

УДК 004.7, 004.8, 004.9

МЕХАНИЗМЫ ОБМЕНА ИНФОРМАЦИЕЙ И ПЕРЕДАЧИ ЗНАНИЙ ДЛЯ ЗАДАЧ ВЗАИМОДЕЙСТВИЯ В СРЕДЕ АВТОНОМНЫХ ИНТЕЛЛЕКТУАЛЬНЫХ РОБОТОТЕХНИЧЕСКИХ СИСТЕМ В НЕШТАТНЫХ СИТУАЦИЯХ

Катулин Михаил Сергеевич¹, Кузнецов Евгений Алексеевич², Решетников Андрей Геннадьевич³, Рябов Андрей Русланович⁴, Семашко Сергей Владимирович⁵, Ульянов Сергей Викторович⁶

¹Старший инженер;

Объединенный институт ядерных исследований,
141980, Московская обл., г. Дубна, ул. Жолио-Кюри, д. 6;
e-mail: mikhail@katulin.ru.

²Инженер-программист;

Объединенный институт ядерных исследований,
141980, Московская обл., г. Дубна, ул. Жолио-Кюри, д. 6;
e-mail: evkuz@jinr.ru.

³Старший научный сотрудник;

Объединенный институт ядерных исследований,
141980, Московская обл., г. Дубна, ул. Жолио-Кюри, д. 6;
e-mail: agreshetnikov@gmail.com.

⁴Старший лаборант;

Объединенный институт ядерных исследований,
141980, Московская обл., г. Дубна, ул. Жолио-Кюри, д. 6;
e-mail: rar.16@uni-dubna.ru.

⁵Инженер-программист I кат.;

Объединенный институт ядерных исследований,
141980, Московская обл., г. Дубна, ул. Жолио-Кюри, д. 6;
e-mail: semashko@jinr.ru.

⁶Доктор физико-математических наук, профессор;

Государственный университет «Дубна»;
141980, Московская обл., г. Дубна, ул. Университетская, 19;
Ведущий научный сотрудник;
Объединенный институт ядерных исследований,
141980, Московская обл., г. Дубна, ул. Жолио-Кюри, д. 6;
e-mail: ulyanovsv46_46@mail.ru.

При решении сложных задач автоматизации в проекте «Индустрия 4.0» с использованием нескольких интеллектуальных автономных робототехнических систем возникает необходимость определения механизмов обмена информации и передачи знаний от одной робототехнической системы к другой (типа системы «master – slave»). Существует множество механизмов сетевого взаимодействия, при котором каждый элемент имеет свои сильные и слабые стороны. При разработке интеллектуальных робототехнических систем на испытательном полигоне в лаборатории информационных технологий ОИЯИ (МЛИТ), было выявлено несколько основных задач взаимодействия, требующих выбора специальных механизмов и протоколов передачи данных и знаний. В статье рассмотрены основные механизмы обмена информацией, описаны основные преимущества и недостатки. Представлены задачи, требующие выбора механизма взаимодействия, а также приведены результаты решения этих задач.

Ключевые слова: автономные роботы, операционная система роботов (ROS), сетевое взаимодействие, протокол передачи данных, обмен знаниями, дистанционная настройка.

Для цитирования: Механизмы обмена информацией и передачи знаний для задач взаимодействия в среде автономных интеллектуальных робототехнических систем в нештатных ситуациях / М. С.

Катулин [и др.] // Системный анализ в науке и образовании: сетевое научное издание. 2021. № 4. С. 44–62. URL : <http://sanse.ru/download/450>.

MECHANISMS OF INFORMATION EXCHANGE AND KNOWLEDGE TRANSFER FOR INTERACTION TASKS IN THE HAZARD ENVIRONMENT OF AUTONOMOUS INTELLIGENT ROBOTIC SYSTEMS

Katulin Mikhail S.¹, Kuznetsov Evgeny A.², Reshetnikov Anderey G.³, Ryabov Andrey R.⁴, Semashko Sergey V.⁵, Ulyanov Sergey V.⁶

¹Senior Engineer;

Joint Institute for Nuclear Research,

6 Joliot-Curie Str., Dubna, Moscow region, 141980, Russia;

e-mail: mikhail@katulin.ru.

²Engineer-programmer;

Joint Institute for Nuclear Research,

6 Joliot-Curie Str., Dubna, Moscow region, 141980, Russia;

e-mail: evkuz@jinr.ru.

³Ph.D. in informatic, Associate Professor;

Joint Institute for Nuclear Research,

6 Joliot-Curie Str., Dubna, Moscow region, 141980, Russia;

e-mail: agreshetnikov@gmail.com.

⁴Senior Laboratory Assistant;

Joint Institute for Nuclear Research,

6 Joliot-Curie Str., Dubna, Moscow region, 141980, Russia;

e-mail: rar.16@uni-dubna.ru.

⁵Engineer-programmer 1 cat.;

Joint Institute for Nuclear Research,

6 Joliot-Curie Str., Dubna, Moscow region, 141980, Russia;

e-mail: semashko@jinr.ru.

⁶Grand PhD in Physical and Mathematical Sciences, professor;

Dubna State University;

19 Universitetskaya Str., Dubna, Moscow region, 141980, Russia;

Leading Researcher of LIT JINR;

Joint Institute for Nuclear Research;

6 Joliot-Curie Str., Dubna, Moscow region, 141980, Russia;

e-mail: ulyanovsv@mail.ru

Solving complex automation tasks in the Industry 4.0 project using several intelligent autonomous robotic systems, it becomes necessary to determine the mechanisms of information exchange and knowledge transfer from one robotic system to another (such as the master - slave system). There are many mechanisms of network interaction, each of them has its own strengths and weaknesses. During the development of intelligent robotic systems at the testing ground at the JINR Information Technology Laboratory (MLIT), several basic interaction tasks were identified that require the choice of special mechanisms and protocols for data and knowledge transmission. The article discusses the main mechanisms of information exchange, describes the main advantages and disadvantages. The tasks requiring the choice of an interaction mechanism are presented, and the results of solving these tasks are also presented.

Keywords: autonomous robots, robot operating system (ROS), network interaction, data transfer protocol, knowledge exchange, remote configuration.

For citation: Katulin M. S. et al. Mechanisms of information exchange and knowledge transfer for interaction tasks in the hazard environment of autonomous intelligent robotic systems. System Analysis in Science and Education, 2021;(4):44–62(In Russ). Available from: <http://sanse.ru/download/450>.

1. Современное состояние проблемы

Использование группы взаимодействующих роботов позволяет расширить список задач, решаемых с помощью робототехнических систем. Известны случаи применения групп взаимодействующих роботов при решении задач наблюдения [1], исследования [2], поиска и спасения людей [3]. При этом взаимодействие роботов осуществлялось разными способами: от звуковых волн до беспроводных специальных сетей (*ad hoc networks*).

В первую очередь рассмотрим общедоступные и широко-используемые решения на основе повсеместно распространённых линий связи, например *Wi-Fi*.

Поиск существующих решений выявил чёткое разделение двух типов взаимодействия: машина-машина (*M2M*) и устройство-устройство (*D2D*) [4,5]. В первом случае взаимодействие может происходить через базовую станцию (узел), основное внимание уделяется взаимодействию двух устройств и не так важен маршрут пути, преодолеваемый пакетом данных. Во втором случае речь идёт о непосредственном взаимодействии устройств «напрямую», минуя базовые станции. *D2D* взаимодействие чаще всего упоминается, когда речь идёт о защищённом обмене информацией, например внутри безопасных зон, окружённых глушащим радиоволны фоновым сигналом [6]. Важно отметить, что большинство работ описывающих *M2M* и *D2D* взаимодействия представляют работу с устройствами интернета вещей (*IoT*) [7,8] и поэтому не в полной мере удовлетворяют поставленным в статье задачам из-за узкой специализации подобных устройств.

Существует способ обеспечения взаимодействия между узлами робототехнической системы и между робототехнической системой и внешними устройствами, основанный на программном обеспечении *Robot Operation System (ROS)* [9]. *ROS* – программное обеспечение с открытым исходным кодом, фреймворк для программирования роботов, предоставляющий функциональность для распределённой работы. Данное программное обеспечение позволяет связать сенсоры и исполнительные устройства робота в единые алгоритмы взаимодействия. Файлы описания такого функционала и набор микропрограмм, отвечающих за каждое конкретное устройство, собираются в пакеты, которые могут быть загружены в открытые репозитории.

Примечание. О популярности *ROS* можно судить по факту: на 2020 год было создано и загружено в репозитории более 500 000 000 пакетов, реализующих тот или иной функционал для роботов [10]. Согласно документации, взаимодействие может осуществляться через запросы *XMLRPC* по TCP/IP. А это означает, что можно не только подключаться к внешним устройствам, но и полноценно взаимодействовать, запуская удалённое выполнение команд и даже управлять группой устройств. Например, используя разные схемы организации управления группой, при помощи *ROS* и *ROS2* организовали группу беспилотных летательных аппаратов [11].

Другой способ взаимодействия – создание собственного *REST API* на основе *HTTP* запросов [12]. Таким образом робототехническая система выступает в роли сервиса, которым можно пользоваться и управлять через веб браузер и, соответственно, через HTTP запросы.

Отдельно следует выделить класс промышленных фреймворков для синхронизации, контроля и управления в распределённых промышленных системах. Применительно к задачам реализации программной части системы управления ускорительными комплексами (наподобие *NICA*) таким инструментарием является «*TANGO CONTROLS*» (*TC*) [13-16]. Это современная автоматизированная система управления (*ACU*), основанная на *CORBA*, которая активно развивается в европейских ускорительных организациях, таких как *ESRF*, *Alba*, *Soleil* в течение последних десяти лет. Основная концепция *TANGO* схожа с *ROS* и предоставляет единообразие в управление всеми приборами и подсистемами (в том числе классическими ПИД регуляторами). Системы управления на основе TC позволяют создавать распределённые системы управления любой иерархической сложности.

Структура системы управления на базе TC представлена на рис. 1, где также показаны потоки данных [13] для адаптации и обучения интеллектуальных систем управления (*ИСУ*), встраиваемые программные контроллеры и интерфейсы программного инструментария, разработанные и применяемые в данной работе.

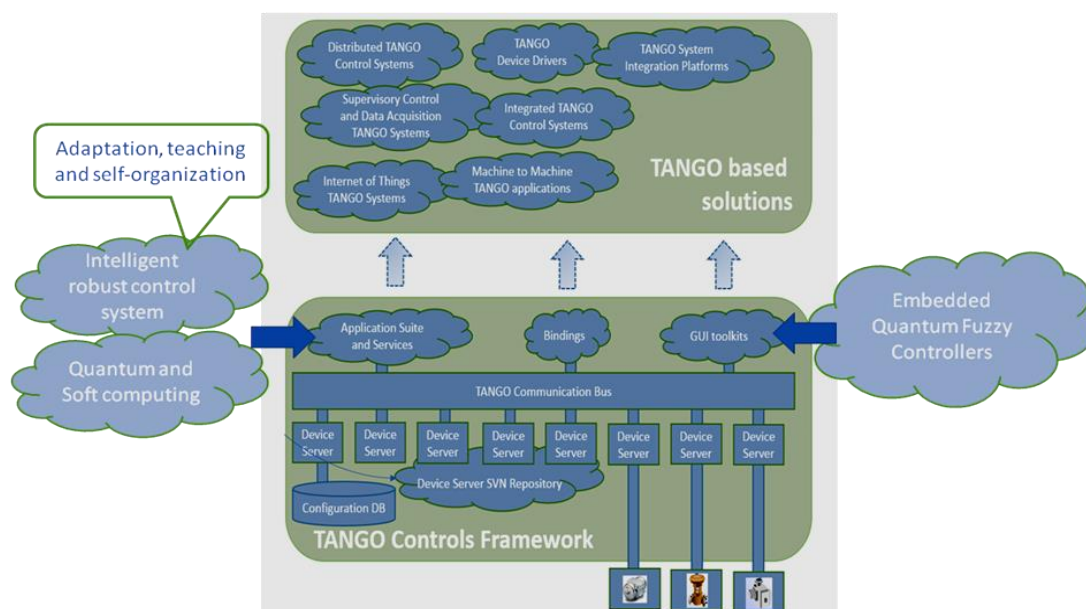


Рис. 1. Взаимосвязь структуры традиционной аппаратной реализацией АСУ и ИСУ

Вычислительная система ускорительных комплексов, построенная на базе технологий ТС, реализует простую и быструю разработку интерфейсов, сторонних приложений и их развёртывание. Однако, базовые решения ТС не поддерживают технологии дистанционного обмена знаниями и интеллектуального робастного управления на основе квантовых и мягких вычислений. Вследствие этого операторы не обеспечены соответствующим инструментарием, позволяющим вводить дополнительные качества управления – адаптацию, обучение, самоорганизацию в систему управления элементами ускорительного комплекса. Каждый раз во время запуска или реконфигурации оборудования, например перевода в новое целевое состояние параметров пучка ускорителя, необходимо проводить заново настройку ускорительного комплекса. Представленные недостатки могут быть компенсированы применением сетевого взаимодействия с инструментами дистанционной настройки и управления базами знаний, например, Оптимизатором баз знаний (ОБЗ) на мягких и квантовых вычислениях *SCOptKB TM* [17-19].

2. Постановка задачи и выбранные методы решения

Развитие технологий когнитивного взаимодействия робототехнических систем, позволяющих решать задачи интеллектуального иерархического управления за счёт перераспределения знаний и функций управления, является одной из ключевых задач в современной робототехнике [17]. Современные подходы к решению данной задачи основываются на теории многоагентных систем, теории роевого искусственного интеллекта, нечётких контроллеров с интегрируемыми базами знаний и др. [20, 21]. За счёт синергетического эффекта обмена информацией многоагентная система способна решать сложные динамические задачи по выполнению совместной работы: поставленная задача может не выполняться каждым элементом системы в отдельности в разнообразных средах без внешнего управления, контроля или координации, но обмен знаниями и информацией позволяет совершать совместную полезную работу для достижения поставленной цели управления. При этом на основе переданной базы знаний может быть совершена полезная работа намного большая по количеству и достигнутому эффекту управления, чем работа, затраченная на её передачу [22].

Речь идёт о сети слабо связанных между собой автономных интеллектуальных робототехнических системах, совместно работающих в целях решения задач, которые выходят за рамки индивидуальных возможностей. Различные узлы подобной сети, как правило, имеют различный уровень интеллектуализации (знания, алгоритмы, вычислительные базы) и разные информационные ресурсы при проектировании. Однако, каждый узел должен быть способен модифицировать своё поведение в зависимости от обстоятельств, а также планировать свои стратегии коммуникации и кооперации с другими узлами. Здесь показателями уровня кооперации являются: характер распределения задач,

объединение различных информационных ресурсов и, конечно, возможность решения общей проблемы в заданное время [23, 24].

Моделирование решений прикладных задач с помощью автономных интеллектуальных робототехнических систем на испытательном полигоне Лаборатории Информационных Технологий ОИЯИ (МЛИТ), непосредственно связано с разработкой и внедрением информационных технологий интеллектуального управления на основе квантовых и мягких вычислений в прикладных задачах института. Типичные задачи, решаемые на полигоне – это взаимодействие нескольких автономных робототехнических систем с целью выполнения совместных задач. Выделим основные из них: манипуляция над физическим объектом; управление сложной технической системой в условиях нештатных и непредвиденных ситуаций управления; дистанционная настройка объектов управления, локальная и глобальная оптимизация процесса в соответствии с заданным критерием качества.

Автономные робототехнические системы, работающие на испытательном полигоне, используют в своей основе микрокомпьютер *Raspberry PI 4* с установленными на него операционной системой *Ubuntu* и программным обеспечением *ROS*. Такая робототехническая система является частным случаем сложной технической системы и состоит из трёх основных элементов: сенсорной системы, исполнительное устройство и регулятора (управляющей программой). Для достижения целей управления в этом случае необходимо наличие надёжного и быстрого механизма обмена информацией в рамках как одной автономной системы, так и комплекса в целом.

В задачах, где несколько автономных интеллектуальных робототехнических систем производят манипуляции над одним и тем же физическим объектом требования к механизму обмена информацией меняются, так как чаще всего возникает необходимость согласования действий между разными системами. Например, необходимо передавать через имеющиеся линии связи информацию, команды, параметры настройки и базы знаний. В одном случае взаимодействие подразумевает обмен информационными сообщениями между устройствами. Сообщения могут содержать команды, статусы выполнения команд или значения показаний каких-либо датчиков. Важно отметить, что скорость передачи информации в этом случае не играет первостепенную роль, на первом месте стоит простота реализации, так как все автономные робототехнические системы, участвующие во взаимодействии, должны быть совместимы с данным механизмом. Плюсом также является простота дешифровки сообщений для целей отладки и анализа эффективности работы. В другом случае взаимодействие может подразумевать обмен большими объёмами данных: файлами, базами знаний, а также потоковая передача данных. Это принципиально другие требования и для их удовлетворения потребуются механизм, обеспечивающий высокую скорость передачи и надёжность соединения. Примером такой задачи может быть передача потокового видео с телекамеры одного робота другому.

Отдельно нужно отметить задачи, требующие обеспечения резервного способа обмена информацией, который позволит взаимодействовать автономным устройствам в случае, если нет возможности обеспечить стабильное беспроводное соединение или если существует вероятность, нештатной ситуации, при которой это соединение будет нарушено.

Таким образом, можно выделить четыре базовые задачи взаимодействия автономных интеллектуальных робототехнических систем:

- обмен данными, обеспечивающий максимальную производительность в рамках одной системы;
- обмен информацией между устройствами, решающими одну задачу;
- передача больших объёмов данных между устройствами (базы знаний);
- обмен информацией в условиях отсутствия связи с центральным диспетчерским пунктом.

Как видно, приведённые задачи сильно различаются как по исходным данным, так и по требованиям к их решению. В то же время, для построения надёжной интеллектуальной многоагентной робототехнической системы необходимо обеспечить решение каждой из них.

2.1. Обмен данными, обеспечивающий максимальную производительность в рамках одной системы

Для обмена информацией в рамках одной системы уже существует механизм *ROS “TOPICS”*. С помощью него осуществляется публикация сообщений с данными в очередь, из которой эти данные

могут быть получены несколькими потребителями. Можно было бы остановиться на этом решении, однако, реализация “TOPICS” в ROS достаточно «громоздкая».

Рассмотрим результаты исследования. Было проанализировано поведение трафика сообщений при различных частотах отправки (рис. 2-5).

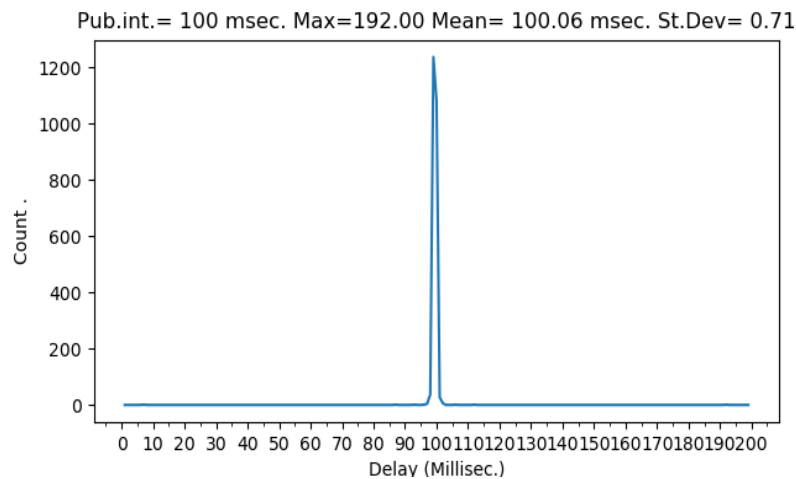


Рис. 2. Поведение трафика при частоте обмена сообщениями 10 Гц (millisecond – msec - мс)

При обмене сообщениями с частотой 10 Гц (см. рис. 2) был обнаружен статистический выброс в задержке сообщения до 192 мс. При частоте 20 Гц (рис. 3) есть выброс в задержке сообщения до 132 мс, но стандартное отклонение начинает увеличиваться.

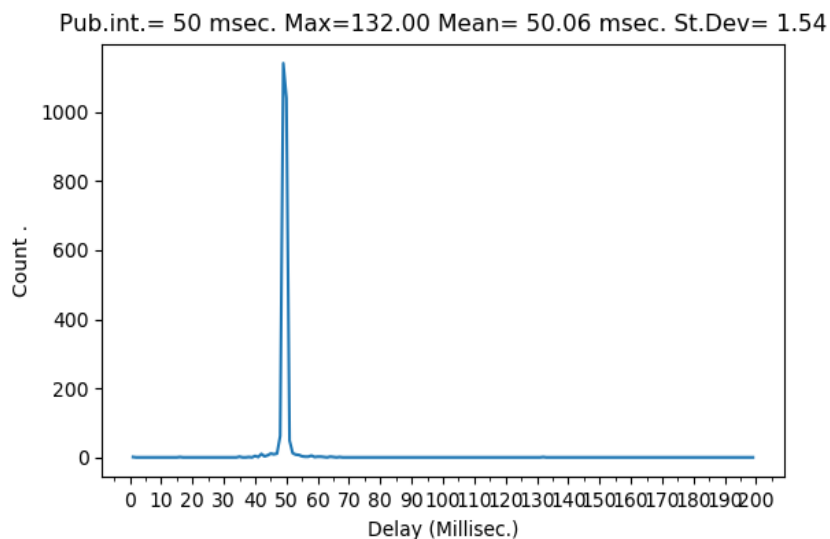


Рис. 3. Поведение трафика при частоте обмена в 20 Гц

При частоте 25 Гц (рис. 4) есть выброс в задержке сообщения до 141 мс, при этом стандартное отклонение достигает 11.49 мс и сообщения начинают «слипаться» – появление пика в районе 0. Средний интервал равен 40.156 мс, но медиана интервала явно ближе к 42-43 мс. Таким образом, сообщения либо «слипаются», либо идут с интервалом в 42 мс.

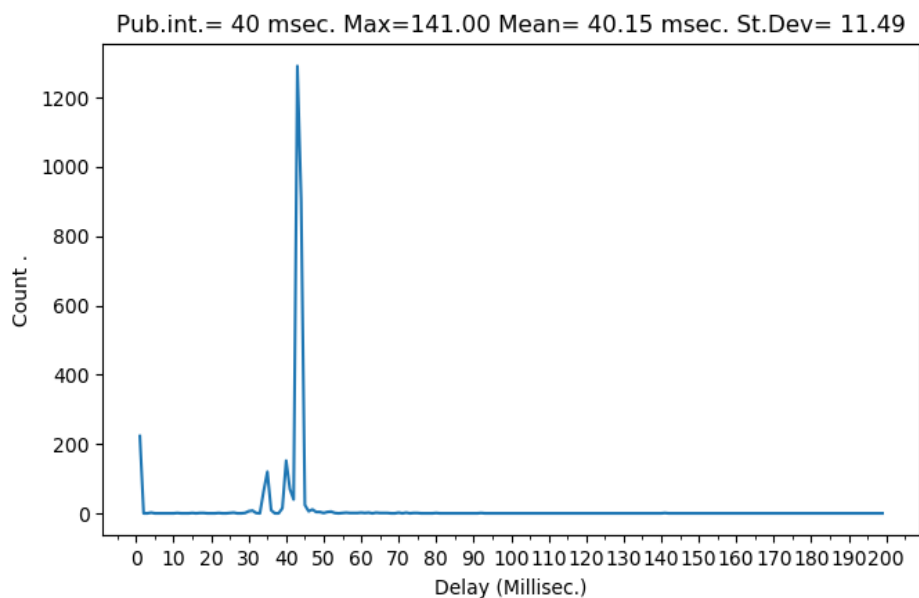


Рис. 4. Поведение трафика при частоте обмена сообщениями в 25 Гц

При частоте 40 Гц (рис. 5) есть выброс в задержке сообщения до 127 мс, механизм демонстрирует неработоспособность.

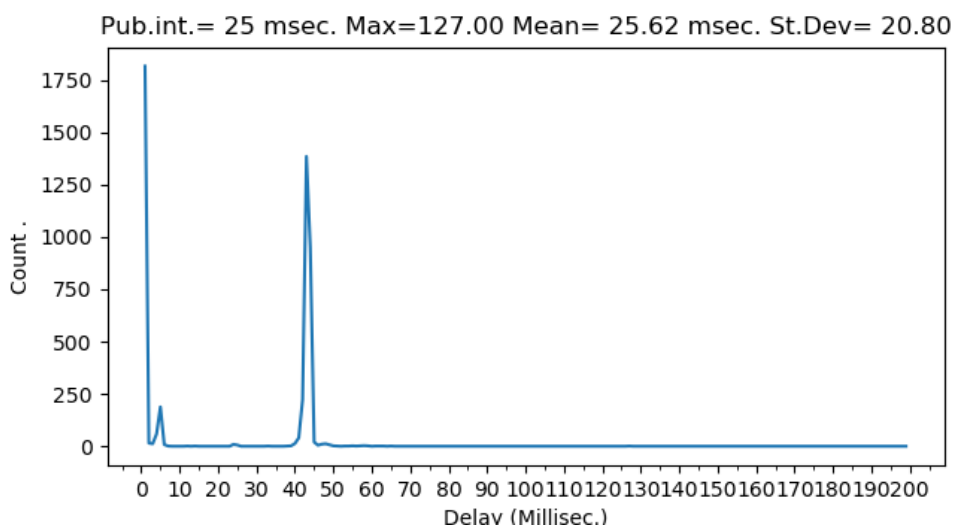


Рис. 5. Поведение трафика обмена при частоте 40 Гц

Стандартное отклонение на уровне среднего значения. Большая часть сообщений «слиплась». Медиана интервала между сообщениями опять между 40 и 45.

Из собранной статистики можно сделать вывод, что максимальная рабочая частота механизма ROS “TOPICS” – 20 Гц. Очевидно, что данный механизм не применим в задачах, где требуется быстрое реагирование, т.е. сообщения должны передаваться в режиме близком к реальному времени.

После анализа распространённых существующих систем, было принято решение остановиться на системе *RabbitMQ*, которая обеспечивает необходимую передачу информации между различными задачами. Существуют реализации для всех распространённых языков [25]. Механизм обмена данными может работать, как через протокол *TCP/IP* так и через механизм *IPC Posix*, который обладает очень высокой пропускной способностью (обмен память-память). Время передачи блока размером в несколько мегабайт составляет меньше 50 мкс, что позволяет обмениваться базами знаний и изображениями с сенсорных систем (например, видео камер) в реальном масштабе времени. Проведённые измерения показали, что на частотах до 1 кГц не заметно никакого «проседания» по скорости. Дан-

ные характеристики полностью удовлетворяю требованиям, необходимые при поиске решений, исследуемых задачам.

2.2. Обмен информацией между устройствами, решающими одну совместную задачу

Не все устройства в своей работе используют *ROS*, тем не менее все имеют подключение к локальной сети и могут обмениваться информацией, используя сети *Wi-Fi* или *Ethernet*. Крайне желательно, чтобы простые команды или запросы на получение информации можно было выполнить через стандартные протоколы и приложения: *ssh*, *telnet*, *HTTP (web)*. Это бывает необходимо при отладке в полевых условиях. Для этих целей был выбран *REST* (от англ. *Representational State Transfer* – «передача репрезентативного состояния» или «передача «самоописываемого» состояния»). В качестве транспорта для него используется текстовый протокол *HTTP*, реализация которого существует на большинстве языков программирования, а также на микроконтроллерах. *HTTP*-запрос не требует постоянного *TCP*-соединения, может быть выполнен вручную из любого браузера и прост в реализации.

Так как основная цель обмена сообщениями между различными устройствами – обмен числовыми и текстовыми данными мы обратили внимание на формат данных *JSON*. Этот текстовый формат является «человеко-читаемым» и позволяет хранить информацию о состоянии объектов.

Известна проблема – если надо передать параметр запроса, который содержит двоичное значение, то его надо преобразовывать. Чтобы избежать сложностей с перекодировкой, было принято следующее соглашение:

- Если имя параметра заканчивается на *_HEX* – то это означает, что значение параметра представляет собой преобразование вида $0x1A \rightarrow '1A'$. Т.е. шестнадцатеричное значение *1A* преобразуется в строку *'1A'*. Это менее экономично, чем *MIME* кодировка, но более «читаемая» при отладке и проще в реализации на микроконтроллерах.
- Кроме того, принято соглашение, что если в имя ключа/параметра заканчивается на *_Z*, то содержимое упаковано в формате *ZLIB* и преобразовано в стиле *_HEX*. В *JavaScript* это соответствует библиотеке *Pako*.
- После принятия решений о способе взаимодействия и формате данных, посылаемых к устройствам и от них, возникла задача описания: какие именно данные будут передаваться.

Каждое устройство может быть оснащено датчиками и хранить какую-то информацию, доступную только на этом устройстве. При взаимодействии нескольких устройств эта информация может быть важна. Значит имеет место формат запроса конкретных данных. В концепции *ROS* есть аналог такого механизма – *Service* [26]. Этот механизм позволяет получить какие-то данные, сделав запрос с определёнными параметрами. Также в *ROS* есть другой механизм – *Action*. Он используется, когда требуется выполнить продолжительное действие, например движение исполнительными механизмами. Этот тип запросов был принят за основу, так это подходящий способ передавать команды устройствам.

В представленной работе все описываемые устройства являются автономными и могут взаимодействовать друг с другом, а значит важно чтобы каждое устройство умело отправлять информацию о своём состоянии: статус работы модулей, какие команды исполняются в данный момент и т.п. Также для отладки и мониторинга работы могут потребоваться данные о истории состояний устройства.

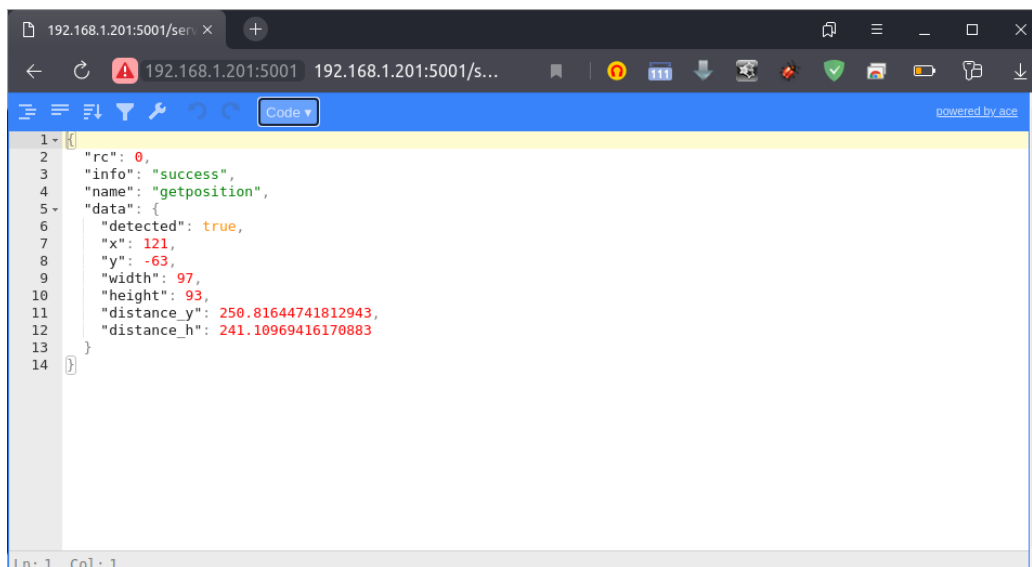
Все устройства подключены к одной локальной сети и имеют встроенные *web*-серверы (для реализации *HTTP*) поэтому адреса устройства имеют формат: *http://device_addr:device_port*.

Далее для простоты восприятия полный адрес будет опускаться, например вместо *http://device_addr:device_port/status* будет просто */status*. Протокол описывает следующие типы запросов:

- запрос информации из *Service*;
- запуск *Action*;
- остановка *Action*;

- запрос статуса работы робототехнической системы или конкретного *Action*;
- запрос логов.

В качестве ответа на запрос направляется текст в формате *JSON* (рис. 6)



```
1 {
2   "rc": 0,
3   "info": "success",
4   "name": "getposition",
5   "data": {
6     "detected": true,
7     "x": 121,
8     "y": -63,
9     "width": 97,
10    "height": 93,
11    "distance_y": 250.81644741812943,
12    "distance_h": 241.10969416170883
13  }
14 }
```

Рис. 6. Пример ответа на запрос информации из Service с названием *getposition*.

Так как формат *JSON* позволяет передавать не только текстовые, но и бинарные данные, данный протокол позволяет обеспечить интеллектуализацию робототехнических систем за счёт обмена информацией и знаний от устройств к центральной управляющей программе и от устройства к устройству для децентрализации управления технологическим процессом.

Полное описание протокола представлено в Приложении 1 к данной работе.

2.3. Передача больших массивов данных между взаимодействующими устройствами

При реальном применении возникают ситуации, когда для выполнения поставленной задачи недостаточно передать несколько параметров с одного устройства на другое. Может потребоваться надёжное соединение для передачи больших объёмов данных, порядка десятков мегабайт, или непрерывная трансляция показаний каких-то датчиков. Самая распространённая ситуация, это когда одно из устройств имеет камеру и требуется обеспечить трансляцию видео на другое устройство.

Для установки таких соединений был выбран протокол передачи *WebSocket*. Он обеспечивает достаточно хорошую производительность и имеет реализацию практически на всех платформах. Контроль целостности переданной информации лежит на протоколе *TCP*, на базе которого построен *WebSocket*. Таким образом, если массив данных был передан без ошибок, то дополнительный контроль не требуется.

Как уже было указано выше, устройства оснащены необходимым оборудованием для подключения к единой локальной сети. Это означает, что организация связи между устройствами через *WebSocket* не подставляет никаких технических сложностей.

2.4. Механизм резервного обмена информацией в условиях отсутствия связи с центральным диспетчерским пунктом

Предполагается, что все устройства работают в одной локальной сети и связь осуществляется через *Wi-Fi* или *Ethernet* соединение. Однако, нельзя исключать ситуаций, когда автономные устройства могут работать в условиях, где нет возможности обеспечить стабильную *Wi-Fi* связь. В таких случаях устройствам необходимо иметь резервный, дублирующий канал связи. На данный момент для этих целей используется технология *ZigBee*.

Zigbee – спецификация, ориентированная на приложения, требующие гарантированной безопасной передачи данных при относительно небольших скоростях и возможности длительной работы сетевых устройств от автономных источников питания (батарей). Основная особенность технологии *Zigbee* заключается в том, что она при малом энергопотреблении поддерживает не только простые топологии сети («точка-точка», «дерево» и «звезда»), но и самоорганизующуюся и самовосстанавливающуюся ячеистую (*mesh*) топологию с ретрансляцией и маршрутизацией сообщений [27].

3. Результаты реализации решения и его особенности

Работы, выполняемые на полигоне связаны, в том числе, с внедрением и апробацией разрабатываемых технологий интеллектуального управления на основе квантовых и мягких вычислений. В созданные робототехнические макеты (типовые объекты теории правления) заложены технологии проектирования и обмена базами знаний. Описанные выше технические решения являются базисом коммуникационной среды распределенной робототехнической системы и применяются в рамках решения задач по обмену информацией, действиями и знаниями.

3.1. Пример обмена информацией

На полигоне установлены три стационарных манипулятора, все оборудованы системой машинного зрения (рис. 7).

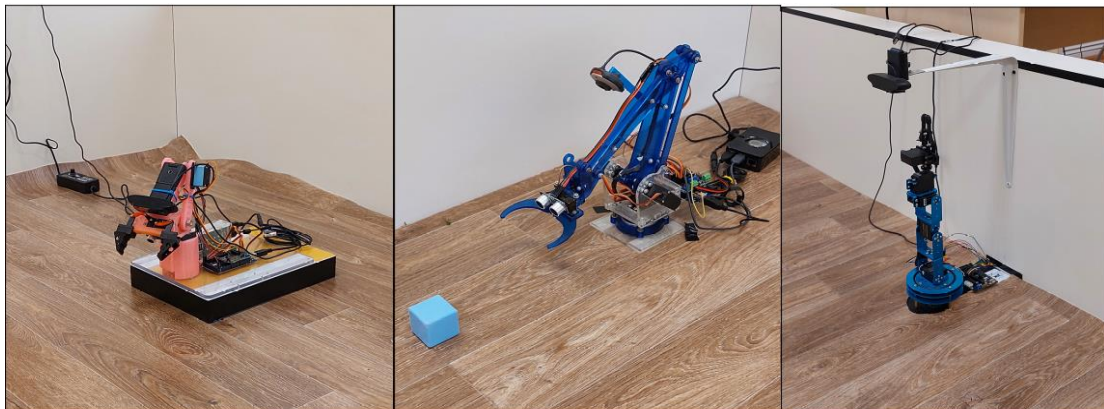


Рис. 7. Стационарные манипуляторы, расположенные на полигоне

На всех манипуляторах установлен *http*-сервер и реализовано взаимодействие с использованием *REST*-протокола, описанного выше. Таким образом манипуляторы могут принимать команду на действия, отправлять информацию о состоянии выполнения команд (рис. 8).

```
192.168.1.158:5001/stat x +
← ↻ ⚠ 192.168.1.158:5001 192.168.1.158:5001/st...
{
  "rc": 0,
  "info": "success",
  "name": "BlueMan",
  "status": "init",
  "action_list": [
    {
      "name": "catchcube",
      "state": "run",
      "info": "",
      "st_time": 1646472361.6056051,
      "fin_time": 0,
      "result": 0
    }
  ]
}
```

Рис. 8. Ответ робота на запрос информации о состоянии

Видеопоток с установленных камер, транслируется с помощью механизма *websocket*, изображение с телекамеры можно просматривать в браузере (рис. 9).

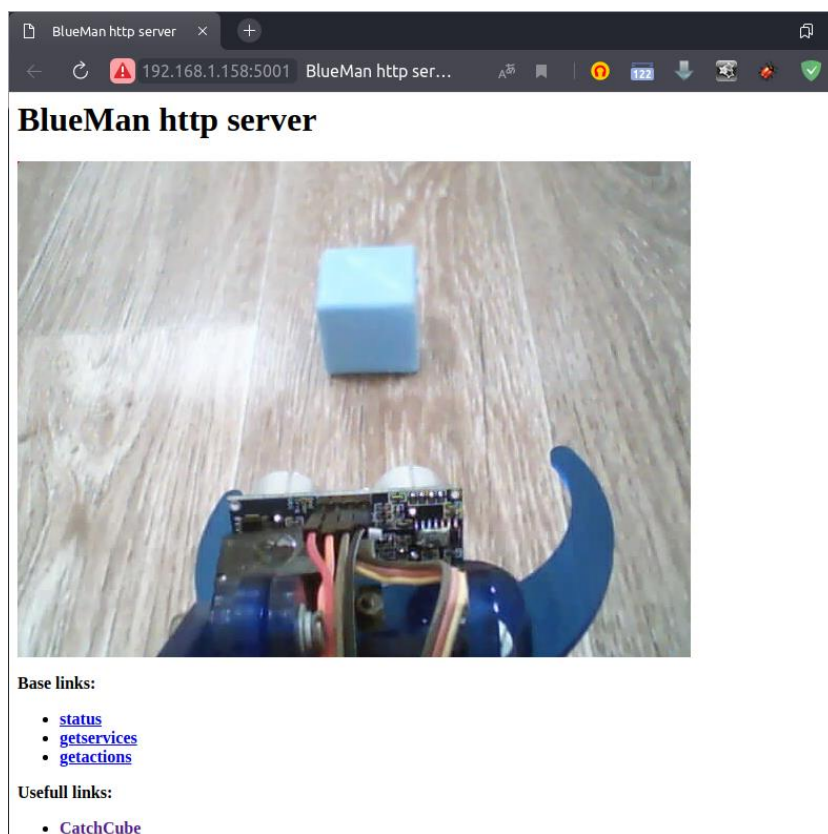


Рис. 9. Видеопоток с камеры, установленной на манипуляторе

На данный момент анализ изображений происходит непосредственно на одноплатном компьютере *Raspberry PI*, управляющем манипулятором по *COM* порту.

3.2. Пример обмена действиями

Один из используемых на полигоне роботов – мобильный манипулятор. Это гусеничный робот с установленным на передней части 3-х осевым манипулятором. Особенностью робота является его разбивка на несколько автономных модулей, которые взаимодействуют друг с другом в процессе выполнения задач. Модули установлены на два одноплатных компьютера *Raspberry PI* (рис. 10).

Первый модуль отвечает за движение. Под него выделен отдельный компьютер, который считывает данные с системы наблюдения Лидар и энкодеров, управляя приводами гусениц. Второй модуль считывает информацию с камеры, установленной под захватом, и выполняет функцию обнаружения объекта. Если объект в кадре обнаружен, то вычисляется расстояние до него. Третий модуль управляет захватом и вычисляет положение приводов на основании расстояния до объекта с помощью предварительно настроенных баз знаний. Задача мобильного манипулятора подъехать в зону расположения объекта, обнаружить его и захватить. Для выполнения этой задачи задействованы все три модуля, описанные выше. Дополнительная сложность задачи заключается в том, что манипулятор жёстко зафиксирован и для захвата объекта робот должен подъехать так, чтобы объект был точно по центральной оси перед роботом.

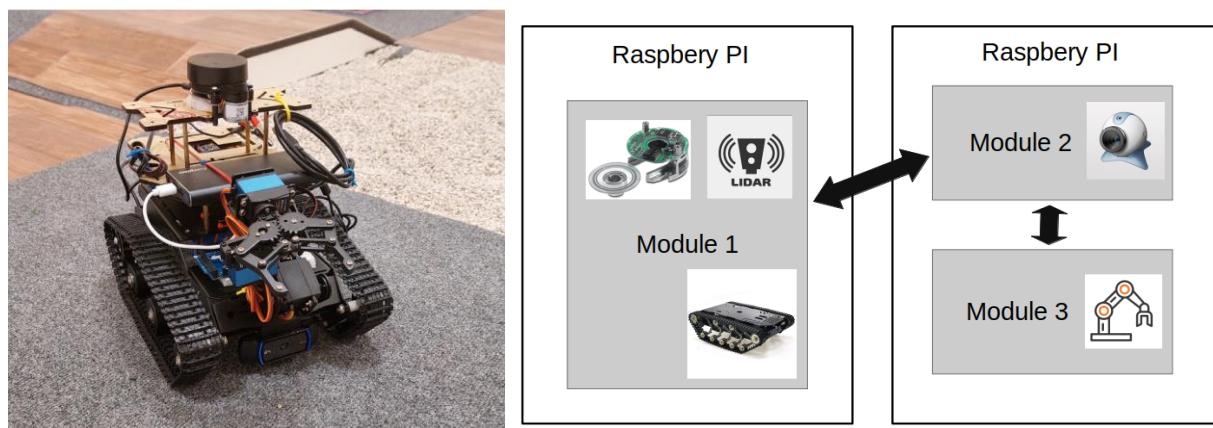


Рис. 10. Общий вид мобильного манипулятора и схема взаимодействия модулей

Первый модуль обеспечивает перемещение манипулятора к зоне расположения объекта, далее второй модуль выполняет обнаружение объекта и отправляет команды движения первому модулю для точного наведения манипулятора на объект. Команды на движение содержат направление и угол поворота, а также величину смещения вперёд или назад. Когда объект попадает в зону досягаемости захвата, второй модуль направляет команду на захват третьему модулю.

3.3. Пример применения передачи знаний в задаче локальной оптимизации динамически неустойчивого объекта

Конструирование баз знаний (БЗ) нечётких систем управления осуществляется с помощью разрабатываемого в МЛИТ ОИЯИ Оптимизатора Баз Знаний (ОБЗ) [19,28]. Разработанный интеллектуальный инструментариум позволил проектировать робастные БЗ на основе решения одной из алгоритмически трудно решаемых задач теории искусственного интеллекта – извлечения, обработки и формирования объективных знаний без использования экспертных оценок. В данном оптимизаторе используются три ГА, которые позволяют спроектировать оптимальную структуру нечёткого регулятора (вид и число физических процессов, их параметры, а также число самих правил нечёткого вывода), аппроксимирующую обучающий сигнал с требуемой ошибкой. При этом автоматически проектируется оптимальная структура нечёткой нейронной сети и формируется модель универсального аппроксиматора в виде нечёткого регулятора с конечным числом продукционных правил в БЗ.

Типовая модель, на которой можно наглядно продемонстрировать работу данного класса алгоритмов – задача обратного (перевернутого) маятника (рис. 11) [29].

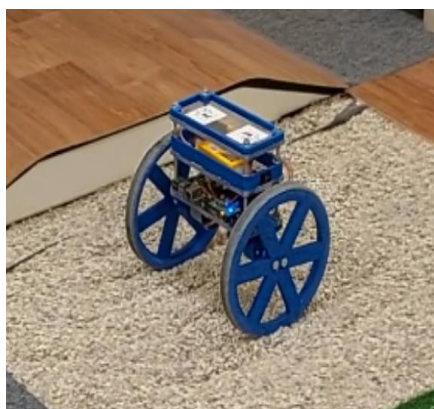


Рис. 11. Вид робототехнической системы типа обратный маятник

В данной работе рассмотрено удалённое соединение с объектом управления для настройки БЗ нечёткого регулятора (НР) на основе ОБЗ с применением технологии мягких вычислений. Такая возможность позволяет проектировать ИСУ без использования системы стохастического моделирования, что даёт преимущество при проектировании нечётких систем управления для сложных и слабо формализованных ОУ в непредвиденных ситуациях управления.

Кроме того, дистанционная настройка БЗ даёт возможность максимально адаптировать нечёткую систему управления для конкретной (непредвиденной) ситуации управления не зависимо от времени и места нахождения объекта управления. Такого рода самоорганизующиеся ИСУ с дистанционным проектированием БЗ важны для ликвидации последствий аварий на АЭС, при разборе завалов при землетрясениях, крушении поездов, для работы в загрязненной и радиоактивной окружающей среде и т.д.

Связь с объектом управления осуществляется по WiFi с использованием механизма *WebSocket*. Схема взаимодействия робототехнической системы и компьютера с ОБЗ [17] представлена на рис. 12.



Рис. 12. Схема соединения настраиваемого устройства и ОБЗ

Система управления считывает показания датчиков и отправляет их на компьютер для последующей обработки. Приняв входные значения, ОБЗ оценивает предыдущее решение (БЗ нечёткого регулятора) и осуществляет нечёткий вывод для проверки следующего решения (БЗ нечёткого регулятора). Результат нечёткого вывода отправляется на удалённое устройство. После этого, система управления, обработав входные значения, вырабатывает управляющее воздействие.

Синхронизация ОБЗ и системы управления осуществляется на стороне удалённого устройства. С этой целью была разработана специальная программа (прошивка), реализующая алгоритм, представленный на рис. 13.

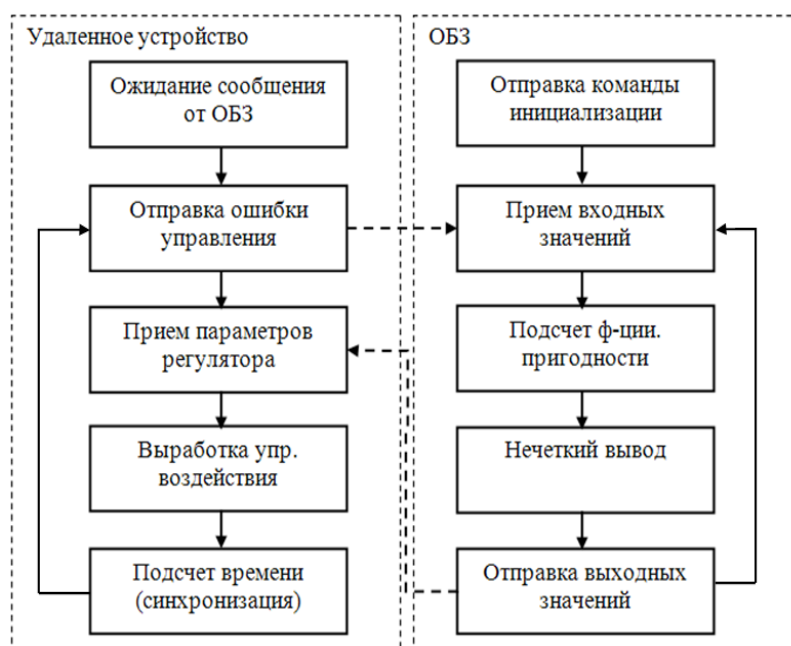


Рис. 13. Алгоритм работы удалённой настройки

Устройство открывает сокет, через который передаются числа с плавающей точкой в символьном виде. Подключение к ОБЗ осуществляется через разработанный плагин. Технология беспроводной настройки также была применена робототехнической системе мобильного манипулятора. Здесь интеллектуальная система управления используется для быстрого и точного перемещения к объекту распознавания, а ошибкой является отклонение от прямолинейного курса в условных единицах.

Заключение

- Развитие технологий проектирования взаимодействия активных агентов в многоагентных системах представляет как практический, так и научный интерес. Для возможности реализации таких систем в реальной среде необходимо, чтобы каждый активный агент имел встроенную интеллектуальную систему управления, так как при взаимодействии систем возникает множество нештатных и непредвиденных ситуаций, которые сильно влияют на результат такого взаимодействия.
- Механизмы, обеспечивающие взаимодействие, не должны выступать ограничением при работе интеллектуальных систем. После изучения всех возможных задач был составлен перечень механизмов взаимодействия и их характеристик.
- Применимость выбранных механизмов была продемонстрирована на задачах обмена информацией, знаниями и действиями между автономными роботами.
- Возможность использования дополнительного программного инструментария и интеллектуальной дистанционной настройки показала свою применимость для интеллектуальных систем.
- Разработанные варианты взаимодействия показывают возможности применения таких систем в широком спектре задач, таких как автоматизация складов и производств, автоматические заведения общественного питания, ликвидация последствий чрезвычайных происшествий и т.п.

Список источников

1. Cooperative robotic networks for underwater surveillance: An overview / G. Ferri [et al.] // IET Radar Sonar Navig. 2017. Vol. 11. P. 1740-1761. URL: <https://ietresearch.onlinelibrary.wiley.com/doi/10.1049/iet-rsn.2017.0074>.
2. A cooperative approach for multi-robot area exploration / J. Yuan [et al.] // In Proc. of the 2010 IEEE/RSJ Intern. Confer. on Intelligent Robots and Systems, Taipei, Taiwan, 18–22 October 2010. - P. 1390–1395. URL: <https://www.semanticscholar.org/paper/A-cooperative-approach-for-multi-robot-area-Yuan-Huang/513abbc38f347bfa8ae91f5b5678c3c86eb5af9d>.
3. Cooperative semi-autonomous robotic network for search and rescue operations / G. Herman [et al.] // Intern. Conf. on Universal Access in Human-Computer Interaction. Springer: Berlin/Heidelberg, Germany, 2014. pp. 636–647. DOI: https://doi.org/10.1007/978-3-319-07446-7_61.
4. Machine-to-Machine Communication for Device Identification and Classification in Secure Telerobotics Surgery / P. Meghana [et al.] // Security and Communication Networks, 2021. Vol. 2021. Article ID: 5287514. DOI: <https://doi.org/10.1155/2021/5287514>.
5. Optimal caching scheme in D2D networks with multiple robot helpers / Y. Lin [et al.] // Computer Communications. 2022. Vol. 181. P. 132-142. URL: <https://doi.org/10.1016/j.comcom.2021.09.027>.
6. Cooperative Friendly Jamming Techniques for Drone-Based Mobile Secure Zone / Jeon Ga-Hye [et al.] // Sensors, 2022. Vol. 22. №3. URL: <https://www.mdpi.com/1424-8220/22/3/865>.
7. Arce L. IoT in Automotive Industry-Connected cars // Politecnico di Milano, Spring April 17, 2016.
8. How the Internet of things (IoT) is revolutionizing healthcare // Plasma. 17.08.2015. URL: <http://www.plasmacomp.com/blogs/how-internet-of-thing>.
9. Zou J. A data transmission method between the ROS-based robot and the industrial controller // Journal of Physics: Conference Series, 2021. Vol. 2029. DOI: <http://dx.doi.org/10.1088/1742-6596/2029/1/012003>.

10. ROS:Home // Open Robotics, 2021. [Электронный ресурс]. URL: <https://www.ros.org/>.
11. Shi L., Marcano N. J. H., Jacobsen R. H. A review on communication protocols for autonomous unmanned aerial vehicles for inspection application // *Microprocessors and Microsystems*. 2021. Vol. 86. P. 104340. DOI: <https://doi.org/10.1016/j.micpro.2021.104340>.
12. A Web-oriented Architecture for Deploying Multiple Unmanned Vehicles as a Service / C.N. Au [et al.] // *The Intern. J. on Marine Navigation and Safety of Sea Transportation*, 2021. Vol.15. № 1. DOI: <http://dx.doi.org/10.12716/1001.15.01.15>.
13. TANGO Controls // Tango Control, 2015. [Электронный ресурс]. URL: <http://www.tango-controls.org> (дата обращения: 20.12.2021)
14. TANGO STANDARD SOFTWARE TO CONTROL THE NUCLOTRON BEAM SLOW EXTRACTION / V. Andreev [и др.] // *Письма в ЭЧАЯ*. 2016. Т. 13. №5(203). С. 951-956.
15. Andreev V. et al. NUCLOTRON CONTROL SYSTEM, Laboratory of High Energies Joint Institute for Nuclear Research, Dubna, Russia. URL: <http://nucloweb.jinr.ru/nucloserv/text/august2003/html/3p041.htm>.
16. Gorbachev E. V. et al. Nuclotron and NICA control system development status JINR, Dubna, Russia Proc. of ICALEPCS 2015, Melbourne, Australia. - P. 437-440.
17. Ульянов С. В., Решетников А. Г. Синергетика информационно-когнитивного взаимодействия в интеллектуальных робототехнических системах с дистанционным обменом знаниями // *Software & Systems*. - 2017. - Т. 30. - No 4. - С. 593-600.
18. Intelligent control of mobile robot with redundant manipulator & stereovision: quantum / soft computing toolkit / K.V. Koshelev, A.V. Nikolaeva, A.G. Reshetnikov, S.V. Ulyanov // *Artificial Intelligence Advances*. 2020. Vol. 2. N o 2. DOI: <https://doi.org/10.30564/aia.v2i2.1440>.
19. Ulyanov S.V. Soft computing optimizer of intelligent control system structures. - US Patent No US 7,219,087 B2. Date of Patent: May 15, 2007.
20. Зайцев А. А., Курейчик В. В., Полупанов А. А. Обзор эволюционных методов оптимизации на основе роевого интеллекта // *Изв. ЮФУ: Технические науки*. 2010. № 12. С. 7-12.
21. Cazenille L., Bredeche N., Halloy J. Automated optimization of multi-level modes of collective behavior in a mixed society of animals and robots // *arXiv preprint*. URL: arxiv.org/pdf/1602.05830v1.pdf.
22. Интеллектуальная когнитивная робототехника. Том I: Технология мягких вычислений в интеллектуальных когнитивных системах управления — информационно-термодинамический закон / О.Ю. Тятюшкина, А.Г. Решетников, С.В. Ульянов, В.С. Ульянов, - М.: КУРС. 2022. (на англ.)
23. A taxonomy for multi-agent robotics / G. Dudek, M. Jenkin, E. Milios, D. Wilkes // *Autonomous Robots*. 1996. Vol. 3. No 4. P. 375-397.
24. Ota J. Multi-agent robot systems as distributed autonomous systems // *Advanced Engineering Informatics* – 2006. Vol. 20. No 1. P. 59–70.
25. Clients Libraries and Developer Tools. // [VMware](https://www.rabbitmq.com/devtools.html), Inc. 2007-2022: URL: <https://www.rabbitmq.com/devtools.html> (дата обращения: 20.12.2021)
26. Введение в ROS: Работа с Service . [Электронный ресурс]. URL: <http://docs.voltbro.ru/starting-ros/messaging/rabota-s-service.html> (дата обращения: 20.12.2021)
27. ZigBee : [Электронный ресурс]. URL: <http://machinepedia.org/index.php?title=ZigBee> (дата обращения: 20.12.2021)
28. Ульянов С. В., Николаева А. В., Решетников А. Г. Интеллектуальные системы управления в непредвиденных ситуациях. Оптимизатор баз знаний на мягких вычислениях. LAPLAMBERT Acad. Publ., OmniScriptum GmbH&Co. KG, 2013.
29. Ульянов С. В., Решетников А. Г., Решетников Г. П. Технологии интеллектуальных вычислений: Квантовые вычисления и программирование в самоорганизующихся интеллектуальных системах управления. Дубна: ОИЯИ, 2015. 246 с. ISBN 978-5-9530-0422-0.

Приложение 1. Описание REST протокола взаимодействия

1. Запрос статуса

Запрос статуса может осуществляться по адресу: `/status[?param=value[¶m2=value2]&]`.

В первой версии протокола описан только один параметр – *action* (например, *action="name[,name2[...]]"*). Параметр указывает, что нужно вернуть статус конкретных *action*, имена которых могут быть перечислены через запятую.

Ответ на запрос статуса будет выглядеть следующим образом:

```
{
  "name": "str - device name",
  "rc": "int - request result code",
  "info": "str - text interpretation of return code",
  "state": "str - global device status: init | run | fail",
  "action_list": [
    {
      "name": "str - action_name",
      "state": "str - action status: none | init | run | success | fail",
      "info": "str - any information",
      "st_time": "int - timestamp of last start",
      "fin_time": "int - timestamp of finish",
      "result": "int - action result code"
    },
    ...
  ]
}
```

Значения параметра *rc*:

- 0 – запрос выполнен успешно;
- -1 – неверные параметры;
- -2 – Action с таким именем не найден.

action_list содержит в себе массив статусов *action*. По умолчанию (запрос без параметров), возвращается массив статусов активных на данный момент *action*. Если были указаны имена с помощью параметра *action*, то *action_list* содержит статусы указанных *action*. При этом не важно был ли этот конкретный *action* запущен на момент запроса.

2. Запрос логов

Запрос логов осуществляется по адресу: `/history?type=value[&name=value2[&n=20]]&`.

Протокол относительно запроса логов будет доработан позднее.

Доступные параметры:

- *type* – тип запрашиваемых логов. Может быть: *action, system*;
- *name* – используется с типом *action*, позволяет указать конкретные имена *action*, историю статусов которых нужно вернуть;
- *n* – верхнее ограничение на количество возвращаемых элементов.

Ответ на запрос логов имеет вид:

```
{
  "rc": "int - request result code",
  "info": "str - text interpretation of return code"
  "data": [
    ...
  ]
}
```

Если значение *type* – *action*, то поле *data* содержит массив объектов типа статус *action*, для всех *action*, имена которых были указаны в параметре *name*, или для всех *action*, если параметр *name* не задан.

Если значение *type* – *system*, то *data* содержит массив строк.

3. Запуск action

Запуск *action* можно осуществить с помощью запроса по адресу: `/action?name=action_name[¶m=value[...]]&`.

Наличие параметров в данном случае зависит от конкретного Action. Все параметры в формате GET, например: `/action?name=act_name¶m1=value1¶m2=value2&`.

Ответ на запрос запуска Action выглядит следующим образом:

```
{
  "name": "str - action name",
  "rc": "int - request result code",
  "info": "str - text interpretation of return code"
}
```

Предопределённые значения *rc*:

- 0 – Action запущен.
- -1 – Action с таким именем не найден.
- -2 – Action с таким именем не запустился.
- -3 – Action с таким именем уже запущен.

4. Остановка action

Остановка *action* осуществляется запросом вида: `/reset[?action=action_names&]`.

Доступные параметры:

- *action* – например, `action="name[,name2[...]]"`. Параметр указывает что нужно остановить конкретные *action*. Имена *action* перечислены через запятую. Если параметр не указан, будут завершены все активные *action*.

Ответ на запрос остановки *action* будет выглядеть следующим образом:

```
{
  "name": "reset",
  "rc": "int - request result code",
  "info": "str - text interpretation of return code"
  "data": {
```

```

    "action_name1": "int - reset_status"
    ...
}
}

```

Поле *data* в ответе будет содержать словарь, в котором ключ – имя *action*, значение – код запуска операции остановки. Если были заданы имена конкретных *action* через параметр *action*, то словарь будет содержать столько значений, сколько *action* было указано.

В противном случае в словаре будет столько значений, сколько активных *action* было на момент запроса.

Предопределённые значения *rc*:

- 0 – команда выполнена успешно
- -1 – не все перечисленные *action* удалось найти

Предопределённые значения *reset_status*:

- 0 – операция остановки успешно запущена
- -1 – произошла ошибка при попытке остановки
- -2 – *action* не запущен

5. Запрос информации из опции **service**

Запрос осуществляется по адресу: `/service?name=service_name[¶m=value[...]]&`.

Как и в случае с *Action* параметры зависят от конкретного *Service*.

Рассматривался вариант сделать так, чтобы *service_name* мог содержать несколько имён, разделённых между собой запятыми. С одной стороны такая возможность позволила бы за один запрос получить всю необходимую информацию, с другой, такой функционал приводит к появлению дополнительных сложностей и противоречий. В первую очередь беспокоят возможные коллизии в именах параметров *Service*. Чтобы не усложнять протокол на данном этапе было принято решение, что *service_name* будет содержать имя только одного сервиса.

Ответ на запрос информации из *Service* выглядит следующим образом:

```

{
  "name": "str - service name",
  "rc": "int - request result code",
  "info": "str - text interpretation of return code"
  "data": {
    ...
  }
}

```

Предопределённые значения *rc*:

- 0 – запрос успешный
- -1 – *Service* с таким именем не найден
- -2 – *Service* с таким именем не ответил

Помимо структуры запроса было утверждено, что каждое устройство должно иметь два обязательных сервиса:

- *getactions* – возвращает список доступных *Action*
- *getservices* – возвращает список доступных *Service*

Результат вызова доступных *getactions* должен в поле *data* содержать структуру с полями:

- *name* – имя *action*;
- *description* – текстовое описание *action*, что он делает;
- *parameters* – структура типа словарь со списком принимаемых параметров, где ключ – имя параметра, значение – описание параметра;
- *statuses* – структура типа словарь содержащая список статусов с описание, где ключ – имя статуса, значение – описание данного статуса.

Результат вызова *getservices* аналогичен результату вызова *getactions*, за исключением отсутствующего поля *statuses*.