УДК 004.896 **DOI** 10.15622/sp.57.6

В.П. Андреев, П.Ф. Плетенев

МЕТОД ИНФОРМАЦИОННОГО ВЗАИМОДЕЙСТВИЯ ДЛЯ СИСТЕМ РАСПРЕДЕЛЕННОГО УПРАВЛЕНИЯ В РОБОТАХ С МОДУЛЬНОЙ АРХИТЕКТУРОЙ

Андреев В.П., Плетенев П.Ф. **Метод информационного взаимодействия для систем распределенного управления в роботах с модульной архитектурой.**

Аннотация. В статье приведено решение проблемы информационного взаимодействия между встраиваемыми вычислительными устройствами при реализации *распределенного управления* в информационно-измерительной и управляющей системе (ИИУС) роботов с модульной архитектурой. Распределенное управление реализуется за счет проектирования каждого модуля как устройства со своей собственной ИИУС, содержащей все необходимые для выполнения своего функционала компоненты, включая вычислительные устройства. Вследствие такой функциональной завершенности модулей происходит *распараллеливание вычислительного процесса* функционирования робота как единой мехатронной системы. В результате существенно снижаются требования к мощности вычислительных устройств ИИУС модулей, в качестве которых оказывается возможным использовать недорогие микроконтроллеры и одноплатные ЭВМ — встраиваемые вычислительные устройства.

Предложена сетевая организация структуры ИИУС робота, что позволило перенести свойство *реконфигурируемости* сети на структуру модульного робота. Анализ различных топологий сети показал, что топология типа «звезда» имеет ряд преимуществ по сравнению с топологией типа «шина» для применения в гетерогенных модульных роботах.

Показано, что использование Robot Operating System (ROS) для реализации информационного взаимодействия между встраиваемыми вычислительными устройствами либо невозможно, либо существенно затруднено. Предложена спецификация, предназначенная для создания соответствующих программных интерфейсов и языка межмодульного взаимодействия, обеспечивающих включение модулей сторонних производителей в режиме «plug and play». Спецификация основана на принципах ROS, но позволяет реализовать ПО на встраиваемых вычислительных устройствах. На основе многокритериальной оптимизации по Парето получены рекомендации для выбора соответствующих аппаратно-программных средств.

Работоспособность предложенного решения была доказана в ходе экспериментов на установке, состав которой приближен к условиям работы ИИУС гетерогенного модульного робота. Эксперименты показали, что совместная работа программной и аппаратной частей удовлетворяет всем обозначенным требованиям и применима для передачи сообщений исполнительного уровня с частотой до 100 Гц при любой нагрузке на сеть.

Ключевые слова: модульный робот, гетерогенный робот, мобильный робот, реконфигурируемость, система управления, распределенное управление.

1. Введение. Мобильные роботы (МР), предназначенные для работы в условиях, когда заранее невозможно определить вид предполагаемой работы, должны допускать быстрое изменение своей структуры непосредственно на месте проведения работ и во время самих работ. В работе А.В. Лопоты и Е.И. Юревича [1] отмечено, что «Именно в экстремальных ситуациях различных катастроф, аварий и активных противодействий имеет место предельная априорная

неопределенность как условий предстоящих работ, так и самого их перечня. Что делает особо актуальной возможность компоновать состав роботов непосредственно на месте работы и корректировать его в ходе самих работ».

Аналогичные требования предъявляются и к роботам, предназначенным для космических исследований. Так, в обзоре [2] робототехники, необходимой для создания лунной базы, приводится перечисление роботов различного назначения: от роботов-бульдозеров до роботов-строителей и роботов-ремонтников. Такое большое разнообразие объясняется тем, что «у лунной робототехники есть важная особенность. Обычно под роботом понимается машина, способная функционировать в заранее «недоопределенной» среде …».

Такой широкий спектр роботов экономически невыгодно завозить на Луну или в зону с экстремальными условиями по отдельности. Достаточно создать набор функциональных модулей и механизм автоматической реконфигурации (средствами самих модулей робота или внешней робототехнической или роботизированной системы). Тогда появляется возможность на месте автоматически собирать из этого набора роботы необходимого назначения.

Следовательно, для выполнения таких работ должны использоваться роботы с переменной структурой, то есть *реконфигурируемые* модульные мобильные роботы.

К реконфигурируемым модульным мобильным роботам относятся *гомогенные* и *гетерогенные* модульные роботы.

Гомогенные роботы состоят из множества одинаковых модулей. Каждый модуль содержит в себе все необходимые для работы компоненты — датчики, двигатели, движители, аккумуляторы, системы управления и так далее. По сути, гомогенный модульный робот — это суперпозиция множества отдельных одинаковых роботов. В качестве примера можно привести модульный робот M-TRAN [3]. Несмотря на простоту выполняемых каждым из модулей движений, множество модулей, построившись в некую структуру, способны выполнять сложные согласованные движения. M-TRAN, как и другие гомогенные роботы, например, ATRON [4] и TRANSMOTE [5], имеют общую особенность — несмотря на то, что каждый из модулей независим, управление работой всех модулей осуществляется модулем-супервизором на *исполнительном уровне*. В результате оказывается необходимым разрабатывать дополнительное достаточно сложное программное обеспечение (ПО), которое переводит команды модуля-супервизора в последовательности исполнительных команд для каждого из модулей-исполнителей и обеспечивает их согласованную работу. Это, в свою очередь, приводит к необходимости устанавливать на модуль-супервизор «мощный» компьютер.

Гетерогенные роботы состоят из конечного набора модулей различной функциональности, которые соединяются друг с другом через унифицированные механические, электрические и программные интерфейсы. В отличие от гомогенных, модули гетерогенных роботов, как правило, не могут нормально функционировать друг без друга. Первым из известных МР, который является гетерогенным и полностью модульным, подразумевающим единство интерфейсов, можно считать робот CEBOT [6]. К классу гетерогенных также относятся роботы SMART [7], Thor [8] и модульный робот для космических исследований [9]. Каждый из данных роботов — это синергетическое объединение различных модулей и субмодулей (отдельных механических компонент модулей) в единую мехатронную систему. Робот Thor является прямым наследником гомогенного модульного робота ODIN [10], включая системы взаимодействия — в обоих роботах используется шина RS-485. Робот SMART несколько отличается — один из его модулей, подключенный по беспроводному каналу сети Bluetooth к супервизору, становится ведущим на шине CAN, через которую он управляет другими модулями, транслируя полученные от супервизора команды, а полученные от остальных модулей данные отсылает через радиоканал супервизору. Иными словами, управление работой всех модулей осуществляется модулем-супервизором также на *исполнительном уровне*, что, как и в случае гомогенных роботов, приводит к необходимости использовать центральный вычислитель большой производительности.

С позиции создания мобильных роботов с переменной структурой, то есть *реконфигурируемых* модульных роботов, ориентированных на использование в случае предельной априорной неопределенности условий и видов предстоящих работ, считаем, что на сегодняшний день *преимуществами обладают гетерогенные роботы*. Также полагаем, что такие роботы должны обладать свойством автоматической перестройки элементов информационно-измерительной и управляющей системы (ИИУС) робота под новые состав и конфигурацию.

2. Система распределенного управления для гетерогенного робота. Одна из основных трудностей создания реконфигурируемых систем заключается в разработке систем управления, способных работать в режиме «plug and play». В таких роботах должна осуществляться автоматическая реорганизация общей структуры системы управления в соответствии с быстро меняющимся составом и конфигурацией роботизированного устройства. Данное требование приводит к значительному увеличению сложности системы скоординированного управления модулями гетерогенного робота как единой мехатронной системы.

Упростить такую систему управления позволяет использование принципа полной функциональности в конструкции и программном обеспечении модулей МР, представляющих собой мехатронные устройства. *Полная функциональность мехатронного устройства — это способность выполнять свою целевую функцию, используя только собственные средства для выполнения команд от внешней системы управления* [11].

Гетерогенный модульный робот — это конструкция из отдельных функциональных устройств — модулей. Каждый модуль в таком роботе — это мехатронное или электронное устройство. Такие устройства должны иметь универсальные механические, электрические и программные интерфейсы. Лучшей взаимозаменяемости модулей и гибкости структуры МР можно достичь, если каждый из модулей разработан для выполнения одной специфической функции и выполняет ее максимально хорошо, то есть является *полнофункциональным устройством*.

Полная функциональность модулей робота достигается путем проектирования каждого модуля как устройства со своей собственной ИИУС, содержащей все необходимые компоненты, включая вычислительные устройства. Такая функциональная завершенность является *основным отличием* нашего решения от рассмотренных гетерогенных роботов, где модуль-супервизор управляет работой всех модулей-исполнителей на исполнительном уровне. В нашем решении модуль-супервизор формирует лишь цель управления, которую передает в модуль-исполнитель, и проверяет только результат ее достижения исполнительным модулем, но не управляет процессом выполнения задачи этим модулем. Как следствие такого подхода, реализуется *распределенное управление*, что существенно снижает требование к вычислительной мощности компьютеров как модуля-супервизора, так и модулей-исполнителей. Такое распределенное управление позволяет распараллеливать вычислительный процесс реализации целевой функции робота за счет разделения процесса на функциональные подзадачи и распределения их между вычислительными устройствами ИИУС модулей. При этом в качестве таких вычислительных устройств должны выступать относительно простые и дешевые малогабаритные микроконтроллеры и/или одноплатные ЭВМ — встраиваемые вычислительные устройства.

Такая распределенная система управления модульного робота нуждается в специфическом механизме информационного взаимодействия, в котором каждый полнофункциональный модуль должен иметь возможность обмениваться информацией (включая

команды управления) *непосредственно с любым другим модулем*. Данное условие обеспечивает распараллеливание и независимость информационных потоков в коммуникационном канале. Такой механизм (framework) уже существует и называется Robot Operating System (ROS) [12]. Эта система считается одной из наиболее развитых для использования в мобильной робототехнике. Однако ее невозможно применить для работы на микроконтроллерах, а использование на многих одноплатных ЭВМ имеет существенные ограничения.

3. Метод информационного межмодульного взаимодействия. Предлагаемый метод предназначен для организации межмодульного информационного взаимодействия в системе распределенного управления гетерогенных модульных роботов с функционально-модульной структурой, работа которой основана на следующих принципах [13]:

1. Информационно-измерительная и управляющая система робота строится как локальная вычислительная сеть, узлами которой являются встраиваемые вычислительные устройства ИИУС модулей.
2. Каждый модуль является полнофункциональным электронным или мехатронным устройством со своей собственной ИИУС.
3. Включение любых мехатронных или электронных устройств в сетевую структуру робота осуществляется с помощью специального программного обеспечения — драйверов.

Сетевая структура ИИУС модульного робота придает роботу свойство *реконфигурируемости* и масштабируемости за счет использования хорошо проработанных сетевых протоколов и развитых библиотек. *Функциональная завершенность модулей* означает конструктивную и функциональную независимость, что позволяет использовать модули сторонних производителей, идентичные по своим функциям. *«Концепция драйверов»* заключается в том, что для каждого электронного или мехатронного устройства создается унифицированный сетевой управляющий протокол (программный интерфейс) на базе существующих низкоуровневых программных интерфейсов взаимодействия. Тогда производитель может сравнительно просто интегрировать свое устройство в ИИУС модульного робота, создав лишь соответствующий драйвер, реализующий детерминированные протоколом программные инструкции управления и протоколы сетевого взаимодействия. При появлении в сети нового устройства ПО системы управления каждого из модулей в процессе взаимодействия с драйвером нового модуля автоматически включает это устройство в общую сеть — реализуется режим «plug and play».

Для реализации предлагаемого метода межмодульного информационного взаимодействия необходимо было решить следующие задачи:

– выбрать тип и топологию коммуникационной сети;

– создать спецификацию — документ, определяющий требования и рекомендации к реализации механизма взаимодействия;

– определить требования к аппаратно-программному обеспечению модулей;

– разработать систему команд для каждого модуля МР.

4. Коммуникационные сети: выбор типа и топологии. В современных модульных роботах, как гомогенных, так и гетерогенных, в связи с миниатюризацией самих модулей и минимизацией количества контактов на механических интерфейсах используются различные последовательные шины:

– I2C на модулях ЦНИИ РТК;

– CAN — в роботах M-TRAN, SMART;

– IrDA (по сути RS-232) — в роботе ATRON;

– RS-485 — в роботах Thor и Odin.

Последовательная шина, в широком смысле, используется и на роботе TRANSMOTE (используется беспроводной интерфейс ZigBee).

Чаще всего коммуникация в известных модульных роботах строится на топологии типа «шина» с локальными ответвлениями (рисунок 1*а*), реже — по топологии типа «звезда» (рисунок 1*б*). Топология «шина» наиболее полно отражает структуру модульного робота — каждый модуль имеет нескольких «соседей», через которых он взаимодействует со всей остальной системой. Недостатком таких шин является их низкая пропускная способность. Но в перечисленных конструкциях пропускной способности «шины» достаточно, поскольку у таких роботов есть, по сути, два варианта работы:

– выполнять набор простых правил, которые разработчики поместили в ПЗУ микроконтроллера в каждом из модулей;

– выполнять команды внешнего компьютера, подаваемые модулям через какой-либо один модуль, к шине которого (а значит, и к общей шине соединенных модулей) подключен компьютер.

Это означает, что роль системы межмодульного взаимодействия сводится только к наиболее быстрой и эффективной передаче команд исполнительного или тактического уровней и показаний датчиков. Тогда информационный поток в процессе межмодульного взаимодействия невелик, поскольку на каждом модуле либо нет датчиков, либо их установлено не так много и, фактически, необходимо передавать лишь команды *исполнительного уровня*.

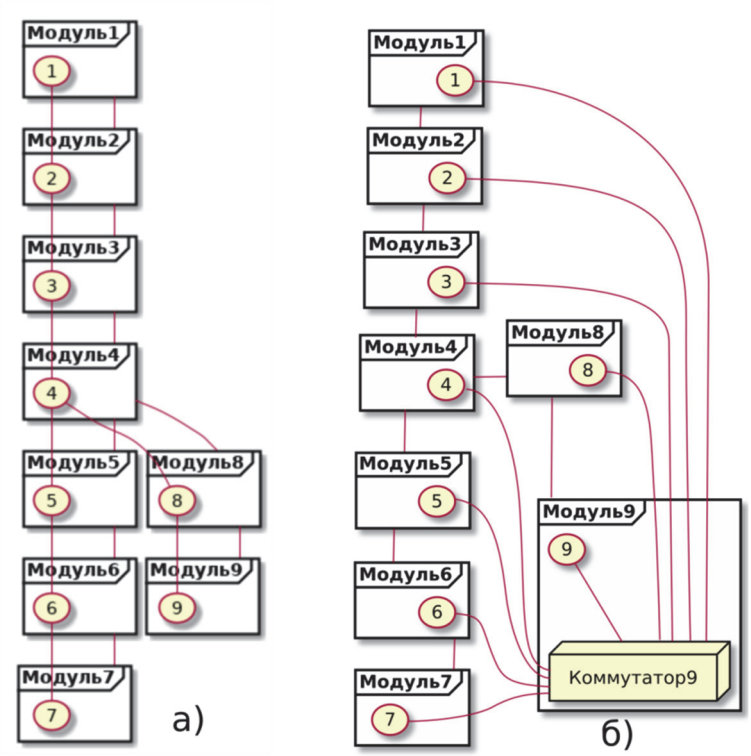


Рис. 1. Топологии сети Ethernet: а) «шина»; б) «звезда»

Многие компании, разрабатывающие средства автоматизации, в настоящее время переходят к использованию коммуникационных сетей, основанных на стандарте Ethernet или его модификациях, ориентированных для работы с детерминированными временами доставки сообщений. Такая популярность Ethernet в среде промышленных сетей является следствием универсальности и открытости стандарта Ethernet. Примерами доработки стандарта являются EtherCAT [14] и Ethernet Powerlink [15]. Оба этих стандарта переопределяют формат кадра, передаваемого по сети, что позволяет добиться значительно меньших времен задержек — 50-100 наносекунд в случае EtherCAT и единиц микросекунд в случае Ethernet Powerlink.

Сравнение различных существующих сетей приведено в таблице 1.

Таблица 1. Свойства различных сетей

Характеристики	Реальное время	Основные топологии	Макс. скорость, Мбит/с	Механизм борьбы с коллизиями	Служебная инф. и макс. длина сообщения, бит	Распространенность оборудования	Цены на оборудование
RS-485	Жесткое[1]	Шина	10 (на 10 м)	Не определен	80[7] / 2088[10]	Высокая	От низких до средних
CAN	Жесткое	Шина	1 (на 40 м)	CR[4]	44 или 64[8] / 64	Средняя	От средних до высоких
Ethernet	Мягкое	Точка-точка, разные[2]	100/1000 (на 100 м)	CD[4]	464 или 336 / 11536 или 11664[9]	Очень высокая	От низких до средних
EtherCAT	Жесткое	Точка-точка, шина, разные[3]	100 (на 100 м)	Master-Slave[5]	160 / 11888	Низкая	Высокие
Powerlink	Жесткое	Шина	100/1000 (на 100 м)	Master-Slave[6]	128 / 11920	Высокая	От низких до средних

Примечания к таблице 1: (1) — зависит от реализации, стандарт не определяет этот параметр; (2) — топология может быть как «шина», так и «звезда» в зависимости от дополнительного оборудования; (3) — может иметь любую топологию, однако большинство устройств поддерживают топологию «шина»; (4) — CAN является синхронной шиной с типом доступа Collision Resolving (CR, разрешение коллизии), который, в отличие от Collision Detect (CD, обнаружение коллизии) в сетях Ethernet, детерминировано обеспечивает доступ на передачу сообщения; (5) — использует специальный формат кадра, который формирует ведущее устройство, а ведомые читают обращенные к ним части кадра и записывают в кадр свои значения, тем самым достигается контроль занятия шины в каждый момент времени; (6) — часы на всех устройствах синхронизируются, ведущий подает сигнал начала обмена и получает или передает данные ведомым устройствам; (7) — для реализации MODBUS; (8) – для обычного и расширенного формата кадра; (9) — для протоколов TCP и UDP соответственно; (10) — для реализации MODBUS длина сообщений строго фиксирована.

Использование принципа функциональной завершенности модулей снимает необходимость в передаче команд *исполнительного уровня* – передаются только команды верхних (тактического и стратегического) уровней. В этом случае допустимы более «мягкие» требования к времени доставки сообщений и использование интерфейсов с «мягким реальным временем». В нашем случае мы планируем использовать в сенсорной системе робота большое количество разнообразных датчиков и многокамерную систему технического зрения (многопотоковое видео). Следовательно, *межмодульный коммуникационный канал должен быть широкополосным*.

Согласно данным таблицы 1, Ethernet является наиболее скоростным интерфейсом, а широкая распространенность соответствующего оборудования обеспечивает его низкую стоимость. В случае использования EtherCAT или Powerlink все межмодульное взаимодействие обязано происходить через «ведущего» шины, что снижает эффективность прямого межмодульного взаимодействия (не реализуется распараллеливание информационных потоков). Дополнительным препятствием для использования сетей EtherCAT и Powerlink является необходимость создания сложных встраиваемых устройств «с нуля» для реализации систем управления отдельными модулями. В случае Powerlink возможно использование готовых одноплатных компьютеров, однако стоимость таких компьютеров на 1-2 порядка больше стоимости решений на основе микроконтроллеров.

Сеть Ethernet изначально не является шиной — в ней используются соединения типа «один к одному». Возможно преобразовать ее в «шину», установив в узлы сети коммутаторы (switch), что позволит создавать соединения (топологию) типа «один ко многим». Такое решение обеспечивает оба варианта топологий сети: «шина» и «звезда».

На рисунках 1*а* и 1*б* приведены топологии сети «шина» и «звезда», наложенные на топологию модульного робота. Сравнение этих топологий приведено в таблице 2. Как видно из таблицы, обе топологии имеют недостатки — «шина» требует либо использовать большое число коммутаторов, либо встраивать в каждое устройство несколько интерфейсов Ethernet (по аналогии с EtherCAT), а «звезда» заставляет прокладывать значительно большее число проводов. Последнее в нашем случае не имеет существенного значения, поскольку проводная система локальной вычислительной сети располагается в пределах небольших размеров конструкции робота. Поэтому, мы предпочли топологию типа «звезда».

Таблица 2. Свойства различных топологий сети

Характеристики	«Шина»	«Звезда»
Централизованность	Децентрализованная	Централизованная
Количество коммутаторов	n коммутаторов с $s+m_i$ интерфейсами	1 коммутатор с $\sum_{i=1}^{n} m_i$ интерфейсами
Число линий Ethernet, проходящих через каждый модуль	$s+m_i$ проводов от коммутатора до физических интерфейсов	В худшем случае $(\sum_{i=1, i\neq i_c}^{n} m_i) - m_k$ проводов, где i_c — модуль с коммутатором
Дополнительные ограничения	Максимум 4 коммутатора на пути соединения «один к одному» для минимизации коллизий	Нет

Описание переменных, используемых в таблице 2: n — число модулей в модульном мобильном роботе; m_i — число Ethernet-подключений в каждом модуле (чаще всего $m_i = 1$); s — число возможных соседей модуля (чаще всего 2 или 6).

При выборе топологии сети для модульного робота следует, по-видимому, руководствоваться сопоставлением целого ряда параметров: помехоустойчивость, желаемая частота обмена сообщениями, наличие необходимых сетевых протоколов и соответствующих библиотек, программного и аппаратного обеспечения, требования к сборочными и экономическим показателям, ограничения функционирования сети той или иной топологии и тому подобное. Например, для роботов змеевидной конструкции, скорее всего, подойдет шинная топология, а для антропоморфных роботов — «звезда». Но этот вопрос требует подробного изучения и выработки соответствующих рекомендаций, что планируется сделать в дальнейших исследованиях.

5. Спецификация. Для совместимости программного обеспечения модулей робота необходимо создать *спецификацию* — документ, определяющий требования и рекомендации по реализации метода межмодульного взаимодействия. Подобные спецификации уже достаточно давно выпускает организация, создающая стандарты для работы Интернета, такие как, например, стандарт работы сети Ethernet [16]. Созданы также системы для описания жизненного цикла открытой спецификации — от разработки до «взросления» и «устаревания» [17]. Одним из требований к создаваемой спецификации является возможность использования готовых библиотек и устройств. Конечная цель — создание на основе такой спецификации специального ПО — драйверов и языка межмодульного взаимодействия, то есть создание соответствующих программных интерфейсов, обеспечивающих включение модулей сторонних производителей в режиме «plug and play».

Как было отмечено ранее, одним из самых известных методов межмодульного информационного взаимодействия является программный каркас (framework) ROS. Для ROS разработано большое количество средств визуализации, симуляции и отладки. Однако ROS не ориентирован для работы непосредственно на встраиваемых системах (микропроцессорах): необходим как минимум компьютер с полноценной Linux-подобной ОС, для которой существует порт этой системы; для работы с ним необходимо устанавливать на вычислительные устройства модулей довольно сложное и объемное ПО; требуется высокий уровень входа для его использования (для конфигурации, например, используются диалекты языка XML).

Наш предыдущий подход к реализации межмодульного информационного взаимодействия, предложенный в работе [18], основывался на библиотеке zmq_robot [19]. Она построена на основе

библиотеки ZMQ [20] и позволяет создавать как ПО для исполнительных модулей, так и управляющие программы для модуля-супервизора. Однако эта библиотека имеет тот же недостаток, что и ROS — для работы с ней необходимо устанавливать на устройства сложное ПО, которое работает только на полноценных ОС и не работает на встраиваемых системах.

Поэтому предлагается относительно простая система, которая совместима с ROS и при этом реализуема на встраиваемых системах. Под совместимостью с ROS здесь понимается возможность создания специального ПО — «моста» между создаваемой системой и узлами ROS. Это, в свою очередь, позволяет прозрачно передавать сообщения между системами и воспользоваться обширной базой существующих библиотек и систем, разработанных для ROS.

Сущность предлагаемого решения заключается в использовании библиотеки ZMQ, работающей по протоколу TCP и дополненной протоколом UDP с широковещательными сообщениями, который дает возможность реализовать спецификацию для встраиваемых систем. В отличие от ROS, здесь каждый модуль имеет один общий для всех модулей интерфейс, через который предоставляет общую информацию о себе: имя модуля, его описание, функциональные интерфейсы и прочее. Подробное описание спецификации привести в данной статье не представляется возможным; полный текст спецификации можно найти на интернет-сайте [21].

6. Требования к аппаратно-программной реализации системы распределенного управления. Программно-аппаратное обеспечение включает в себя аппаратно-программное средство (АПС) и соответствующее программное обеспечение. АПС — это электронное вычислительное устройство или набор таких устройств, работающих совместно и создающих как сетевой интерфейс, так и систему управления с помощью соответствующего программного обеспечения.

Аппаратно-программное средство для систем управления модулей робота должно иметь в минимальном исполнении встраиваемое вычислительное устройство, способное работать в сети Ethernet по протоколу UDP с производительностью, достаточной для реализации функционального назначения модуля. АПС также должно иметь минимальные массогабаритные параметры и минимально возможную стоимость. За счет распределения вычислений между микропроцессорами модулей система управления модульного робота в целом имеет более высокую производительность по сравнению с системами управления, которые построены на одном вычислителе.

Для выбора АПС, на базе которого можно реализовать предложенный механизм межмодульного информационного взаимодействия, был выполнен анализ доступных моделей АПС различных производителей:

- Arduino Uno + Ethernet Shield,
- Arduino Mega + Ethernet Shield,
- Arduino Due + Ethernet Shield,
- Taijiuino Due Pro R3 + Ethernet PHY,
- Arduino Yun,
- Seeeduino Cloud,
- Arduino Uno + Yun Shield,
- Arduino Mega + Yun Shield,
- Intel Galileo gen 2,
- Intel Edison,
- Raspberry Pi B Plus,
- Orange Pi Zero,
- 86duino ZERO,
- 86duino One.

По перечисленным АПС была проведена дискретная многокритериальная оптимизация с использованием критерия Парето [22]. Все критерии качества АПС оценивались по описаниям и техническим характеристикам рассмотренных устройств. Ниже приведены эти критерии, часть из которых являются объективными, часть — субъективными:

- $K1$: Возможность реализации спецификации межмодульного взаимодействия; критерий является суммой подкритериев:
  - $K1.1$: Наличие реализации библиотеки ZeroMQ для АПС (0 – реализации не существует; 0.5 — реализация не завершена, имеются серьезные недоработки; 1 — реализация стабильна).
  - $K1.2$: Наличие ограничений на число одновременных подключений к интерфейсу (-1 — ограничения есть и они существенны; 0 — информации об ограничениях нет; 1 — ограничений нет).
  - $K1.3$: Наличие ограничений на число одновременно работающих интерфейсов (-1 — ограничения есть и они существенны; 0 – информации об ограничениях нет; 1 — ограничений нет).
- $K2$: Возможность создавать приложения, работающие в жестком реальном времени; критерий является суммой подкритериев:

– $K2.1$: Наличие операционной системы реального времени (0.5 — ОС существует, но сложна в установке/удалении, 1 — ОС существует и легка в установке и использовании).

– $K2.2$: Уровень входа в программирование в реальном времени (-1 — высокий; 0 — средний; 1 — низкий).

– $K3$: Наличие аппаратного ускорения операций над вещественными числами (0 — нет, 0.5 — ускорения нет, но имеется хорошая программная реализация, 1 — есть).

– $K4$: Количество входов и выходов общего назначения (0, 1, 2, …).

– $K5$: Разрядность таймеров для генерации ШИМ-сигнала (0, 1, 2, …; обычно — 8, 16, 32).

– $K6$: Количество каналов генерации ШИМ-сигнала (0, 1, 2, …).

– $K7$: Количество каналов прерываний по уровню (0, 1, 2, …).

– $K8$: Количество полноценных каналов работы с энкодерами (0, 1, 2, …).

– $K9$: Количество каналов АЦП (0, 1, 2, …).

– $K10$: Стоимость АПС в рублях в России; если АПС состоит из нескольких частей, то их стоимость складывается. Так как стоимость АПС необходимо минимизировать, то этот критерий умножается на -1.

– $K11$: Стоимость АПС в рублях при заказе из Китая с учетом доставки; если АПС состоит из нескольких частей, то их стоимость складывается. Так как стоимость АПС необходимо минимизировать, то этот критерий умножается на -1.

Сравнительный анализ показал, что из 14 исследованных оптимальными являются 7:

– Arduino Uno + Ethernet Shield

$K_{1...10} = \{-1.5; 2; 0; 10; 8; 6; 2; 1; 6; -2347; -2347\}$,

– Arduino Mega + Ethernet Shield

$K_{1...10} = \{-1.5; 2; 0; 50; 8; 15; 6; 3; 16; -3547; -3547\}$,

– Taijiuino Due Pro R3

$K_{1...10} = \{0.5; 1; 0,5; 54; 12; 12; 54; 27; 12; -2699; -2699\}$,

– Seeeduino Cloud $K_{1...10} = \{3; 2; 0; 18; 8; 7; 2; 1; 12; -2850; -2850\}$,

– 86duino ZERO $K_{1...10} = \{3; 1; 1; 14; 32; 7; 14; 4; 6; -3690; -3641\}$,

– 86duino One $K_{1...10} = \{3; 1; 1; 45; 32; 11; 45; 4; 7; -5250; -5611\}$,

– Raspberry Pi B+ $K_{1...10} = \{3; -0.5; 1; 27; 0; 0; 0; 0; 0; -3100; -3100\}$.

7. Система команд модулей. Система команд для каждого модуля определяется, в первую очередь, его функционалом. По принципу полной функциональности можно выделить следующие функции модулей: транспортную, энергетическую, информационную, коммуникационную, технологическую, вспомогательную функции и функцию общего управления. В рамках разработанной спецификации к настоящему времени разработаны системы команд для следующих функциональных модулей лабораторного макета гетерогенного мобильного робота (рисунок 2):

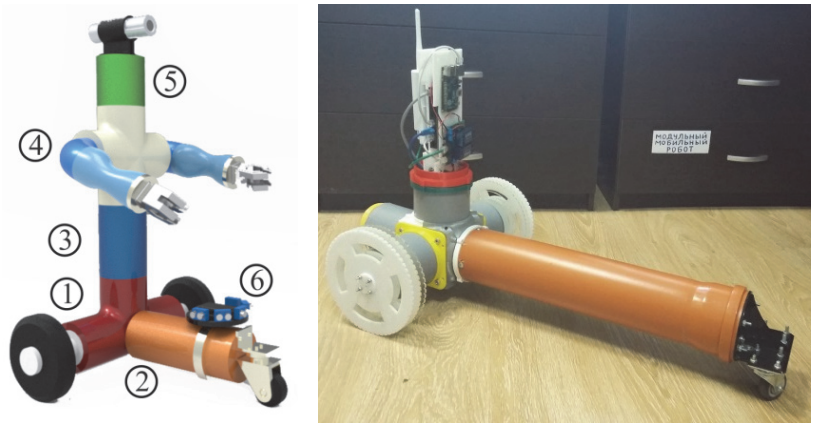


Рис. 2. Компьютерная модель и фотография лабораторного макета гетерогенного модульного мобильного робота: 1 — транспортный модуль, 2 — силовой модуль, 3 — модуль супервизорного управления, 4 — модуль манипуляторов, 5 — сенсорный модуль дальнего радиуса действия, 6 — сенсорный модуль ближнего радиуса действия

1. Транспортный модуль — обеспечивает перемещение робота (транспортная функция).

2. Силовой модуль — обеспечивает энергоснабжение электронных и электромеханических компонент робота и их безопасное включение и отключение (энергетическая функция).

3. Сенсорная система: модули датчиков ближнего и дальнего радиуса действия обеспечивают обнаружение препятствий и объектов манипулирования на разных дистанциях и предоставляют сенсорную информацию другим модулям (информационная функция).

4. Модуль супервизорного управления — обеспечивает управление роботом в целом, синхронизирует информационное взаимодействие всех модулей, выполняет обработку информации и постановку задач, поступающих от внешнего супервизора робота,

формирует и распределяет задания между модулями, являясь, таким образом, для них супервизором, создает и контролирует работу радиоканала (коммуникационная функция и функция общей системы управления). В перспективе коммуникационная функция будет реализована в виде отдельного модуля.

8. Эксперименты. Работоспособность предложенного сетевого и аппаратно-программного решения, предназначенного для реализации метода межмодульного информационного взаимодействия, была проверена в ходе поставленных экспериментов. В программной части использована библиотека ZMQ, дополненная протоколом UDP с широковещательными сообщениями. Была собрана установка, состоящая из следующих компонент:

- два АПС Arduino Uno с Ethernet Shield;
- АПС Arduino Yun с ОС OpenWRT 15.05;
- АПС Orange Pi Zero с ОС Armbian;
- управляющий ноутбук с ОС Arch Linux;
- сетевой коммутатор D-Link 1005C;
- сетевые кабели длиной ~ 1 м;
- шестнадцатиканальный логический анализатор Saleae Logic;
- провода питания.

Состав экспериментальной установки приближен к условиям работы ИИУС гетерогенного модульного робота: сеть построена по топологии типа «звезда», коммуникации выполнены кабелем типа UTP и имеют длину не более метра, узлы сети — рекомендованные к применению АПС (см. раздел 6). Несмотря на то, что следствием реализации спецификации для межмодульного информационного взаимодействия является низкая нагрузка на сеть, возможны ситуации, когда из-за ряда обстоятельств, например, таких как одновременная передача нескольких видеопотоков, возможна большая нагрузка на сеть. Это может повлиять на качество передачи сообщений. Следовательно, необходимо проверить работу сети *под нагрузкой и без нагрузки*.

Для эмуляции нагрузки на сеть использована утилита *iperf*. Она состоит из двух частей — «серверной» и «клиентской». «Серверная» часть принимает трафик и выбрасывает его, замеряя скорость приема, «клиентская» часть генерирует трафик, замеряя скорость передачи. Эмуляция передачи 2-х видеопотоков от 2-х модулей выполняется посредством запуска «серверной» части *iperf* на АПС Orange Pi Zero и «клиентских» частей на управляющем ноутбуке и АПС Arduino Yun.

Проведены следующие эксперименты:

1. Проверка предположения: сообщения, передаваемые по сети, не теряются и передаются последовательно без потерь.

2. Определение максимальной частоты синхронного обмена сообщениями.

3. Определение синхронности получения сообщений.

4. Измерение времени передачи и приема сообщений.

Эксперимент 1. Проверка потерь и перемешивания пакетов в сети. Цель эксперимента — проверить тот факт, что при разных условиях загрузки сети пакеты не теряются и их последовательность не перемешивается (хотя это и не гарантируется протоколом UDP).

Загрузим в АПС Arduino Uno «прошивку», посылающую раз в 10 миллисекунд широковещательное сообщение:

```
{ "data": "digital", "n": "0x000000386f" }
```

Листинг 1. Формат широковещательного сообщения

где ключ «n» содержит номер сообщения в виде шестнадцатеричного числа, записанного в виде строки фиксированной длины; тем самым все сообщение получается фиксированной длины в 43 символа. Будем получать на управляющем ноутбуке широковещательные сообщения от АПС Arduino Uno и расшифровывать их, сравнивая полученный номер с предыдущим. Если предыдущий номер отличается от текущего не на -1, выводим соответствующее сообщение. Проведем тесты в условиях сети без нагрузки и под нагрузкой.

Эксперимент показал, что вне зависимости от нагрузки на сеть, *пакеты не теряются и их последовательность не меняется.*

Эксперимент 2. Определение максимальной частоты синхронного обмена сообщениями. Цель эксперимента — замерить максимальную частоту синхронного обмена сообщениями и влияние на эту величину нагрузки на сеть. Для этого будем посылать с управляющего компьютера на метаинтерфейс одного из АПС Arduino Uno команду «ping» и принимать ответ на нее, замеряя время от начала передачи команды до получения и расшифровки ответа. Проверим для двух условий — без нагрузки и под нагрузкой, замерив 10000 посылок.

Определим минимальное, максимальное, медианное и среднее время задержки (Δt). Рассчитаем также среднеквадратическое отклонение среднего времени (σ) и среднюю частоту передачи сообщений ($\overline{f}$). Результаты экспериментов представлены ниже.

Отправка и получение сообщений без нагрузки:

$$\Delta t_{min} = 1.5 \text{ мс},$$

$$\Delta t_{max} = 17.1 \text{ мс},$$

$$\Delta t_{med} = 1.6 \text{ мс},$$

$$\overline{\Delta t} \pm \sigma = 2.0 \pm 0.9 \text{ мс},$$

$$\overline{f} = 499 \text{ Гц}.$$

Отправка и получение сообщений под нагрузкой:

$$\Delta t_{min} = 1.6 \text{ мс},$$

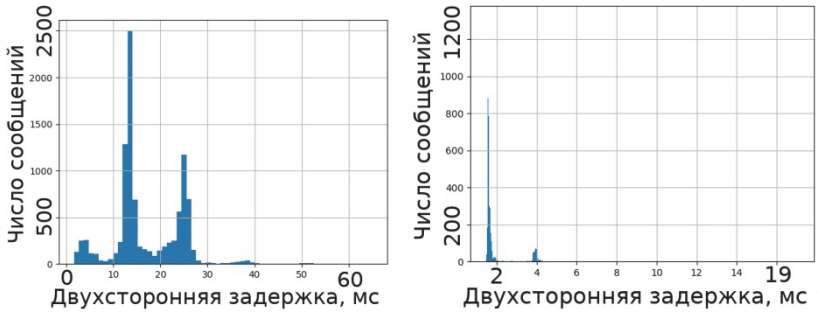
$$\Delta t_{max} = 65.0 \text{ мс},$$

$$\Delta t_{med} = 14.0 \text{ мс},$$

$$\overline{\Delta t} \pm \sigma = 16.9 \pm 7.3 \text{ мс},$$

$$\overline{f} = 59 \text{ Гц}.$$

Результаты эксперимента приведены на рисунке 3. Видно, что скорость передачи падает почти на порядок — с 500 до 60 Гц. Нижняя граница достаточна для управления человеком, *но почти неприменима для таких систем исполнительного уровня, где необходимы частоты порядка 500 Гц и выше.*



а) *б)*

Рис. 3. Гистограмма значений двухсторонней задержки в эксперименте: а) без нагрузки и б) под нагрузкой

Однако такое изменение не скажется на качестве передачи сообщений — представленная ранее спецификация межмодульного взаимодействия рекомендует использовать «синхронные» сокеты только для передачи команд тактического уровня, частота работы которого может быть на один-два порядка ниже, чем полученная в эксперименте частота в 60 Гц.

Эксперимент 3. Определение синхронности получения сообщений. Эксперименты с получением широковещательных сообщений от «асинхронного по UDP» сокета. Цель эксперимента — определить влияние загрузки сети на синхронность получения сообщений. АПС Arduino Uno каждые 10 мс отправляет описанное ранее сообщение (см. листинг 1), а программа на управляющем

ноутбуке замеряет время между получением каждого сообщения. Результаты экспериментов для двух условий — без нагрузки и с нагрузкой представлены ниже:

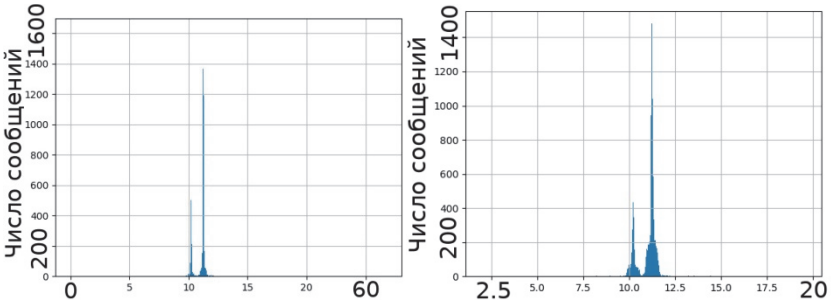
Прием широковещательных сообщений без нагрузки:

$$\Delta t_{min} = 6.5 \text{ мс},$$
$$\Delta t_{max} = 18.5 \text{ мс},$$
$$\Delta t_{med} = 13.3 \text{ мс},$$
$$\overline{\Delta t} \pm \sigma = 13.1 \pm 0.5 \text{ мс}.$$

Прием широковещательных сообщений под нагрузкой

$$\Delta t_{min} = 2.1 \text{ мс},$$
$$\Delta t_{max} = 28.8 \text{ мс},$$
$$\Delta t_{med} = 13.3 \text{ мс},$$
$$\overline{\Delta t} \pm \sigma = 11.2 \pm 0.5 \text{ мс}.$$

Результаты приведены на рисунке 4. Значение Δt_{min} в обоих экспериментах получено как время от запуска программы на управляющем компьютере до момента получения первого сообщения от АПС. Запуск программы на компьютере специально не синхронизовался с передачей сообщений от АПС, и потому Δt_{min} сильно отличается от среднего.



а) *б)*

Рис. 4. Гистограмма значений задержки между приемом сообщений в эксперименте: а) без нагрузки и б) под нагрузкой

По результатам эксперимента можно утверждать, что *нагрузка на сеть незначительно влияет на синхронность передачи широковещательных сообщений с частотой передачи до 100 Гц.*

Эксперимент 4. Измерение времени доставки сообщений. Целью эксперимента является измерение влияния загруженности сети

на время доставки сообщений. Для этого составим две разные программы для двух одинаковых АПС Arduino Uno. Первая программа (сервер) каждые 10 мс меняет состояние одного из выводов АПС на противоположное и отсылает его по сети в виде широковещательного сообщения. Вторая программа (клиент) ожидает прихода сообщения, получая и распаковывая его, устанавливает один из своих выводов в состояние, идентичное полученному по сети. Оба АПС Arduino подключаются к логическому анализатору, который, в свою очередь, подключается к управляющему ноутбуку. Для записи состояний выводов использовалась программа *Saleae Logic* версии 1.2.11. Запишем состояния выводов двух АПС на, соответственно, нулевой и первый канал логического анализатора в двух режимах работы сети — без нагрузки и под нагрузкой. Записи из ПО логического анализатора были помещены в файл и обработаны. В результате были получены следующие характеристики.

Передача сообщений без нагрузки на сеть. Произведена запись в течение 60 секунд; было получено 2725 фронта. Результаты:

$$\Delta t_{min} = 0.71 \text{ мс},$$
$$\Delta t_{max} = 0.80 \text{ мс},$$
$$\Delta t_{med} = 0.76 \text{ мс},$$
$$\overline{\Delta t} \pm \sigma = 0.76 \pm 0.02 \text{ мс}.$$

Передача сообщений при нагрузке на сеть. Произведена запись в течение 60 секунд; было получено 2725 фронта. Результаты:

$$\Delta t_{min} = 0.00 \text{ мс},$$
$$\Delta t_{max} = 1.20 \text{ мс},$$
$$\Delta t_{med} = 0.76 \text{ мс},$$
$$\overline{\Delta t} \pm \sigma = 0.76 \pm 0.03 \text{ мс}.$$

Как видно из результатов эксперимента, нагрузка на сеть *несущественно влияет на время доставки сообщений.*

По итогам приведенных выше экспериментов можно сделать *общий вывод*: выбранный *протокол UDP применим для передачи сообщений исполнительного уровня с частотой обмена сообщениями между устройствами не более 100 Гц при любой нагрузке на сеть. При снижении нагрузки на сеть эта частота может быть увеличена.*

Разработанный механизм межмодульного информационного взаимодействия был реализован для транспортного, силового и частично для модуля сетевого управления в составе лабораторного макета гетерогенного модульного мобильного робота (см. рисунок 2).

Для вычислительных устройств ИИУС модулей были созданы управляющие программы, работающие на основе представленной выше спецификации. Для силового модуля, как наименее производительной части системы, удалось добиться частоты обмена сообщениями до 500 Гц. Эксперименты показали эффективность предложенных решений.

8. Заключение. В качестве основного результата проведенной работы следует считать *решение проблемы информационного взаимодействия* между встраиваемыми вычислительными устройствами при реализации распределенного управления в информационно-измерительной и управляющей системе (ИИУС) роботов с модульной архитектурой.

Предложенное решение основывается на функциональной завершенности модулей робота, которая достигается за счет проектирования каждого модуля как мехатронного или электронного устройства со своей информационно-измерительной и управляющей системой, содержащей все необходимые для реализации своего функционала компоненты. Как следствие такого подхода, реализуется *распределенное управление*, что существенно снижает требование к вычислительной мощности вычислительных устройств отдельных модулей. Это, в свою очередь, позволяет использовать в качестве таких вычислительных устройств недорогие малогабаритные микроконтроллеры и одноплатные ЭВМ.

Предложена сетевая организация структуры ИИУС робота, что позволило перенести такие свойства локальной вычислительной сети как *реконфигурируемость* и масштабируемость на структуру модульного робота. Анализ различных топологий сети показал, что топология типа «звезда» имеет ряд преимуществ по сравнению с топологией типа «шина» для применения в гетерогенных модульных роботах.

Показано, что реализация распределенного управления нуждается в таком механизме информационного взаимодействия, в котором каждый модуль имеет возможность обмениваться информацией *непосредственно с любым другим модулем*. Сравнительный анализ существующих сетей показал, что данному требованию удовлетворяет лишь сеть Ethernet, которая имеет преимущество также по целому ряду других показателей, включая пропускную способность, распространенность оборудования и его стоимость.

Для совместимости программного обеспечения модулей разработана спецификация, предназначенная для создания специального ПО (драйверов) и языка межмодульного информационного взаимодействия, то есть соответствующих

программных интерфейсов, обеспечивающих включение модулей сторонних производителей в режиме «plug and play». Спецификация основана на принципах широко известного программного каркаса (framework) ROS и позволяет реализовать ПО на встраиваемых вычислительных устройствах, в отличие от ROS, который невозможно использовать из-за высоких требований к вычислительной мощности АПС. Спецификация построена на основе библиотеки обмена сообщениями по протоколу TCP — библиотека ZeroMQ, и в дополнение используются сокеты по протоколу UDP для прямого взаимодействия со встраиваемыми системами.

На основе многокритериальной оптимизации по Парето получены рекомендации для выбора аппаратно-программных средств, на базе которых можно реализовать разработанный механизм межмодульного информационного взаимодействия. Соответствующим требованиям удовлетворяют 7 доступных АПС, что обеспечивает широкий выбор для разработчиков модулей.

Работоспособность предложенного метода межмодульного информационного взаимодействия, реализованного на базе сетевого (ЛВС типа Ethernet) и аппаратно-программного (встраиваемые АПС) решения, была проверена в ходе экспериментов на установке, состав которой приближен к условиям работы ИИУС гетерогенного модульного мобильного робота. В программной части использована библиотека ZMQ, дополненная протоколом UDP с широковещательными сообщениями, использованные АПС взяты из рекомендованных к применению. Выполнена проверка работы сети под нагрузкой и без нагрузки. Эксперименты показали, что совместная работа программной и аппаратной частей удовлетворяет всем обозначенным требованиям и *применима для передачи сообщений исполнительного уровня с частотой до 100 Гц, которая может быть увеличена при снижении нагрузки на сеть.*

Предлагаемый метод межмодульного информационного взаимодействия был реализован для транспортного, силового и частично для модуля-супервизора в составе лабораторного макета гетерогенного модульного мобильного робота. Библиотеки функций, реализующих разработанную спецификацию, созданы для трех встраиваемых устройств (Arduino Yun, Arduino Mega2560, Raspberry Pi B+) и использованы для реализации соответствующих драйверов. Разработана система команд для перечисленных модулей.

В дальнейшем необходимо создать реализации спецификации для других архитектур встраиваемых устройств (таких как ARM или Xtensa). Необходимо разработать системы команд для таких модулей,

как модуль манипулятора, сенсорный модуль дальнего радиуса действия и других.

Также предполагается разработать алгоритмы супервизорного управления движением робота в недетерминированной среде и алгоритмы супервизорного управления манипулированием различными объектами.

Литература

1. *Лопота А.В., Юревич Е.И.* Этапы и перспективы развития модульного принципа построения робототехнических систем // Научно-технические ведомости СПбГПУ. Информатика. Телекоммуникации. Управление. СПб: Изд-во Политехнического ун-та. 2013. №1(164). С. 98–103.
2. *Платонов А.К.* Робототехника лунной базы // XXXIV Чтения по космонавтике. URL: http://www.keldysh.ru/section5/report.xhtm?src=section5.xml&filter=3. (дата обращения: 10.06.2017).
3. *Murata S. et al.* M-TRAN: self-reconfigurable modular robotic system // IEEE/ASME Transactions on Mechatronics. 2002. vol. 7(4). pp. 432–441.
4. *Østergaard E.H., Kassow K., Beck R., Lund H.H.* Design of the ATRON lattice-based self-reconfigurable robot // Autonomous Robots. 2006. vol. 21(2). pp. 165–183.
5. Qiao G. et al. Design of Transmote: a Modular Self-Reconfigurable Robot with Versatile Transformation Capabilities // Proceedings of the 2012 IEEE International Conference on Robotics and Biomimetics. 2012. pp. 1331–1336.
6. *Fukuda T., Ueyama T., Kawauchi Y., Arai F.* Concept of cellular robotic system (CEBOT) and basic strategies for its realization // Computers Elect Engng. 1992. vol. 18. no. 1. pp. 11–39.
7. *Baca J., Ferre M., Aracil R.* A heterogeneous modular robotic design for fast response to a diversity of tasks // Robotics and Autonomous Systems. 2012. vol. 60. no. 4. pp. 522–531.
8. *Lyder A.H. et al.* On sub-modularization and morphological heterogeneity in modular robotics // Intelligent Autonomous Systems of Advances in Intelligent Systems and Computing. 2013. vol. 193. no. 12. pp. 649–661.
9. *Hancher M.D., Hornby G.S.* A modular robotic system with applications to space exploration // 2nd IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT'06). 2006. pp. 132–140.
10. *Garcia R.F.M., Lyder A., Christensen D.J., Stoy K.* Reusable Electronics and Adaptable Communication as Implemented in the Odin Modular Robot // IEEE International Conference on Robotics and Automation. 2009. pp. 1152–1158.
11. *Andreev V., Kim V., Pletenev P.* The principle of full functionality – the basis for rapid reconfiguration in heterogeneous modular mobile robots // Proceedings of the 28th DAAAM International Symposium. 2017. pp. 0023–0028.
12. *Quigley M. et al.* ROS: an open-source Robot Operating System // ICRA workshop on open source software. 2009. vol. 3. no. 3.2. pp. 5.
13. *Андреев В.П., Ким В.Л., Подураев Ю.В.* Сетевые решения в архитектуре гетерогенных модульных мобильных роботов // Робототехника и техническая кибернетика. 2016. № 3(12). С. 23–29.
14. EtherCAT Technology Group, Industrial Ethernet Technologies. URL: https://www.ethercat.org/download/documents/Industrial_Ethernet_Technologies.pdf. (дата обращения: 17.03.2016).
15. Ethernet POWERLINK Communication Profile Specification Version 1.2.0. URL: http://www.ethernet-powerlink.org/en/downloads/technical-documents/action/open-

download/download/epsg-ds-301-v120-communication-profile-specification/element/5158/?no_cache=1 (дата обращения: 17.03.2016).
16. RFC: 791 Internet Protocol. URL:https://tools.ietf.org/html/rfc791 (дата обращения: 17.05.2017).
17. Digital Standards Organization COSS – Consensus Oriented Specification System. URL: https://web.archive.org/web/20161002092144/http://www.digistan.org/spec:1/coss (дата обращения: 17.05.2017).
18. *Андреев В.П., Плетенев П.Ф.* Разработка технологии межмодульного общения в гетерогенном модульном мобильном роботе // Труды международной научно-технической конференции «Экстремальная робототехника». 2016. С. 245–255.
19. *Kirsanov K.* Software architecture of control system for heterogeneous group of mobile robots // Procedia Engineering. 2015. vol. 100. pp. 216–221.
20. *Hintjens P.* "0MQ – The Guide". URL: http://zguide.zeromq.org/page:all (дата обращения: 17.03.2016).
21. *Плетенев П.Ф.* 1/ПММВ – Протокол взаимодействия в гетерогенном модульном мобильном роботе. URL: https://asmfreak.github.io/modular_robots_rfc/1/ПММВ/ (дата обращения: 20.01.2017).
22. *Глухих И.Н.* Оптимизация векторного критерия. Парето-оптимальные решения. URL: http://systematy.ru/articles/44_optimizatsiya_vektornogo_kriteriya__pareto-optimalnyie_resheniya (дата обращения: 17.03.2017).

Андреев Виктор Павлович — к-т физ.-мат. наук, д-р техн. наук, профессор кафедры сенсорных и управляющих систем, Московский государственный технологический университет «Станкин» (МГТУ "СТАНКИН"). Область научных интересов: мехатроника, робототехника, информационно-измерительные и управляющие системы, системы технического зрения. Число научных публикаций — 127. andreevvipa@yandex.ru; Вадковский пер., 1, Москва, 127055; р.т.: +7(965)210-7951.

Плетенев Павел Филиппович — аспирант кафедры робототехники и мехатроники, Московский государственный технологический университет «Станкин» (МГТУ "СТАНКИН"). Область научных интересов: робототехника, мехатроника, распределенные системы управления, самоорганизующиеся системы управления. Число научных публикаций — 7. cpp.create@gmail.com; Вадковский пер., 1, Москва, 127055; р.т.: +7(903)2547693.

Поддержка исследований. Работа выполняется при финансовой поддержке РФФИ (гранты 16-07-00811а и 16-07-01264а).

DOI 10.15622/sp.57.6

V.P. ANDREEV, P.F. PLETENEV

METHOD OF INFORMATION INTERACTION FOR DISTRIBUTED CONTROL SYSTEMS OF ROBOTS WITH MODULAR ARCHITECTURE

Andreev V.P., Pletenev P.F. **Method of Information Interaction for Distributed Control Systems of Robots with Modular Architecture.**

Abstract. The paper presents a solution to the problem of information interaction between embedded computing devices in the implementation of *distributed control* in the information-measuring and control system (IMCS) of robots with modular architecture. Distributed control is realized by designing each module as a device with its own IMCS containing all the components necessary to perform its functionality, including computing devices. As a consequence of such a functional completion of the modules, the computations needed to control the robot as a single mechatronic system are *parallelized*. As a result, the requirements to the power of computing devices of modules' IMCS are significantly reduced, as it is possible to use inexpensive microcontrollers and single-board computers — embedded computing devices.

The network organization of the IMCS structure of the robot is proposed, which allowed the transfer of the *reconfigurability* property of the network to the structure of the modular robot. Analysis of various network topologies has shown that the topology of the "star" type has several advantages over the topology of the "bus" type for use in heterogeneous modular robots.

It is shown that the use of the Robot Operating System (ROS) for the implementation of information interaction between embedded computing devices is either impossible or significantly hampered. A specification is proposed for the creation of appropriate programming interfaces and a language for inter-module communication, which enable the inclusion of third-party modules in the plug and play mode. The specification is based on the principles of ROS, but allows one to implement software on embedded computing devices. On the basis of Pareto's multi-criteria optimization, recommendations were obtained for selecting the appropriate hardware and software.

The efficiency of the proposed solution was proved in the course of experiments on an installation which composition is close to the working conditions of the IMCS of a heterogeneous modular robot. The experiments showed that the joint operation of the software and hardware parts meets all the specified requirements and is applicable for the transmission of messages of the executive level with a frequency of up to 100 Hz under any load on the network.

Keywords: modular robot, heterogeneous robot, mobile robot, reconfigurability, control system, distributed control.

Andreev Victor Pavlovich — Dr. Sci., professor of sensory and drive systems department, Moscow State University of Technology «STANKIN» (MSTU «STANKIN»). Research interests: mechatronics, distributed control systems, self-organizing control systems. The number of publications — 127. andreevvipa@yandex.ru; 1, Vadkovsky per., Moscow, 127055, Russia; office phone: +7(965)210-7951.

Pletenev Pavel Filippovich — Ph.D. student of robotics and mechatronics department, Moscow State University of Technology «STANKIN» (MSTU «STANKIN»). Research interests: robotics, mechatronics, distributed control systems, self-organizing control systems. The number of publications — 7. cpp.create@gmail.com; 1, Vadkovsky per., Moscow, 127055, Russia; office phone: +7(903)2547693.

Acknowledgements. This research is supported by RFBR (grants 16-07-00811a and 16-07-01264a).

References

1. Lopota A.V., Jurevich E.I. [Stages and development prospects of robotic systems design modular principle]. *Nauchno-tehnicheskie vedomosti SPbGPU. Informatika. Telekommunikacii. Upravlenie – St. Petersburg State Polytechnical University Journal. Computer Science. Telecommunication and Control Systems*. 2013. vol. 1(164). pp. 98–103. (In Russ.).
2. Platonov A.K. [Robotics of a moon base]. *XXXIV Chtenija po kosmonavtike – XXXIV Readings on Cosmonautics*. Available at: http://www.keldysh.ru/section5/report.xhtm?src=section5.xml&filter=3 (accessed 10.06.2017). (In Russ.).
3. Murata S. et al. M-TRAN: self-reconfigurable modular robotic system. *IEEE/ASME Transactions on Mechatronics*. 2002. vol. 7(4). pp. 432–441.
4. Østergaard E.H., Kassow K., Beck R., Lund H.H. Design of the ATRON lattice-based self-reconfigurable robot. *Autonomous Robots*. 2006. vol. 21(2). pp. 165–183.
5. Qiao G. et al. Design of Transmote: a Modular Self-Reconfigurable Robot with Versatile Transformation Capabilities. Proceedings of the 2012 IEEE International Conference on Robotics and Biomimetics. 2012. pp. 1331–1336.
6. Fukuda T., Ueyama T., Kawauchi Y., Arai F. Concept of cellular robotic system (CEBOT) and basic strategies for its realization. *Computers Elect Engng*. 1992. vol. 18. no. 1. pp. 11–39.
7. Baca J., Ferre M., Aracil R. A heterogeneous modular robotic design for fast response to a diversity of tasks. *Robotics and Autonomous Systems*. 2012. vol. 60. no. 4. pp. 522–531.
8. Lyder A.H. et al. On sub-modularization and morphological heterogeneity in modular robotics. Intelligent Autonomous Systems of Advances in Intelligent Systems and Computing. 2013. vol. 193. no. 12. pp. 649–661.
9. Hancher M.D., Hornby G.S. A modular robotic system with applications to space exploration. 2nd IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT'06). 2006. pp. 132–140.
10. Garcia R.F.M., Lyder A., Christensen D.J., Stoy K. Reusable Electronics and Adaptable Communication as Implemented in the Odin Modular Robot. IEEE International Conference on Robotics and Automation. 2009. pp. 1152–1158.
11. Andreev V., Kim V., Pletenev P. The principle of full functionality – the basis for rapid reconfiguration in heterogeneous modular mobile robots. Proceedings of the 28th DAAAM International Symposium. 2017. pp. 0023–0028.
12. Quigley M. et al. ROS: an open-source Robot Operating System. ICRA workshop on open source software. 2009. vol. 3. no. 3.2. pp. 5.
13. Andreev V.P., Kim V.L., Poduraev Yu.V. [Network–based design of heterogeneous modular mobile robotic systems] *Robototehnika i tehnicheskaja kibernetika – Robotics and Technical Cybernetics*. 2016. vol. 3(12). pp. 23–29. (In Russ.).
14. EtherCAT Technology Group, Industrial Ethernet Technologies. Available at: https://www.ethercat.org/download/documents/Industrial_Ethernet_Technologies.pdf. (accessed: 17.03.2016).
15. Ethernet POWERLINK Communication Profile Specification Version 1.2.0. Available at: http://www.ethernet-powerlink.org/en/downloads/technical-documents/action/open-download/download/epsg-ds-301-v120-communication-profile-specification/element/ 5158/?no_cache=1 (accessed: 17.03.2016).
16. RFC: 791 Internet Protocol. Available at: https://tools.ietf.org/html/rfc791 (accessed: 17.05.2017).
17. Digital Standards Organization COSS – Consensus Oriented Specification System. Available at: https://web.archive.org/web/20161002092144/http://www.digistan.org/spec:1/coss (accessed: 17.05.2017).

18. Andreev V.P., Pletenev P.F. [Developing inter-modular communication for heterogeneous mobile robot]. *Trudy mezhdunarodnoj nauchno-tehnicheskoj konferencii "Jekstremal'naja robototehnika"* [Proceedings of the International Scientific and Technological Conference "Extreme robotics"]. 2016. pp. 245–255. (In Russ.).
19. Kirsanov K. Software architecture of control system for heterogeneous group of mobile robots. *Procedia Engineering*. 2015. vol. 100. pp. 216–221.
20. Hintjens P. "0MQ – The Guide". Available at: http://zguide.zeromq.org/page:all (accessed: 17.03.2016).
21. Pletenev P.F. [1/PIMI – Protocol of intermodular interaction]. Available at: https://asmfreak.github.io/modular_robots_rfc/1/ПММВ/ (accessed: 20.01.2017). (In Russ.).
22. Gluhih I.N. [Optimization of a vector criteria. Pareto-optimal solutions]. Available at: http://systematy.ru/articles/44_optimizatsiya_vektornogo_kriteriya__pareto-optimalnyie_resheniya (accessed: 17.03.2017). (In Russ.).