16. *Fan Q., Zhuo W., Tang C.K., Tai C.L.* Few-Shot Object Detection with Attention-RPN and Multi-Relation Detector, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 4013-4022.
17. OpenMMLab. MMFewShot: A Toolbox for Few Shot Learning Detection and Classification. Available at: https://mmfewshot.readthedocs.io/en/latest/ (accessed 14 January 2026).
18. *Girshick R.* Fast R-CNN, *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1440–1448.
19. *He K., Zhang X., Ren S., Sun J.* Deep Residual Learning for Image Recognition, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770-778. doi: 10.1109/CVPR.2016.90.
20. *Robbins H.E.* A Stochastic Approximation Method, *Annals of Mathematical Statistics*, 1951, Vol. 22, pp. 400-407.

Бабин Степан Евгеньевич – АО «Научно-производственное предприятие «Авиационная и морская электроника»; e-mail: babinstepan12@yandex.ru; г. Санкт-Петербург, Россия; тел.: 88123274667; тестировщик программного обеспечения.

Гессен Павел Алексеевич – АО «Научно-производственное предприятие «Авиационная и морская электроника»; e-mail: gessen_pa@nppame.ru; г. Санкт-Петербург, Россия; тел.: 88123274667; зам. начальника центра средств интеллектуальной обработки изображений НПК РТС СН.

Павлова Валерия Анатольевна – АО «Научно-производственное предприятие «Авиационная и морская электроника»; e-mail: pavlova@nppame.ru; г. Санкт-Петербург, Россия; тел.: 88123274667; зам. директора НПК РТС СН по НИОКР.

Тупиков Владимир Алексеевич – АО «Научно-производственное предприятие «Авиационная и морская электроника»; e-mail: tupikov@nppame.ru; г. Санкт-Петербург, Россия; тел.: 88123274667; зам. ген. директора; директор НПК РТС СН.

Babin Stepan Evgenievich – Scientific and Production Enterprise Aviation and Marine Electronics, JSC; e-mail: babinstepan12@yandex.ru; St. Petersburg, Russia; phone: +78123274667; software tester.

Gessen Pavel Alekseevich – Scientific and Production Enterprise Aviation and Marine Electronics, JSC; e-mail: gessen_pa@nppame.ru; St. Petersburg, Russia; phone: +78123274667; deputy head of the Intelligent Image Processing Center of the Scientific and Production Corporation RTS SN.

Pavlova Valeria Anatolyevna – Scientific and Production Enterprise Aviation and Marine Electronics, JSC; e-mail: pavlova@nppame.ru; St. Petersburg, Russia; phone: +78123274667; deputy director for R & D, Scientific and Production Complex of Special-Purpose Robotic Systems.

Tupikov Vladimir Alekseevich – Scientific and Production Enterprise Aviation and Marine Electronics, JSC; e-mail: tupikov@nppame.ru; St. Petersburg, Russia; phone: +78123274667; deputy general director, director of the Scientific and Production Complex of Special-Purpose Robotic Systems.

УДК 004.428 DOI 10.18522/2311-3103-2026-2-258-271

Н.А. Бочаров, К.А. Суминов, М.А. Кирилюк, Н.Б. Парамонов

ИССЛЕДОВАНИЕ ПОДХОДОВ К УНИФИКАЦИИ СВЯЗУЮЩЕГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ РОБОТОТЕХНИЧЕСКИХ КОМПЛЕКСОВ ДЛЯ АППАРАТНО-ПРОГРАММНОЙ ПЛАТФОРМЫ «ЭЛЬБРУС»

Импортозамещение аппаратно-программных систем управления является одним из ключевых факторов развития современной отечественной робототехники. Ограниченность программной экосистемы российских вычислительных платформ затрудняет их применение в составе бортовых вычислительных систем (БВС) робототехнических комплексов (РТК), несмотря на появление современных отечественных многоядерных процессоров, например, семейства «Эльбрус», обеспечивающих конкурентоспособные характеристики по производительности и энергоэффективности. ROS (Robot Operating System) и его развитие — ROS 2 — широко используются для разработки программного обеспечения РТК, однако их официальная поддержка ограничена архитек-

турами x86 и ARM под управлением распространённых Linux-дистрибутивов. Сохранение значительного объёма прикладного программного обеспечения на ROS, несмотря на прекращение его официальной поддержки, а также активное развитие ROS 2 обусловливают потребность в обеих версиях, функционирующих на отечественной аппаратно-программной платформе (АПП). Целью работы является исследование подходов к унификации связующего программного обеспечения робототехнических комплексов на базе ROS и ROS 2 для АПП «Эльбрус». Авторами выполнено портирование с использованием бинарной трансляции на уровне приложений и полностью нативной сборки, включающей адаптацию архитектурных зависимостей, модификацию архитектурно-зависимого кода и формирование deb-пакетов. Проведены экспериментальные исследования в различных конфигурациях запуска, включая совместную работу ROS и ROS 2. Результаты тестирования подтверждают возможность интеграции АПП «Эльбрус» в экосистему современной робототехники и демонстрируют преимущество нативной реализации по показателям производительности по сравнению с использованием бинарного транслятора. Практическая значимость работы заключается в расширении инструментальной базы разработки РТК на отечественной АПП и снижении технологической зависимости при создании систем управления РТК.

ROS; ROS 2; «Эльбрус»; портирование программного обеспечения; бинарная трансляция; нативная сборка; робототехнические комплексы.

N.A. Bocharov, K.A. Suminov, M.A. Kirilyuk, N.B. Paramonov

APPROACHES TO MIDDLEWARE UNIFICATION FOR ROBOTIC SYSTEMS ON THE ELBRUS HARDWARE–SOFTWARE PLATFORM

Import substitution of hardware–software control systems is a key enabler for the development of Russia's domestic robotics industry. The limited software ecosystem of Russian computing platforms still constrains their use in onboard computing systems (OCS) of robotic systems (RS), even though modern domestic multicore processors—such as the Elbrus—offer competitive performance and energy efficiency. ROS (Robot Operating System) and its successor ROS 2 are widely adopted for robotics software development, but official support is largely restricted to x86 and ARM architectures running mainstream Linux distributions. At the same time, a substantial amount of deployed application software remains ROS-based, while ROS 2 continues to evolve rapidly. This creates a practical need to support both ROS and ROS 2 on domestic hardware–software platforms. This work investigates approaches to unifying robotics middleware based on ROS and ROS 2 for the Elbrus platform. We implemented two porting strategies: (1) application-level binary translation and (2) a fully native build. The native approach included adapting architecture-specific dependencies, modifying architecture-dependent code, and producing Debian (deb) packages. We then evaluated both approaches under multiple runtime configurations, including scenarios where ROS and ROS 2 operate together. Experimental results confirm that the Elbrus platform can be integrated into the modern robotics software ecosystem. They also show that the native implementation delivers higher performance than the binary-translated solution. The proposed work expands the toolchain available for robotics development on domestic platforms and reduces technological dependence when building control systems for robotic systems.

ROS; ROS 2; Elbrus; software porting; binary translation; native compilation; robotic systems.

Введение. В современной отечественной робототехнике одним из ключевых направлений является импортозамещение вычислительных систем и программного обеспечения (ПО). Появление новых отечественных многоядерных процессоров, таких как «Эльбрус-2С3» [3], «Эльбрус-8С2» [4] открывает новые возможности в части аппаратного обеспечения робототехнических комплексов. Однако программная среда ОС «Эльбрус» [5] пока содержит лишь ограниченный набор инструментов для разработки систем управления роботами, что затрудняет применение платформы в робототехнике. Таким образом оснащение систем управления РТК вычислительной техникой на базе российских микропроцессоров [1] и разработка отечественного ПО остается актуальной задачей [2].

На текущий момент в мире нет общепринятых стандартов программного обеспечения бортовых информационно-вычислительных систем РТК ВН или же для групп таких роботов [6]. Существующие системы существенно различаются, и при работе с каждой из них возникают специфические трудности. Вместе с тем уже появились программные

платформы, предназначенные для стандартизации подходов к разработке робототехнических систем, а также для предоставления единой среды управления и контроля. В табл. 1 приведено сравнение программных платформ для разработки ПО для РТК.

Таблица 1

Сравнение программных платформ для разработки ПО для РТК

Фреймворк	ОС	Протоколы коммуникации	Поддержка реального времени	Год обновления	Лицензия
1	2	3	4	5	6
ROS	Ubuntu Linux	TCPROS/ UDPROS	нет	2025	BSD
ROS2	Linux Windows macOS и др.	DDS	да	2025	Apache 2.0
MSR-DS	Windows 7 и выше	DSS/CCR	нет	2012	MS-PL
ERSP	Windows XP и выше	нет данных	нет	2004	Коммерческая
Open-RTM	Linux Windows macOS Raspberry PiOS	CORBA TCP UDP	да	2024	LGPL
OROCOS	Linux Windows RTOS	CORBA TCP UDP CAN RS232	да	2016	LGPL
OPRoS	Linux Windows RTOS	TCP UDP CAN RS232	да	2013	LGPL

Наиболее распространённой из них является ROS (Robot Operating System). В мировом сообществе ROS широко используется для разработки ПО робототехнических систем [7, 8]. ROS – свободно распространяемый фреймворк, обеспечивающий абстракцию от аппаратных средств, функционирование в парадигме «издатель-подписчик» и межпроцессное взаимодействие через топики и сообщения. В 2025 году официальная поддержка ROS 1 была прекращена [9], и дальнейшее развитие экосистемы осуществляется исключительно в рамках ROS 2. Несмотря на то что переход от ROS 1 к ROS 2 продолжается уже достаточно долго, в настоящее время сохраняется значительный объём программного обеспечения, разработанного для ROS 1, перенос которого на ROS 2 требует существенных ресурсов. Это обуславливает сохраняющуюся потребность в использовании как ROS 1, так и ROS 2.

Официальная поддержка ROS 1 и ROS 2 ограничена Linux-дистрибутивами, в первую очередь Ubuntu и Debian, под архитектуры x86, ARM. Это означает, что для использования ROS 1, ROS 2 на отечественной платформе «Эльбрус» необходимо провести их портирование.

Унификация связующего программного обеспечения на базе Robot Operation System. ROS изначально был разработан в лаборатории искусственного интеллекта стэндфордского университета, а затем продолжил свое развитие при поддержке научно-исследовательского института робототехники [10]. ROS представляет собой надстройку над операционной системой, а также набор различных библиотек: OpenCV [11] (алгоритмы компьютерного зрения и обработки изображений), PCL [12] (облака 3D-точек), Ogre (объектно-ориентированный графический движок), Orocos (алгоритмы управления движением) и многие другие [13]. К основным особенностям ROS относятся: организа-

ция одноранговой сети между узлами, базирование программного обеспечения на уже существующих решениях в виде популярных библиотек, поддержка различных языков программирования, открытый исходный код. Каждое устройство в ROS может обратиться к любому другому узлу, а ведущее устройство используется только для маршрутизации и обеспечивает механизм поиска. В основе взаимодействия между устройствами под управлением ROS лежит пересылка сообщений.

Основным принципом обмена данными между программными компонентами является модель взаимодействия «издатель-подписчик» (publisher-subscriber) [14]. Это асинхронный механизм, при котором один узел (издатель) публикует сообщения на определенный топик, а другие узлы (подписчики) подписываются на этот топик и получают данные.

Этот подход позволяет создавать гибкие и масштабируемые системы, где каждый узел может работать независимо от других, получая или отправляя данные по мере необходимости. Благодаря использованию такого принципа множество сенсоров и исполнительных механизмов могут взаимодействовать друг с другом без жёсткой синхронизации.

Топики в данном случае служат каналами для передачи сообщений между узлами. Каждый топик связан с определенным типом сообщений, и все узлы, которые публикуют или подписываются на этот топик, обмениваются данными этого типа. Взаимодействие через топики позволяет создавать системы, где множество узлов может обмениваться данными без прямого подключения и работать независимо друг от друга, обеспечивая параллельное выполнение различных задач и обеспечивать функционирование в БВС на базе разнородных вычислительных комплексов различной аппаратной архитектуры.

Схема обработки сообщений в ROS не зависит от языка программирования, таким образом одновременно могут работать модули, написанные на различных языках программирования. ROS базируется на наборе автономных библиотек, что позволяет использовать готовые проверенные решения для своего робота. Клиент-серверная архитектура ROS представлена на рис. 1.

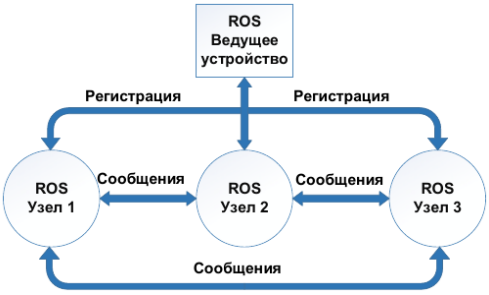

Рис. 1. Структура системы ROS

Развитием ROS является ROS 2, это следующее поколение, которое было анонсировано на ROSCon 2014 и выпущено в декабре 2017 года [15]. Проведенный анализ показал, что ROS 2 имеет ряд отличий от ROS 1 [16, 17], которые значительно влияют на процесс портирования [2] на другую аппаратно-программную платформу. Среди таких особенностей стоит отметить следующие:

1. ROS 2 использует новые библиотеки и технологии поддержки реального времени и встраиваемых систем.

2. В ROS 2 узлы не имеют глобальных имен и могут быть частью группы (namespace), а топики в ROS 2 имеют типы и качество обслуживания (QoS), которые определяют, как сообщения будут доставляться и храниться.

3. ROS 2 использует различные реализации DDS (Data Distribution Service) [18] в качестве промежуточного слоя, такие как Fast-RTPS, CycloneDDS и RTI Connext, для обеспечения низкоуровневой коммуникации между узлами.

4. В отличие от ROS, ROS 2 имеет меньше зависимостей от сторонних библиотек и инструментов, таких как Boost, и использует более современные стандарты, такие как Си++14 и Python 3.

5. ROS 2 требует наличия Python 3 и CMake для установки и сборки.

6. ROS 2 использует новый формат пакетов ament, который основан на CMake и позволяет более гибко настраивать процесс сборки и тестирования.

7. ROS 2 также использует новый формат сообщений, называемый IDL, который позволяет более легко интегрировать различные языки программирования и сериализаторы.

8. ROS 2 также имеет новую концепцию пространств имен (namespace) и групп (group), которые позволяют организовывать узлы в логические единицы.

9. ROS 2 имеет новые инструменты для работы с роботами, такие как ROS 2cli, rclcpp, rclpy и другие.

10. ROS 2 также поддерживает совместимость с ROS 1 через специальный мост, который позволяет обмениваться сообщениями между двумя версиями фреймворка.

Пример архитектуры взаимодействия компонентов ROS 2 представлен на рис. 2.

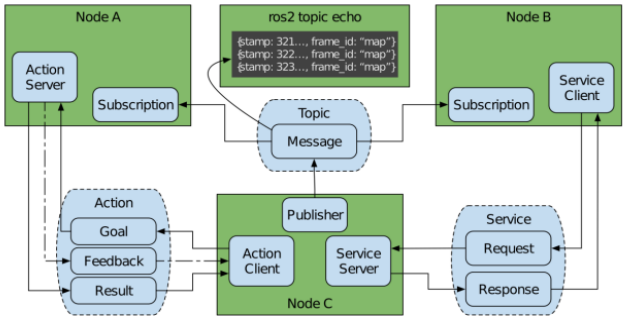

Рис. 2. Пример архитектуры взаимодействия компонентов ROS 2

Проведённый анализ показал, что актуальные дистрибутивы ОС для АПП «Эльбрус» не содержат значительную часть зависимостей фреймворков ROS и ROS 2, в том числе таких, как Gazebo, Ignition, Ogre, EmPY, PCL. Таким образом, для создания связующего программного обеспечения РТК данные пакеты необходимо портировать или разработать их функциональные аналоги для АПП «Эльбрус». С учетом проведенного анализа зависимостей по пакетам и версиям ядра для реализации связующего программного обеспечения РТК для платформы «Эльбрус» были определены следующие версии дистрибутивов: ROS 2 Galactic Geochelone, ROS 1 Noeitc Ninjemys, которые адаптированы под дистрибутивы Debian и Ubuntu, функционирующие на ядре Linux 5.10.

Особенности АПП «Эльбрус» при реализации ROS. Для адаптации открытых исходных кодов ROS 1, ROS 2 были выстроены деревья топологических зависимостей для всех пакетов версий base, а также некоторых дополнительных пакетов, применение которых распространено в области наземной робототехники, таких как ros-bridge для осуществления взаимодействия ROS 1 и ROS 2, систем логирования и пр. В рамках портирования ROS Noetic Ninjemys на АПП «Эльбрус» потребовалась адаптация более 90 пакетов ROS, а также более 200 пакетов зависимостей. Для ROS 2 Galactic Geochelone потребовалась адаптация 197 пакетов ROS 2 и более 1200 пакетов зависимостей. Кроме того, при анализе исходных кодов этих пакетов было выявлено, что многие из них, несмотря на использование в дистрибутивах ROS 1 и ROS 2, являются нестабильными, о чём явно указано в документации к ним.

С учётом этих особенностей и большого количества пакетов были рассмотрены два подхода: сначала портирование с использованием бинарного транслятора, а затем полностью нативное портирование.

Программное обеспечение платформы «Эльбрус» включает два метода двоичной трансляции. Первый метод – это система полной двоичной трансляции. В ней транслятор размещается между микропроцессором и исполняемыми x86-кодами. Он переводит коды BIOS, ОС, драйверов и приложений. Вычислительный комплекс на базе процессора

«Эльбрус» с такой трансляцией полностью идентичен для пользовательской программы вычислительным комплексам на x86-процессорах. При втором подходе система двоичной трансляции выступает как обычное Linux приложение и функционирует под контролем ОС Linux. Она обеспечивает запуск x86-Linux приложений, которые могут выполняться параллельно с нативными приложениями платформы «Эльбрус».

Для реализации был выбран второй вариант использования. При этом работа с бинарным транслятором уровня приложений предполагает установку гостевой файловой системы на ВК «Эльбрус». Для функционирования ROS 2 на АПП «Эльбрус» в таком режиме в качестве гостевой системы можно использовать любую из официально поддерживаемых ROS, например Ubuntu 20.04 Focal Fossa. Схема работы такой системы показана на рис. 3 [19].

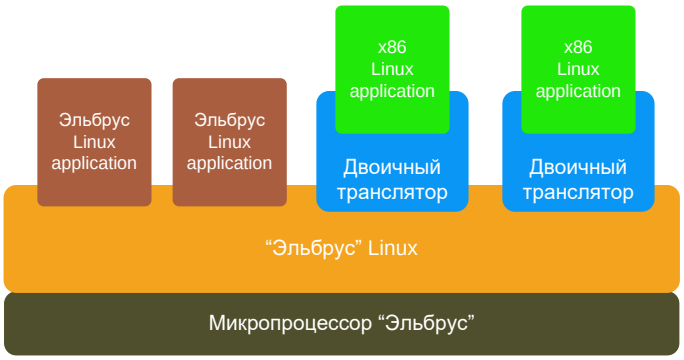

Рис. 3. Схема работы системы двоичной трансляции Linux приложений

При нативном портировании ROS 1 и ROS 2 использовались архитектурно-зависимые пакеты, поставляемые в комплекте с ОС «Эльбрус Линукс 8.0» [20]. Открытые исходные коды ROS 1 Noetic Ninjemys и ROS 2 Galactic Geochelone потребовали существенной доработки и адаптации для корректного исполнения на аппаратно-программной платформе «Эльбрус» с архитектурой команд не ниже версии v4 [21]. Необходимость внесения изменений обусловлена принципиальными архитектурными особенностями микропроцессоров семейства «Эльбрус».

Хотя ROS, ROS 2, поддерживает различные процессорные архитектуры (x86, ARM), эти архитектуры значительно отличаются от архитектуры «Эльбрус» рядом особенностей, таких как [22]:

1. Принцип широкого командного слова (VLIW), предполагающий одновременное исполнение множества операций обработки информации (до 23 операций для скалярных данных и до 33 операций для векторных форматов) за один такт.

2. Принципы явного параллельного исполнения команд (EPIC), предполагающие передачу функций распараллеливания вычислений оптимизирующему компилятору, позволяющему обеспечить максимальную производительность используемого микропроцессора.

3. Обеспечение в микропроцессоре многоуровневой системы параллелизма, включающей параллелизм операций, векторный параллелизм, параллелизм потоков управления на общей памяти, параллелизм задач.

Ключевая черта архитектуры «Эльбрус» – так называемое явное управление параллелизмом, когда компилятор, зная внутреннее устройство процессора, сам определяет распределение операций по ядрам микропроцессора и возможность их одновременного исполнения. В традиционных архитектурах типа RISC или CISC на вход процессора поступает поток инструкций, которые рассчитаны на последовательное исполнение. Процессор может в определенной мере детектировать независимые операции и запускать их параллельно и даже менять их порядок. Однако динамический анализ зависимостей и поддержка внеочередного исполнения имеет свои ограничения: лучшие современные процессоры способны анализировать и запускать на исполнение до 8 команд за такт.

В архитектуре «Эльбрус» основную работу по анализу зависимостей и оптимизации порядка операций берет на себя компилятор. Процессору на вход поступают широкие команды (ШК), в каждой из которых закодированы инструкции для всех ядер микропроцессора, которые должны быть запущены на данном такте. От процессора не требуется анализировать зависимости между операндами или переставлять операции между широкими командами: все это делает компилятор, исходя из анализа исходного кода и планирования ресурсов микропроцессора.

Компилятор способен анализировать исходный код гораздо тщательнее, чем аппаратура RISC/CISC процессора, и находить больше независимых операций. Поэтому в архитектуре «Эльбрус» больше параллельно работающих исполнительных устройств, чем в традиционных архитектурах.

Проблемы реализации ROS на АПП «Эльбрус». Ввиду наличия представленных выше архитектурных и программных особенностей исходные коды практически всех необходимых пакетов потребовали адаптации как на уровне программной реализации, так и на уровне систем сборки.

В пакетах с небольшим объемом кода на C/C++ изменения зачастую не требовались. Однако доля таких пакетов в общем объеме зависимостей оказалась незначительной. В исходных кодах подавляющего большинства пакетов потребовались модификации, обусловленные наличием ассемблерных вставок, ориентированных на архитектуры x86 или ARM и требовавших переписывания либо замены на архитектурно-независимые реализации, «жестко» закодированных констант, определяющих корректность работы на конкретной архитектуре или операционной системе, скрытых архитектурных предположений, оформленных в виде так называемых «магических чисел», использования встроенных компиляторных расширений (например, GCC built-ins), не полностью поддерживаемых компилятором платформы «Эльбрус».

Во многих случаях такие константы явно не документировались и не были вынесены в конфигурационные параметры. В результате исходные коды успешно компилировались, однако выполнение программ завершалось ошибками, чаще всего связанными с некорректным обращением к памяти. Это проявлялось в виде ошибок сегментации, повреждения кучи, некорректной работы многопоточных участков кода.

Среди регулярно встречающихся проблем можно выделить:

1. Фиксированные размеры буферов, приводящие к переполнению при изменении соглашений о выравнивании данных.

2. Жестко заданные шаги выравнивания структур в памяти (например, предположение о 16- или 32-байтном выравнивании, характерном для SSE/AVX).

3. Использование системных констант и флагов, специфичных для glibc определенной версии.

4. Предположения о размере типов данных (например, implicit приведение указателей к long, различия в Application Binary Interface).

В подобных ситуациях требовался детальный анализ исходного кода, отладка с использованием инструментов трассировки и профилирования, а также поэтапная замена архитектурно-зависимых участков на переносимые реализации.

Отдельные пакеты, такие как rclpy, PCL и CycloneDDS, потребовали значительных изменений ввиду сложной внутренней архитектуры, активного использования многопоточности, шаблонных метапрограммных конструкций, а также наличия большого количества архитектурных зависимостей. В частности, в PCL потребовалась корректировка архитектурно-оптимизированных участков, а в CycloneDDS — адаптация низкоуровневых механизмов сетевого взаимодействия и синхронизации.

Еще одной особенностью явилось то, что используемый дистрибутив ОС «Эльбрус Линукс 8.0», выбранный в качестве базовой платформы, уже содержит часть пакетов, необходимых для функционирования выбранных версий ROS и ROS 2. Однако в исходных кодах ROS и ROS 2 нередко используются версии библиотек, отличающиеся от поставляемых в составе дистрибутива. Это может быть обусловлено, например, исторической преемственностью, требованиями совместимости с конкретными релизами и т.д.

В результате возникали конфликты версий библиотек. Простая замена версии системного пакета на требуемую для ROS не всегда допустима, поскольку другие компоненты операционной системы зависят от актуальной версии библиотеки. Для разрешения подобных конфликтов применялся следующий подход:

- установка альтернативной версии библиотеки в изолированный каталог;
- корректировка путей поиска (RPATH, LD_LIBRARY_PATH);
- модификация CMake- и Make-скриптов для явного указания требуемой версии;
- при необходимости – сборка статических версий библиотек.

Аналогичные конфликты наблюдались между зависимостями ROS 1 и ROS 2, использующими различные версии одних и тех же библиотек. Проблема решалась разделением сред сборки и раздельной изоляцией зависимостей с последующей корректной настройкой workspace.

Дополнительной сложностью стала распространенная практика автоматической загрузки зависимостей из сети в процессе сборки. Некоторые пакеты при установке инициировали скачивание исходных кодов сторонних библиотек, патчей, тестовых данных или конфигураций. Часто такой механизм приводил к цепочкам рекурсивных загрузок, когда один скачанный пакет инициировал загрузку других. Поскольку целевая установка предполагается в оффлайн-среде, а также с учетом необходимости адаптации всех компонентов под архитектуру «Эльбрус», такой механизм оказался неприемлем.

Была проведена полная трассировка внешних обращений к сетевым ресурсам, после чего все требуемые внешние зависимости были загружены локально, и их исходные коды адаптированы под целевую архитектуру. При этом скрипты сборки были модифицированы для использования исключительно локальных ресурсов. Подобная ситуация, в частности, наблюдалась при работе с пакетами семейства iceoryx.

Дополнительных изменений потребовал пакет log4cxx, который в исходной конфигурации не обеспечивал корректный вывод отладочной информации при запуске ROS-узлов. Наблюдались проблемы с форматированием сообщений, уровнем логирования и выводом в консоль. Проблема была решена путем перекомпиляции библиотеки с корректными флагами, изменением значений по умолчанию для уровней логирования, а также корректировкой конфигурации вывода.

Существенный объем работ был связан с адаптацией скриптов сборки на базе make и CMake. Характерный объем таких скриптов для одного пакета составляет тысячи строк, содержащих многочисленные условные ветвления, проверки платформы и выбор конфигураций.

В среде «Эльбрус» используется собственный оптимизирующий компилятор, совместимый с GCC на уровне интерфейса, но имеющий особенности в наборе поддерживаемых флагов, модели оптимизации и генерации кода. Кроме того, различные аппаратные платформы (2C3, 8C2) требуют корректировки параметров компиляции и линковки.

В скриптах сборки нередко присутствовали проверки архитектуры вида if (CMAKE_SYSTEM_PROCESSOR STREQUAL "x86_64"), фиксированные наборы флагов оптимизации для SSE/AVX, архитектурно-зависимые пути к библиотекам, «магические числа», отражающие особенности изначально поддерживаемых платформ.

Также была проведена адаптация setup-скриптов среды ROS. Setup-скрипты предназначены для инициализации рабочего пространства (workspace). Скрипты обеспечивают настройку переменных окружения (PATH, LD_LIBRARY_PATH, PYTHONPATH и д.р.), регистрации путей к пакетам (ROS_PACKAGE_PATH), конфигурации среды сборки и выполнения в соответствии с особенностями выполненных реализаций и отличий от предусмотренных в исходных кодах.

Изменения потребовались как при сборке из исходных кодов, так и при установке из бинарных deb-пакетов, поскольку в ОС «Эльбрус» структура каталогов установки по умолчанию отличается от Debian/Ubuntu. Были скорректированы пути к библиотекам, Python-модулям и исполняемым файлам, а также обеспечена корректная работа workspace.

Для каждой аппаратной платформы были разработаны тестовые примеры и проверочные сценарии, позволяющие после установки выполнить контроль работоспособности среды. Тесты включали запуск базовых узлов и обмен сообщениями, проверку работы DDS-слоя, тестирование взаимодействия ROS 1 и ROS 2 через bridge, проверку многопоточной обработки данных и т.д.

С учетом вышеуказанных проблем и путей их решения был разработан экспериментальный образец связующего программного обеспечения на базе ROS 1 Noetic и ROS 2 Galactic для АПП «Эльбрус», функционирующий в двух режимах – нативного исполнения и бинарной трансляции.

Экспериментальные исследования разработанного связующего программного обеспечения на АПП «Эльбрус». Экспериментальные исследования разработанного экспериментального образца проводились с целью подтверждения корректности функционирования всех механизмов, оценки совместимости двух версий фреймворка при использовании ros1_bridge, проверки работы в гетерогенной вычислительной системе, сравнения производительности нативного исполнения и режима бинарной трансляции, а также для подтверждения состоятельности предложенного подхода по унификации связующего программного обеспечения РТК за счет использования двух режимов работы ROS.

Тестирование выполнялось на ВК, включающем два вычислительных модуля (ВМ) на основе «Эльбрус-2С3» и «Эльбрус-8СВ» под управлением ОС «Эльбрус Линукс 8.0», объединённых в сеть и имитирующих БВС РТК, структура которого представлена на рис. 5.

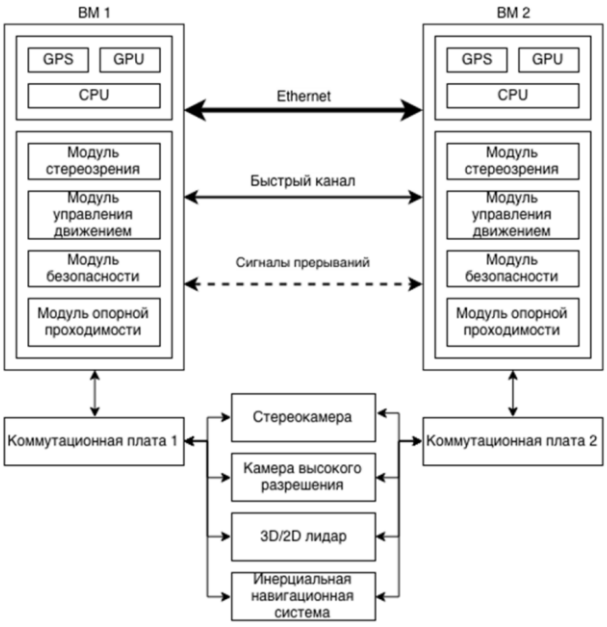

Рис. 5. Структура вычислительного комплекса

В ходе испытаний были реализованы и проанализированы несколько вариантов развёртывания программной среды, отличающихся режимами исполнения и распределением вычислительной нагрузки.

Первый вариант предусматривал запуск ROS 1 и ROS 2 в среде бинарной трансляции на одном вычислительном модуле «Эльбрус». В данной конфигурации вся программная экосистема функционировала как совокупность транслируемых x86-приложений в пределах единой операционной среды. Схема запуска представлена на рис. 6 слева.

Второй вариант представлял собой смешанный режим функционирования, при котором ROS 1 запускался в нативном режиме, ROS 2 – через бинарный транслятор. В данном случае для корректного функционирования потребовалась дополнительная настройка сети для ROS 2. Схема запуска представлена на рис. 6 справа.

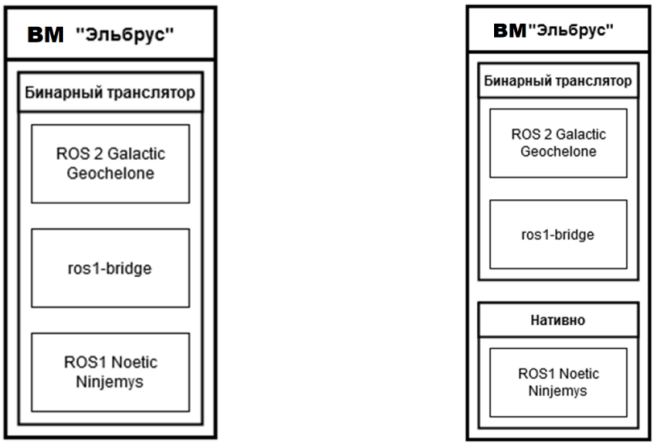

Рис. 6. Схема 1 запуска ROS 1, ROS 2 (слева), схема 2 запуска ROS 1, ROS 2 (справа)

Третий вариант включал в себя распределённую конфигурацию, в которой узлы ROS 1 и ROS 2 размещались на различных ВМ из состава ВК. Рассматривался вариант, когда оба вычислительных комплекса работали под бинарной трансляцией, а также сценарии, в котором один из них функционировал нативно. Схемы запуска представлены на рис. 7 сверху и рис. 7 снизу соответственно.

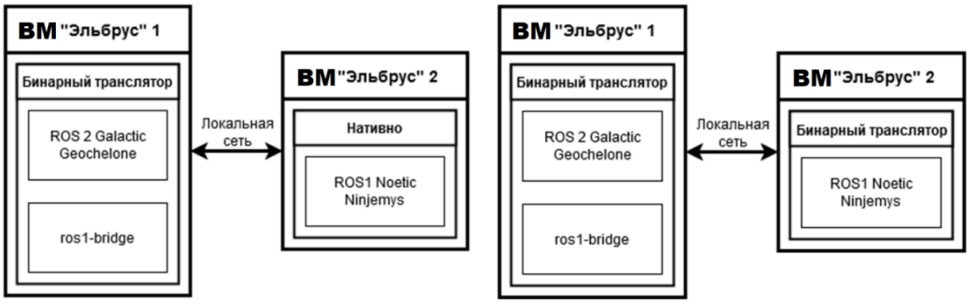

Рис. 7. Схемы 3 запуска ROS 1, ROS 2

Четвёртый вариант включал конфигурации с полностью нативным запуском ROS 1 и ROS 2 как на одном вычислительном комплексе, так и на двух узлах. Схемы запуска представлены на рис. 8 слева и рис. 8 справа соответственно.

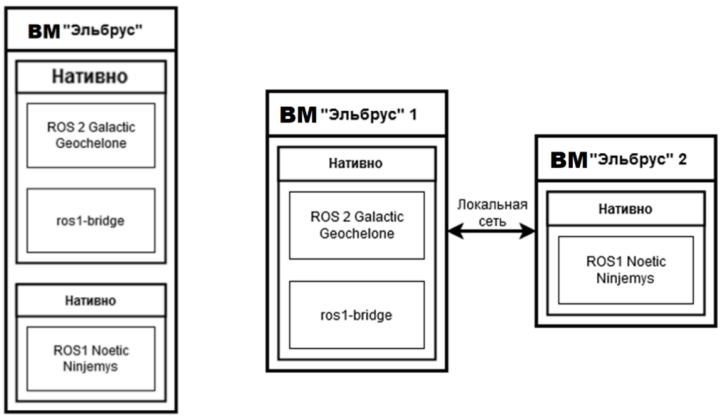

Рис. 8. Схемы 4 запуска ROS 1, ROS 2

Во всех вариантах функциональная проверка включала тестирование механизма публикации и подписки с использованием сообщений различного объёма и структуры. Подтверждена корректность передачи сообщений, отсутствие потерь данных и стабильность передачи при длительном непрерывном функционировании. Дополнительно проверялась работа сервисов и механизмов, включая обработку длительных операций, прерывание задач и повторные вызовы. Подтверждено корректное функционирование ros1_bridge, обеспечивающего взаимодействие между ROS 1 и ROS 2.

В ходе тестирования была проведена оценка производительности с точки зрения измерения средней задержки передачи сообщений. Для определения латентности использовалась публикация сообщений с временными метками и вычисление разницы между моментом отправки и получения. Измерения проводились для сообщений малого, среднего и большого объёма. Нативная реализация продемонстрировала меньшие задержки и более стабильные временные характеристики. При использовании бинарного транслятора увеличение задержки составляло в среднем 10-25% в зависимости от размера сообщений. Тесты пропускной способности показали, что нативный режим обеспечивает более высокую устойчивую частоту публикации без потерь данных, а различие производительности достигало 15-20%.

В распределённых конфигурациях проверялась корректность обнаружения узлов в сети, устойчивость маршрутизации сообщений, а также восстановление соединений после кратковременных сетевых разрывов.

Результаты исследований показали, что экспериментальный образец обеспечивает единое информационное пространство для запуска, функционирования и завершения работы программных компонентов РТК. Также экспериментальный образец поддерживает развёртывание (импорт) программного обеспечения РТК, созданного в других программных средах, и обеспечивает функциональную совместимость с ROS Noetic Ninjemys и ROS 2 Galactic Geochelone.

Таким образом, проведённое тестирование подтвердило корректность функционирования портированных версий ROS 1 и ROS 2, возможность их совместного использования, устойчивость работы в распределённой гетерогенной среде и применимость платформы «Эльбрус» для реализации современных робототехнических комплексов.

Заключение. В данной работе проведено исследование по унификации связующего программного обеспечения РТК на базе ROS 1 Noetic Ninjemys и ROS 2 Galactic Geochelone на отечественной АПП «Эльбрус» с использованием двух подходов: бинарной трансляции уровня приложений и полностью нативной сборки исходных кодов.

Проведён анализ зависимостей, адаптация архитектурно-зависимых компонентов, модификация сценариев сборки и формирование установочных пакетов с использованием кросскомпиляции, соответствующих требованиям АПП «Эльбрус». Реализованная программная среда обеспечивает функционирование обеих версий ROS как раздельно, так и совместно с использованием ros1_bridge, что позволяет поддерживать существующие решения на ROS 1 и одновременно развивать новые разработки на базе ROS 2 на отечественной аппаратной платформе.

Проведённые экспериментальные исследования в различных конфигурациях, включая запуск на одном вычислительном модуле, в смешанном режиме исполнения и в распределённой схеме с двумя вычислительными модулями, подтвердило корректность функционирования программных сред ROS и ROS 2 и функциональную совместимость с программным обеспечением, созданным на других аппаратных архитектурах. Анализ показал, что нативная реализация обеспечивает снижение задержек передачи сообщений и прирост производительности порядка 15-20 % по сравнению с использованием бинарной трансляции при сохранении функциональной эквивалентности, однако режим бинарной трансляции обеспечивает более высокий уровень совместимости и переносимости программных средств между процессорами Эльбрус с разной версией архитектуры системы команд.

Практическая значимость работы заключается в расширении инструментальной базы разработки систем управления РТК на российской АПП «Эльбрус».

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Суминов К.А., Бочаров Н.А., Кирилюк М.А.* Оценка аппаратного состава бортовых вычислительных систем робототехнических комплексов на основе решаемых задач // Известия ЮФУ. Технические науки. – 2025. – № 2 (244). – С. 279-289. – DOI 10.18522/2311-3103-2025-2-279-289. – EDN QLGORV.
2. *Тачков А.А., Козов А.В., Вуколов А.Ю.* Особенности портирования Robot Operating System на программно-аппаратную платформу «Эльбрус» // Программные продукты и системы. – 2019. – № 4. – С. 655-664. – EDN TVJSXJ.
3. *Чучко П.А., Бычков И.Н., Панченко Е.Г.* Проблема унификации модулей на основе процессора «Эльбрус-2С3» // Наноиндустрия. – 2022. – Т. 15, № S8-1 (113). – С. 37-38. – DOI 10.22184/1993-8578.2022.15.8s.37.38.
4. *Бочаров Н.А., Гладких А.С., Парамонов Н.Б., Сенченков С.В.* Возможности микропроцессоров Эльбрус-8С и Эльбрус-8СВ для решения задач робототехники // Роботизация Вооружённых Сил Российской Федерации: Сб. статей V военно-научной конференции, Анапа, 29–30 июля 2020 года. Т. 1. – Анапа: Федеральное государственное автономное учреждение.
5. *Гилязов С.С. Кравцунов Е.М., Пантелеев П.В.* Оптимизация ядра ос Linux для архитектуры «Эльбрус» с поддержкой Numa // Вопросы радиоэлектроники. – 2011. – Т. 4, № 3. – С. 60-69. – EDN NZHDAV.
6. *Бурдинский И.Н., Карабанов И.В., Миронов А.С.* Применение бортовой системы DUNE в качестве базового программного обеспечения узлов сенсорной сети акватории // Вестник ТОГУ. – 2016. – № 3 (42). – С. 13-22.
7. *Гульняшкин А.А., Соколов Н.А*. Использование robot operating system для моделирования захвата известного объекта в известных условиях // Политехнический молодежный журнал. – 2021. – № 5 (58). – DOI 10.18698/2541-8009-2021-5-697. – EDN LSKHPM.
8. *Калянов А.А.* Применение robot operating system для разработки программного обеспечения роботов // Научные исследования и разработки молодых ученых: Матер. научно-практической конференции аспирантов и молодых ученых, посвященной Дню аспиранта, Ульяновск, 20 января 2023 года / под ред. д-ра физ.-мат. наук, профессора В.Н. Голованова. – Ульяновск: Ульяновский государственный университет, 2023. – С. 303-306. – EDN MXZXQX.
9. Noetic Ninjemys: The Last Official ROS 1 Release / Open robotics. – URL: https://www.openrobotics.org/blog/2020/5/23/noetic-ninjemys-the-last-official-ros-1-release (дата обращения: 25.02.2026).
10. *Quigley M., Gerkey B., Smart W.D.* Programming Robots with ROS: a practical introduction to the Robot Operating System. – O'Reilly Media, Inc., 2015.
11. *Bradski G., Kaehler A.* Learning OpenCV: Computer vision with the OpenCV library. – O'Reilly Media, Inc., 2008.
12. *Rusu R. B., Cousins S.* 3d is here: Point cloud library (pcl) // 2011 IEEE international conference on robotics and automation. – IEEE, 2011. – С. 1-4.
13. Package from naoqi_libqi repo. – URL: https://index.ros.org/p (дата обращения 25.02.2026).
14. *Ozansoy C.R., Zayegh A., Kalam A.* The real-time publisher/subscriber communication model for distributed substation systems // IEEE transactions on power delivery. – 2007. – Vol. 22, No. 3. – P. 1411-1423.
15. *Macenski S.* et al. Robot operating system 2: Design, architecture, and uses in the wild //Science robotics. – 2022. – Vol. 7, No. 66. – С. eabm6074.
16. Version 2 of the Robot Operating System (ROS) software stack. – URL: https://github.com/ros2 (дата обращения: 25.02.2026).
17. Getting Started. Where to get started with ROS 2. – URL: https://www.ros.org/blog/getting-started (дата обращения: 25.02.2026).
18. *Pardo-Castellote G.* Omg data-distribution service: Architectural overview // 23rd International Conference on Distributed Computing Systems Workshops, 2003. Proceedings. – IEEE, 2003. – P. 200-206.
19. *Воронов Н.В., Гимпельсон В.Д., Маслов М.В. [и др.].* Система динамической двоичной трансляции x86 → «Эльбрус» // Вопросы радиоэлектроники. – 2012. – Т. 4, № 3. – С. 89-108. – EDN OUMJOF.
20. Операционные системы «Эльбрус». – URL: http://www.mcst.ru/Elbrus_OS (дата обращения: 25.02.2026).
21. *Бочаров Н.А.* Исследование подходов к унификации бортовых вычислительных комплексов // Известия ЮФУ. Технические науки. – 2023. – № 1 (231). – С. 275-287. – DOI 10.18522/2311-3103-2023-1-275-287. – EDN AIYGIE.
22. Руководство по эффективному программированию на платформе «Эльбрус». – URL: http://mcst.ru/doc/elbrus_prog/elbrus-prog-1.2_2024-02-28.pdf (дата обращения: 25.02.2026).

REFERENCES

1. *Suminov K.A., Bocharov N.A., Kirilyuk M.A.* Otsenka apparatnogo sostava bortovykh vychislitel'nykh sistem robototekhnicheskikh kompleksov na osnove reshaemykh zadach [Evaluation of the hardware composition of on-board computing systems of robotic complexes based on the tasks to be solved], *Izvestiya YuFU. Tekhnicheskie nauki* [Izvestiya SFedU. Engineering Sciences], 2025, No. 2 (244), pp. 279-289. DOI 10.18522/2311-3103-2025-2-279-289. EDN QLGORV.
2. *Tachkov A.A., Kozov A.V., Vukolov A.Yu.* Osobennosti portirovaniya Robot Operating System na programmno-apparatnuyu platformu «El'brus» [Features of porting Robot Operating System to the Elbrus hardware and software platform], *Programmnye produkty i sistemy* [Software Products and Systems], 2019, No. 4, pp. 655-664. EDN TVJSXJ.
3. *Chuchko P.A., Bychkov I.N., Panchenko E.G.* Problema unifikatsii moduley na osnove protsessora «El'brus-2S3» [The problem of unification of modules based on the Elbrus-2S3 processor], *Nanoindustriya* [Nanoindustry], 2022, Vol. 15, No. S8-1 (113), pp. 37-38. DOI 10.22184/1993-8578.2022.15.8s.37.38.
4. *Bocharov N.A., Gladkikh A.S., Paramonov N.B., Senchenkov S.V.* Vozmozhnosti mikroprotsessorov El'brus-8S i El'brus-8SV dlya resheniya zadach robototekhniki [Capabilities of Elbrus-8S and Elbrus-8SV microprocessors for solving robotics problems], *Robotizatsiya Vooruzhennykh Sil Rossiyskoy Federatsii: Sb. statey V voenno-nauchnoy konferentsii, Anapa, 29–30 iyulya 2020 goda* [Robotization of the Armed Forces of the Russian Federation: Collection of articles of the V military-scientific conference, Anapa, July 29-30, 2020]. Vol. 1. Anapa: Federal'noe gosudarstvennoe avtonomnoe uchrezhdenie.
5. *Gilyazov S.S. Kravtsunov E.M., Panteleev P.V.* Optimizatsiya yadra os Linux dlya arkhitektury «El'brus» s podderzhkoy Numa [Optimization of the Linux kernel for the Elbrus architecture with Numa support], *Voprosy radioelektroniki* [Questions of Radio Electronics], 2011, Vol. 4, No. 3, pp. 60-69. EDN NZHDAV.
6. *Burdinskiy I.N., Karabanov I.V., Mironov A.S.* Primenenie bortovoy sistemy DUNE v kachestve bazovogo programmnogo obespecheniya uzlov sensornoy seti akvatorii [Application of the onboard DUNE system as basic software for the nodes of the water area sensor network], *Vestnik TOGU* [Bulletin of the Pacific National University], 2016, No. 3 (42), pp. 13-22.
7. *Gul'nyashkin A.A., Sokolov N.A.* Ispol'zovanie robot operating system dlya modelirovaniya zakhvata izvestnogo ob"ekta v izvestnykh usloviyakh [Using a robot operating system to simulate the capture of a known object under known conditions], *Politekhnicheskiy molodezhnyy zhurnal* [Polytechnic Youth Journal], 2021, No. 5 (58). DOI 10.18698/2541-8009-2021-5-697. EDN LSKHPM.
8. *Kalyanov A.A.* Primenenie robot operating system dlya razrabotki programmnogo obespecheniya robotov [Application of robot operating system for developing robot software], *Nauchnye issledovaniya i razrabotki molodykh uchenykh: Mater. nauchno-prakticheskoy konferentsii aspirantov i molodykh uchenykh, posvyashchennoy Dnyu aspiranta, Ul'yanovsk, 20 yanvarya 2023 goda* [Scientific research and development of young scientists: Proceedings of the scientific and practical conference of graduate students and young scientists dedicated to the Postgraduate Day, Ulyanovsk, January 20, 2023], ed. by dr. of phys. and math. sc., professor V.N. Golovanova. Ul'yanovsk: Ul'yanovskiy gosudarstvennyy universitet, 2023, pp. 303-306. EDN MXZXQX.
9. Noetic Ninjemys: The Last Official ROS 1 Release, Open robotics. Available at: https://www.openrobotics.org/blog/2020/5/23/noetic-ninjemys-the-last-official-ros-1-release (accessed 25 February 2026).
10. *Quigley M., Gerkey B., Smart W.D.* Programming Robots with ROS: a practical introduction to the Robot Operating System. O'Reilly Media, Inc., 2015.
11. *Bradski G., Kaehler A.* Learning OpenCV: Computer vision with the OpenCV library. O'Reilly Media, Inc., 2008.
12. *Rusu R. B., Cousins S.* 3d is here: Point cloud library (pcl), *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 1-4.
13. Package from naoqi_libqi repo. Available at: https://index.ros.org/p (accessed 25 February 2026).
14. *Ozansoy C.R., Zayegh A., Kalam A.* The real-time publisher/subscriber communication model for distributed substation systems, *IEEE transactions on power delivery*, 2007, Vol. 22, No. 3, pp. 1411-1423.
15. *Macenski S.* et al. Robot operating system 2: Design, architecture, and uses in the wild, *Science robotics*, 2022, Vol. 7, No. 66, pp. eabm6074.
16. Version 2 of the Robot Operating System (ROS) software stack. Available at: https://github.com/ros2 (accessed 25 February 2026).
17. Getting Started. Where to get started with ROS 2. Available at: https://www.ros.org/blog/getting-started (accessed 25 February 2026).

18. *Pardo-Castellote G.* Omg data-distribution service: Architectural overview, *23rd International Conference on Distributed Computing Systems Workshops, 2003. Proceedings*. IEEE, 2003, pp. 200-206.
19. *Voronov N.V., Gimpel'son V.D., Maslov M.V. [et al.].* Sistema dinamicheskoy dvoichnoy translyatsii x86 → «El'brus» [Dynamic binary translation system x86 → "Elbrus"], *Voprosy radioelektroniki* [Questions of radio electronics], 2012, Vol. 4, No. 3, pp. 89-108. EDN OUMJOF.
20. Operatsionnye sistemy «El'brus» [Elbrus operating systems]. Available at: http://www.mcst.ru/Elbrus_OS (accessed 25 February 2026).
21. *Bocharov N.A.* Issledovanie podkhodov k unifikatsii bortovykh vychislitel'nykh kompleksov [Study of approaches to the unification of onboard computing systems], *Izvestiya YuFU. Tekhnicheskie nauki* [Izvestiya SFedU. Engineering Sciences], 2023, No. 1 (231), pp. 275-287. DOI 10.18522/2311-3103-2023-1-275-287. – EDN AIYGIE.
22. Rukovodstvo po effektivnomu programmirovaniyu na platforme «El'brus» [Guide to effective programming on the Elbrus platform]. Available at: http://mcst.ru/doc/elbrus_prog/elbrus-prog-1.2_2024-02-28.pdf (accessed 25 February 2026).

Бочаров Никита Алексеевич – ПАО «ИНЭУМ им. И.С. Брука»; e-mail: bocharov.na@phystech.edu; г. Москва, Россия; тел.: +79167346437; д.т.н.; зам. руководителя управления – главный научный сотрудник.

Суминов Константин Александрович – ПАО «ИНЭУМ им. И.С. Брука»; e-mail: suminov_k@ineum.ru; 119334, г. Москва, Россия; тел: +74991355336; к.т.н.; начальник отдела.

Кирилюк Михаил Андреевич – ПАО «ИНЭУМ им. И.С. Брука»; e-mail: kirilyuk_m@ineum.ru; г. Москва, Россия; тел.: +79055273011; к.т.н.; начальник отдела.

Парамонов Николай Борисович – ПАО «ИНЭУМ им. И.С. Брука»; e-mail: paramon_n@ineum.ru; г. Москва, Россия; тел: +74991355336; д.т.н.; профессор; руководитель управления.

Bocharov Nikita Alekseevich – JSC «INEUM»; e-mail: bocharov.na@phystech.edu; Moscow, Russia; phone: +79167346437; dr. of eng. sc.; deputy head of division – chief researcher.

Suminov Konstantin Aleksandrovich – JSC «INEUM»; e-mail: suminov_k@ineum.ru; Moscow, Russia; phone: +74991355336; cand. of eng. sc.; head of department.

Kirilyuk Mikhail Andreevich – JSC «INEUM»; e-mail: kirilyuk_m@ineum.ru; Moscow, Russia; phone: +79055273011; cand. of eng. sc.; head of department.

Paramonov Nikolay Borisovich – JSC «INEUM»; e-mail: paramon_n@ineum.ru; Moscow, Russia; phone: +74991355336; dr. of eng. sc.; professor; head of division.

УДК 004.853 DOI 10.18522/2311-3103-2026-2-271-285

И.С. Фомин, А.А. Базельцев

ИСПОЛЬЗОВАНИЕ СИСТЕМЫ КАМЕР КРУГОВОГО ОБЗОРА ДЛЯ ОБЕСПЕЧЕНИЯ БЕЗОПАСНОСТИ ДВИЖЕНИЯ МОБИЛЬНОГО РОБОТА

Рассматривается проблема обеспечения безопасности движения мобильной робототехнической платформы в окружении с известными классами объектов на основе результатов обнаружения объектов по системе камер кругового обзора. Хорошо известны решения по обеспечению безопасности по камерам с использованием оптических потоков или иных способов выделения подвижных объектов в поле зрения. Здесь предлагается использовать для обеспечения безопасности положение и форму объектов, рассчитанные по результатам работы нейросетевого детектора. Система кругового обзора (СКО) состоит из 4 камер с широким углом зрения. Рассмотрены несколько подходов к подаче объектов в нейросетевой детектор. Показаны различия по качеству и быстродействию для этих подходов, сформированы методические рекомендации по использованию. Для расчета положений объектов в системе координат камер и через положения камер - в СК робота – предложены 2 решения на основе калибровки камер и некоторых допущений. В первом случае положение робота на поверхности принимается за горизонтальное, а высота камеры над поверхностью – постоянной и известной, а во втором – предварительно известным принимается один метрический размер объекта для каждого класса. Предложенное решение