice Assessment of Software Technologies for Robotics”, Bonn-Rhein-Sieg University, 2010

4. Jean-Christophe Baillie, Akim Demaille, Quentin Hocquet, Matthieu Nottale, Samuel Tardieu, “The Urbi Universal Platform for Robotics”. Workshop Proceedings of SIMPAR 2008 Intl. Conf. on SIMULATION, MODELING and PROGRAMMING for AUTONOMOUS ROBOTS, Venice(Italy) 2008 November, 3-4.

5. «The Urbi Software Development Kit», 2011, [Онлайн]. Ссылка: http://www.gostai.com/downloads/urbi/2.x/doc/urbi-sdk.pdf

6. B. Gerkey, R. Vaughan, and A. Howard, “The player/stage project: Tools for multi-robot and distributed sensor systems,” In Proc. of the International Conference on Advanced Robotics, 2003.

7. Toby H.J. Collett, Bruce A. MacDonald, and Brian P. Gerkey, "Player 2.0: Toward a Practical Robot Programming Framework". In Proceedings of the Australasian Conference on Robotics and Automation (ACRA 2005), Sydney, Australia, December 2005.

8. H. Bruynickx, «Open robot control software: the orocos project». In IEEE International Conference on Robotics and Automation", 2001, 2523–2528.

9. H. Bruynickx, P. Soetens, «The OROCOS Project», 2007, [Онлайн]. Ссылка: http://www.orocos.org/orocos/whatis

10. Ly D., Regenstein K., Asfour T., Dillmann, R. «A Modular and Distributed Embedded Control Architecture for Humanoid Robots». IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2004.

11. K.-U. Scholl. «Introduction to Modular Controller Architecture (MCA)», 2005, [Онлайн]. Ссылка: http://www.mca2.org/introduction.html

12. T. Asfour, K. Regenstein, P. Azad, J. Schroder, A. Bierbaum, N. Vahrenkamp, R. Dillmann. «ARMAR-III: An Integrated Humanoid Platform for Sensory-Motor Control». IEEE/RAS International Conference on Humanoid Robots (Humanoids), 4-6 Dec. 2006, pp. 169-175.

13. M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J.Leibs, E. Berger, R. Wheeler, “ROS: an open-source Robot Operating System”. ICRA Workshop on Open Source Software, 2009.

14. «Robot Operating System / Introduction», 2012, [Онлайн]. Ссылка: http://www.ros.org/wiki/ROS/Introduction.

15. A. Makarenko, A. Brooks, and T. Kaupp, “On the benefits of making robotic software frameworks thin,” in IEEE International Conference on Intelligent Robots and Systems. Workshop on Evaluation of Middleware and Architectures., 2007.