Рождественский Кирилл Всеволодович – e-mail: kvrxmas@yahoo.com; тел.: 88127142923; д.т.н.; профессор, проректор по МСНиО.

Рыжов Владимир Александрович – e-mail: ryzhov@smtu.ru; тел.: 88124940936; кафедра прикладной математики и математического моделирования; д.т.н.; профессор.

Смольников Александр Васильевич – e-mail: smol@smtu.ru; тел.: 88127141321; к.т.н.; проректор по УР.

Kozhemyakin Igor Vladilenovich – Saint-Petersburg State Marine Technical University; e-mail: 1861vp@mail.ru; 3, Lotsmanskaya street, St. Petersburg, 190008, Russia; phone: +78127146822; head of Division Defense Research and Development.

Rozhdestvensky Kirill Vsevolodovich – e-mail: kvrxmas@yahoo.com; phone: +78127142923; dr. of eng. sc.; professor; Vice-Rector for International Science & Education.

Ryzhov Vladimir Alexandrovich – e-mail: ryzhov@smtu.ru; phone: +78124940936; the department of applied mathematics and mathematical modeling; dr. of eng. sc.; professor.

Smolnikov Alexander Vasilevich – e-mail: smol@smtu.ru; phone: +78127141321; cand. of eng. sc.; Vice-Rector for Academic Affairs.

УДК 551.46.077:629.584

А.И. Боровик, Л.А. Наумов

КОМПОНЕНТНО-ОРИЕНТИРОВАННАЯ ПРОГРАММНАЯ ПЛАТФОРМА ДЛЯ АВТОНОМНЫХ МОБИЛЬНЫХ РОБОТОВ

Рассматриваются вопросы создания системы управления автономным мобильным роботом. Перечисляются требования, которым должна соответствовать среда разработки подобных систем. Предлагается новое средство разработки систем управления – компонентно-ориентированная программная платформа RCE. Описывается архитектура данной платформы, используемые ею парадигмы программирования, язык и технологии организации транспорта данных. Описываются утилиты, входящие в состав платформы. Приводится сравнение архитектуры RCE с архитектурами других популярных программных платформ.

Мобильный робот; автономный робот; система управления; программная платформа; RCE.

A.I. Borovik, L.A. Naumov

COMPONENT-ORIENTED PROGRAM FRAMEWORK FOR AUTONOMOUS MOBILE ROBOTS

This article focuses on development of the control system software for autonomous mobile robots. It describes requirements to the development environment for such systems. It proposes new component-oriented framework for control system development – RCE and describes its programming language, paradigm and communication techniques. It reviews utilities, included in the RCE framework. Article concludes with a comparison of RCE architecture and architectures of other popular robotics frameworks.

Autonomous robot; mobile robot; control system; robotics framework; RCE.

Введение. Одним из важнейших компонентов мобильного робота является система управления (СУ) – совокупность программных средств, обеспечивающих функционирование робота и выполнение им поставленных задач.

На заре робототехники системы управления представляли собой монолитные программы, написанные одним или несколькими программистами. В настоящее время круг задач, решаемых роботами, значительно расширился. Современные системы управления представляют собой сложные программные комплексы, включающие драйверы оборудования, алгоритмы движения, программы обработки и анализа данных от исследовательского оборудования, аварийную систему, программы для взаимодействия с оператором, алгоритмы командной работы и т.п. Очевидно, что для создания подобного комплекса программ требуется привлечение большого количества экспертов из разнообразных отраслей знаний. Для организации их эффективной работы необходима проработка методологии создания систем управления, которая позволила бы экспертам сосредоточиться на решении конкретной прикладной задачи, без необходимости изучать принципы построения систем управления, взаимодействия программ в комплексе и т.п. Проработка подобной методологии позволит не только повысить эффективность и увеличить скорость разработки конкретной системы управления, но и позволит создать общую библиотеку алгоритмов и решений типовых робототехнических задач, которые могут использоваться в рамках разных проектов.

Техническим инструментом указанной методологии может стать программная робототехническая платформа – среда разработки систем управления, в рамках которой определены понятия программных компонент, способы их создания и взаимодействия между собой. Платформа должна предоставлять средства по созданию, настройке и запуску компонентов, их адресации и транспорту сообщений между ними. Подобная платформа может содержать описание некоторых базовых алгоритмов робототехники, драйверы для часто используемого оборудования, утилиты облегчающие сборку и настройку систем управления. Данная статья описывает программную платформу RCE (Robot Components Engine), разработанную автором в рамках создания системы управления автономным необитаемым подводным аппаратом (АНПА) в ИПМТ ДВО РАН.

Требования к программной платформе. Программная робототехническая платформа должна использоваться в качестве инструмента создания систем управления для широкого спектра мобильных роботов. Исходя из целевого назначения, можно сформулировать следующие требования к платформе:

1. Возможность создания систем управления с любой из использующихся в настоящее время моделей архитектуры.
2. Наличие продуманной парадигмы программирования отдельных компонентов и системы управления в целом.
3. Использование удобного языка разработки компонент, а также наличие полнофункциональной среды разработки.
4. Возможность произвольного размещения созданных компонент по компьютерам, входящим в бортовую сеть робота, а также компьютерам операторов и наблюдателей.
5. Эффективность созданных систем управления, возможность их применения в промышленных роботах, а не только в лабораторных образцах.

Все архитектуры систем управления в современной робототехнике можно условно разделить на три модели: *«очувствление-планирование-действие»*, *«поведенческая»* и *«гибридная»* [1–4]. Модель типа *«очувствление-планирование-действие»* предполагает разделение компонентов системы по вертикальным уровням. Цепочка компонентов последовательно выполняет сбор данных сенсоров, занесение данных в исходную модель мира, декомпозицию задачи управления, выработку и передачу управляющих воздействий на актуаторы [1, 4]. Согласно

поведенческой модели все компоненты системы управления разделяются по горизонтальным уровням, работающим параллельно и независимо. Результирующие команды управления формируются специальным компонентом-арбитром в соответствии с заложенной системой приоритетов [1, 3]. *Гибридная* модель использует вертикальные уровни компонентов, на нижнем из которых параллельно и независимо выполняются компоненты, отвечающие за формирование управляющих команд [2, 3].

Исходя из приведенной типизации архитектур, универсальная программная платформа, на базе которой можно построить систему управления любой модели, должна предоставлять следующие возможности:

- параллельное исполнение компонент системы управления;
- транспорт синхронных и асинхронных запросов, диспетчеризация работы компонентов;
- система приоритетов.

Большинство современных робототехнических платформ используют событийно-ориентированную парадигму программирования для описания компонент системы управления. Согласно данной парадигме компонент описывается как совокупность реакций на внешние события. Таким образом, основной частью компонента является цикл ожидания события, который должен включать следующие шаги [5]:

- ожидание некоторого явления;
- регистрация явления и его интерпретация как события;
- обработка события.

Данная парадигма представляется наиболее применимой для описания компонентов сложной системы, поскольку дает четкое представление о выполняемых компонентом задачах, входных и выходных данных, а также принципах его работы.

В качестве парадигмы описания системы управления наиболее оптимальной является компонентно-ориентированная парадигма, успешно используемая в таких популярных программных платформах как Player [6] и Orca [7]. Согласно данной парадигме каждый компонент представляет собой законченный программный модуль, который не зависит от остальных компонентов и может храниться в виде исходного кода или скомпилированного бинарного файла. Для связи компонентов между собой используются специальные списки правил взаимодействия – интерфейсы. Интерфейсы утверждаются на начальном этапе разработки системы и не меняются в процессе. При изменении интерфейса требуется изменение всех компонентов, зависящих от него, при этом программная платформа разработки компонентно-ориентированных систем не должна позволять использовать в рамках одной системы компоненты, написанные для разных версий одного интерфейса [8].

Робототехническая платформа может либо предлагать свой язык для описания компонент систем управления, либо использовать один из существующих универсальных языков программирования. Использование популярного языка программирования значительно облегчит работу экспертам, позволит им использовать привычные среды разработки, кроме того, значительно расширит возможности созданных компонентов, поскольку позволит использовать не только функционал, предоставляемый платформой, но и функционал языка программирования, а также сторонние библиотеки функций.

Требование возможности произвольного размещения созданных компонент по всем компьютерам робототехнического комплекса влечет за собой требование кроссплатформенности программной платформы. В состав современного робототехнического комплекса может входить несколько компьютеров, находящихся под управлением разных операционных систем. Так, на компьютере робота может

быть размещена операционная система реального времени, вроде QNX или Windows CE, тогда как оператору удобнее работать за компьютером под управлением системы с привычным графическим интерфейсом – Windows семейства NT или современном дистрибутиве Linux. Программная платформа должна работать на всех операционных системах, используемых в современной робототехнике (ОС семейств Linux, Windows NT, Windows CE, QNX, Solaris, VxWorks).

Для того чтобы созданные на базе программной платформы системы управления могли успешно применяться в реальных задачах, она должна удовлетворять следующим требованиям [9]:

- стабильность (ошибки, возникающие в отдельных модулях системы не должны приводить к выходу из строя всей системы);
- низкая ресурсоемкость (возможность программного обеспечения СУ функционировать на промышленных одноплатных компьютерах, используемых в бортовой сети АНПА);
- малое время отклика (данные от сенсоров, непосредственно участвующих в управлении движением, должны доставляться регуляторам движения с минимально возможными задержками).

Эти требования должны быть поставлены во главу угла при разработке программной платформы.

Архитектура платформы RCE. Данная статья посвящена программной платформе Robot Components Engine (RCE), разработанной с учетом указанных требований. Изначально платформа создавалась для построения на ее основе модульной системы управления АНПА, однако полученный программный продукт и предлагаемый им метод разработки могут использоваться для создания СУ любого мобильного робота. На архитектуру и принцип построения платформы значительным образом повлияли такие программные продукты, как Player/Stage Project, Orca, CARMEN и Microsoft Robotics Studio.

Платформа RCE предлагает собственную компонентную модель, базовые понятия которой определены следующим образом:

- Компонент – выделенная часть программного обеспечения системы управления роботом. Компонентами являются драйверы устройств, программы, реализующие различные алгоритмы поведения и т.п.
- Сообщение – набор логически связанных данных, единовременно передаваемых между компонентами. Сообщение, например, может содержать текущую глобальную позицию робота или данные некоторого датчика.
- Интерфейс – совокупность описаний нескольких сообщений, объединенных некоторым общим признаком, например источником данных. Так, можно говорить об интерфейсе позиционной системы или интерфейсе сонара.

Для написания компонентов используется один из самых популярных в настоящее время языков программирования – C++ [10]. Платформа предоставляет разработчику описание базового класса, наследование от которого позволит легко описать компонент любого типа. Согласно принципам компонентно-ориентированной парадигмы, компонент должен быть полностью независим от других компонентов и не может использовать классы, функции и методы, описанные в их исходном коде. Компонент может быть скомпилирован одним из двух способов:

- В виде разделяемой (динамической) библиотеки.
- В виде исполняемого бинарного файла (приложения).

При компиляции в виде динамической библиотеки для запуска компонента должна использоваться специальная программа – *хост-сервер RCE*. Код компонента исполняется в одном потоке операционной системы. В случае компиляции компонента как динамической библиотеки возможен запуск нескольких компонент-потоков внутри одного процесса операционной системы – *хост-сервера*. Та-

кой подход значительно увеличивает скорость обмена данными между компонентами (поскольку используется общая память процесса), однако уменьшает надежность всего программного комплекса системы управления (критическая ошибка внутри одного потока-компонента приведет к терминированию всего хост-процесса операционной системой). Решение об объединении компонентов в один процесс принимается архитектором системы управления на основе принципиальной связанности компонентов.

Описание интерфейса представляет собой заголовочный файл языка C++. Все сообщения, принадлежащие данному интерфейсу, должны быть описаны в рамках одного заголовочного файла. Для описания сообщения необходимо наследоваться от базового класса сообщения, предоставляемого платформой. Сообщение может содержать не только данные, но и методы работы с ними, в полном соответствии с объектно-ориентированной парадигмой программирования. На описание сообщения накладываются некоторые специфические ограничения (так, запрещено наследование от каких-либо других классов, кроме базового класса сообщения), однако в целом описание полностью соответствует концепции, задаваемой языком C++. Часть интерфейсов включены в поставку программной платформы. Созданные пользователем интерфейсы компилируются в виде динамических или статических библиотек с использованием специальной программы «*препроцессор RCE*», входящей в поставку платформы.

Платформа RCE организует между компонентами связь гибридного типа. В сети может присутствовать сервер, осуществляющий координацию работы узлов сети, однако передача сообщений между компонентами осуществляется напрямую [11]. Главная задача сервера – поддержание актуальной таблицы интерфейсов, согласно которой каждому используемому в системе интерфейсу ставится в соответствие номер порта. Роль сервера может играть конфигурационный файл, в котором заранее описана данная таблица.

Сообщения между компонентами могут передаваться двумя способами: *синхронно* и *асинхронно*. При *синхронной* передаче источник отправляет сообщение одному конкретному получателю и обязательно ждет ответа (другого сообщения, зарегистрированного в качестве ответного в интерфейсе). При *асинхронной* передаче источник публикует сообщение, которое будет доставлено системой для всех заинтересованных получателей автоматически; никаких подтверждений о получении и обработке такого сообщения не предусмотрено.

Для доставки асинхронных сообщений используется технология групповой передачи данных IP-Multicast [12]. Каждому интерфейсу системы соответствует определенный порт multicast-адреса. Таблица соответствия зарегистрированных в системе интерфейсов номерам портов передается компоненту при запуске из конфигурационного файла или в сообщении от сервера. При публикации синхронного сообщения компонентом, платформа RCE отправляет его внутри конкретной Multicast-группы, соответствующей интерфейсу. Все компоненты, подписанные на данный интерфейс, получат опубликованное таким образом сообщение. Если конкретное сообщение в интерфейсе компонентом не обрабатывается, оно будет отфильтровано средствами платформы. Таким образом достигается оптимальное использование системных и сетевых ресурсов. Технология Multicast гарантирует, что сообщение не будет скопировано большее количество раз, чем это необходимо для доставки его каждому потребителю [12]. В случае если физическая реализация сети конкретного робота не поддерживает технологию IP-Multicast, система позволяет использовать вместо нее обычную широковещательную технологию (IP Broadcast) или прямое соединение.

Помимо сетевых функций программная платформа RCE также предоставляет широкий набор средств по конфигурированию компонентов. В состав средств платформы входит функция парсинга конфигурационных файлов определенного синтаксиса. Платформа предполагает максимально возможную настройку каждого компонента посредством конфигурационного файла. Соглашения между разработчиками должны запрещать компоненту получать настройки иным способом.

Компоненты RCE реализованы согласно принципам событийно-ориентированного программирования. Алгоритм работы компонента может быть представлен следующим образом:

1. Запуск компонента. Получение настроек из конфигурационного файла посредством функций платформы. Инициализация оборудования и прочие действия, связанные с настройкой функционала компонента согласно указанным в конфигурационном файле параметрам.
2. Подписка на асинхронно публикуемые сообщения. Информирование платформы о том, какие синхронные сообщения могут обрабатываться компонентом и какие синхронные и асинхронные сообщения будут им публиковаться. Установка длины очереди для каждого типа сообщений. Регистрация таймеров. Регистрация событий. Событие – это явление, при возникновении которого происходит выход из функции ожидания события. Событием может быть получение некоторых сообщений или срабатывание таймеров.
3. Основной цикл работы компонента.
4. Завершающий этап работы компонента – отключение оборудования, очистка динамически выделенной памяти и прочие действия, связанные с остановкой работы компонента.

Переход к последнему пункту алгоритма может быть осуществлен при получении компонентом специального сообщения о завершении работы, генерации соответствующего сигнала операционной системой или возникновении некоторого критического внутреннего состояния – отказа работы оборудования, ошибки алгоритма и т.п.

Основной цикл работы компонента состоит из последовательного выполнения следующих действий:

1. Ожидание события.
2. Идентификация события.
3. Реакция на событие.

В качестве реакции может быть публикация компонентом некоторых сообщений (к примеру, при срабатывании таймера публикации), ответ на синхронный запрос (синхронные сообщения всегда регистрируются как события), некоторые действия с оборудованием и т.п.

Утилиты платформы. В состав платформы RCE входят следующие утилиты:

- *Библиотека RCE* – совокупность классов и функций языка C++, реализующих функционал платформы по описанию компонентов, транспорту сообщений между ними, разбору конфигурационных файлов, функционированию таймеров и т.п.
- *Препроцессор RCE* – программа, обрабатывающая файлы описания интерфейсов и формирующая на их основе код для статических или динамических интерфейсных библиотек.
- *Хост-сервер RCE* – приложение, которое может служить в качестве процесса-контейнера для модулей-компонентов.
- *Сетевой сервер RCE* – приложение, выполняющее синхронизацию и координацию работы сети компонентов, согласно гибридной технологии организации связи.

- *Среда RCE* – программная среда, с помощью которой можно разрабатывать и настраивать модули RCE, а также создавать на их основе системы управления. С точки зрения технической реализации, среда представлена набором скриптов системы CMake [13], документации и программы по конфигурированию систем управления.

Предлагаемый средой RCE алгоритм создания систем управления. Среда RCE постулирует следующую последовательность действий при разработке систем управления мобильным роботом:

1. Разделение всего функционала системы управления по отдельным компонентам. Определение входных и выходных данных каждого компонента. Выделение событий.
2. Решение о повторном использовании уже написанных ранее компонентов RCE (входящих в поставку платформы или имеющихся в наличии у разработчиков).
3. Проработка и утверждение интерфейсов. Выбор используемых базовых интерфейсов из поставки платформы, описание новых интерфейсов. Компиляция разработанных интерфейсов в виде статических или динамических библиотек.
4. Разработка компонента-заглушки, публикующего тестовые данные и подписывающегося на все интерфейсы, определенные в системе.
5. Параллельная и независимая разработка компонентов разными группами исследователей.
6. Финальная сборка системы управления, решение о размещении компонент по системным единицам обработки и бортовым компьютерам, осуществляемая архитектором СУ. Тестирование готовой системы.

В дальнейшем планируется создание эмулирующего комплекса RCE, который сможет публиковать симуляционные данные по всем базовым интерфейсам. Тогда в алгоритм разработки системы управления вместо разработки компонента-заглушки потребуется включить разработку плагинов для эмулирующего комплекса, работающих с новыми интерфейсами.

Сравнение платформы RCE с существующими аналогами. Проведем краткое сравнение функционала RCE с функционалом других популярных робототехнических платформ. Сравнение скорости и эффективности работы созданных систем управления выходит за рамки данной статьи и приведено не будет.

Наиболее схожей концепцией с RCE обладает программная платформа Player [6, 14]. Между тем, основной режим работы компонентов платформы Player реализован в виде потоков общего процесса сервера платформы. Как было отмечено выше, данный режим отличается низкой стабильностью целевой системы управления – ошибка в одном компоненте может привести к завершению работы всего хост-процесса. Транспорт между компонентами, работающими под управлением разных серверов, возможен, однако осуществляется посредством прямых TCP- или UDP-соединений между серверами, что может привести к неэффективному использованию сетевых ресурсов. Платформа Player жестко разделяет все компоненты на две категории: *серверные* (выполняющиеся на роботе) и *клиентские* (выполняющиеся на компьютерах оператора и наблюдателей), при этом последние обладают изначально неполной функциональностью (не могут публиковать сообщения). Подобный подход приводит к усложнению проектирования системы управления. Платформа Player не содержит четкой концепции события, а также возможности регистрировать таймеры. Кроме того, в Player используется собственный язык описания интерфейсов; сообщения описываются как структуры языка C (т.е. нет возможности включить методы обработки информации в интерфейс).

Платформа Orca [1, 7] использует промежуточное программное обеспечение ICE [15] для организации транспорта сообщений между компонентами. Как следствие, разработка системы управления ставится в прямую зависимость от ICE, которое имеет ряд проблем при компиляции на целевых операционных системах, в частности QNX. ICE работает по принципу клиент-серверного обмена данными, поэтому для функционирования системы управления необходима работа нескольких служб ICE (IceGrid Registry и IceStorm) на одном из компьютеров сети. При выходе данного компьютера из строя (или аварийной остановки служб ICE) вся система управления станет неработоспособна. Как и Player, Orca предоставляет собственный язык для описания интерфейсов. Кроме того, в Orca отсутствует возможность исполнять компоненты в разных потоках общего процесса сервера.

Другая популярная программная платформа CARMEN [16] также не предоставляет возможности исполнения компонентов внутри общего сервера. Как и Orca, CARMEN использует клиент-серверный принцип обмена данными между компонентами. Кроме того, CARMEN не поддерживает операционную систему QNX (хотя промежуточное программное обеспечение, на котором она основана – IPC – в системе QNX работает).

В настоящее время RCE не содержит средств симуляции (хотя симуляционный интерфейс предусмотрен платформой), тогда как Player содержит 2d- и 3d-симуляторы, а ORCA позволяет использовать симуляторы из проекта Player для собственных компонентов. Разработка полноценного 3d-симулятора является наиболее приоритетным шагом по развитию платформы.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Oreback*. A Component Framework for Autonomous Mobile Robots. PhD thesis, KTH Numerical Analysis and Computer Science, Stockholm, Sweden, 2004.
2. *Агеев М.Д., Киселев Л.В., Матвиенко Ю.В. и др. /* Под общ. ред. М.Д. Агеева. «Автономные подводные роботы: системы и технологии». – М.: Наука, 2005.
3. *Arkin R.C.* Behavior-Based Robotics. The MIT Press, Cambridge, Massachusetts, London, England, 1998.
4. *Kortenkamp D., Bonasso R.P., Murphy R.* Artificial Intelligence and Mobile Robots – Case Studies of Successful Robot Systems", AAAI Press / The MIT Press, 1998.
5. *K. Mani Chandy.* Event-Driven Applications: Costs, Benefits and Design Approaches. California Institute of Technology, 2006
6. Player [Электронный ресурс] / The Player Project. – Режим доступа: http://playerstage.sourceforge.net/ index.php?src=player, свободный. – Загл. с экрана. – Яз. англ.
7. Orca: Components for Robotics [Электронный ресурс] / Orca Robotics; Web-мастер Tobias Kaupp. – Режим доступа: http://orca-robotics.sourceforge.net /, свободный. – Загл. с экрана. - Яз. англ.
8. *Szyperski.* Component Software – Beyond Object-Oriented Programming. Addison-Wesley, 1998.
9. *Наумов Л.А., Боровик А.И., Баль Н.В.* RCE – программная платформа для системы управления АНПА // Подводные исследования и робототехника. – 2011. – № 2 (12). – С. 18-25.
10. TIOBE Programming Community Index for June 2012 [Электронный ресурс] / TIOBE. – Режим доступа: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html, свободный. – Загл. с экрана. – Яз. англ.
11. *Lua, Eng Keong; Crowcroft, Jon; Pias, Marcelo; Sharma, Ravi; Lim, Steven*. A survey and comparison of peer-to-peer overlay network schemes // IEEE Communications Surveys and Tutorials – COMSUR. – 2005. – Vol. 7, №. 1-4. – P. 72-93.
12. *Bob Quinn, Dave Shute.* Windows Sockets Network Programming. – M.: Addison-Wesley, 1995.
13. CMake [Электронный ресурс] / CMake – Cross Platform Make. – Режим доступа: http://www.cmake.org/, свободный. – Загл. с экрана. – Яз. англ.

14. *Brian Gerkey, Richard T. Vaughan and Andrew Howard*. The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems. In Proceedings of the 11th International Conference on Advanced Robotics (ICAR 2003). – Coimbra, Portugal, June 2003. – P. 317-323.
15. The Internet Communications Engine [Электронный ресурс] / ZeroC, Inc. – Режим доступа: http://www.zeroc.com/ice.html, свободный. – Загл. с экрана. – Яз. англ.
16. Carmen Robot Navigation Toolkit [Электронный ресурс] / CARMEN-Team. – Режим доступа: http://carmen.sourceforge.net/, свободный. – Загл. с экрана. – Яз. англ.

Статью рекомендовал к опубликованию д.т.н., профессор А.А. Светлаков.

Боровик Алексей Игоревич – Институт проблем морских технологий Дальневосточного отделения Российской академии наук; e-mail: alexey@borovik.me; 690091, г. Владивосток, ул. Суханова, 5А; тел.: 84232215545, доб. 445; телекоммуникационный центр; стажер-исследователь.

Наумов Леонид Анатольевич – e-mail: naumov@marine.febras.ru; тел.: 84232432651; д.т.н.; директор.

Borovik Alexey Igorevich – – Institute of Marine Technology Problems of the Far Eastern Branch of the Russian Academy of Sciences; e-mail: alexey@borovik.me; 5A, Sukhanova street, Vladivostok, 690091, Russia; phone: +74232215545, ext. 445; Center of telecommunications; graduate employee.

Naumov Leonid Analolievich – e-mail: naumov@marine.febras.ru; phone: +74232432651; dr. of eng. sc.; director.

УДК 621.31

В.А. Герасимов, А.Ю. Филоженко, П.И. Чепурин

СТРУКТУРА СИСТЕМЫ ЭЛЕКТРОСНАБЖЕНИЯ АВТОНОМНОГО НЕОБИТАЕМОГО ПОДВОДНОГО АППАРАТА

Рассмотрена задача определения структуры системы бесконтактной передачи электрической энергии для зарядки аккумуляторных батарей автономного необитаемого подводного аппарата. Определены требования к практическим схемным реализациям импульсных преобразователей системы – инвертору и зарядному устройству, а также к выбору силовых и защитных компонентов этих узлов. Приведены расчетные соотношения для силового высокочастотного трансформатора, определяющие эффективность передачи электроэнергии на подводный аппарат, и обозначены соответствующие конструктивные решения. Показаны результаты математического моделирования в среде MATLAB-Simulink и экспериментального исследования полномасштабного лабораторного макета системы. Сделано заключение о возможных путях параметрической оптимизации системы.

Подводный аппарат; энергообеспечение; аккумуляторы; бесконтактная передача энергии; высокочастотный трансформатор; моделирование; экспериментальные исследования.

V.A. Gerasimov, A.J. Filozhenko, P.I. Chepurin

STRUCTURE OF THE SYSTEM NONCONTACT ENERGY ISSUE OF THE AUTONOMOUS UNDERSEA DEVICE

Article is devoted to the problem of structure of determination for contactless power transmission system, used for supplying electric energy charge underwater vehicle`s batteries. The requirements for the practical circuit realization of pulse converters are determined – invertor and charger, but also selection of power and protective components of these devices. Calculated ratios for the high-frequency power transformer, determining the efficiency of electricity transmission to the underwater vehicle, also identified by the corresponding design solutions. The results of math-