Программный комплекс навигации и управления беспилотными транспортными средствами[1]

К.С. Яковлев, А.В. Петров, В.В. Хитьков

Аннотация. Работа посвящена методам и подходам к проектированию и разработке интеллектуальных систем управления беспилотными транспортными средствами, в частности беспилотными летательными аппаратами, способными к автономному функционированию в окружающей среде. Рассматриваются вопросы, связанные с концептуальным проектированием тактического уровня системы управления и с программной реализацией системы. Описывается экспериментальный образец программного комплекса навигации и управления, реализованный на базе открытого ПО и с использованием сред облачных вычислений.

Ключевые слова: беспилотное транспортное средство, интеллектуальная система управления, навигация, свободно-распространяемое ПО, открытое ПО.

Введение

В настоящее время наблюдается существенный прогресс в области повышения производительности электронных вычислительных машин, создания компактных, встраиваемых вычислительных систем, создания новых типов сенсорных систем (лазерные дальномеры, радары) и уменьшения линейных размеров при повышении точности измерений существующих систем (акселерометры, гироскопы, видеокамеры и др.). Эти обстоятельства создают предпосылки для создания транспортных систем (средств) нового поколения, а именно, беспилотных транспортных средств (БТС), способных к автономному выполнению поставленных задач, в том числе при функционировании в динамической среде. Одно из заметных достижений в этой области – появление и широкое распространение унифицированных робототехнических платформ транспортных средств, оснащенных необходимым набором датчиков и исполнительных механизмов (актуаторов), а также программным обеспечением с открытыми интерфейсами, что позволяет исследователям сосредоточиться непосредственно на программной реализации системы управления (СУ) аппаратом. Значительный интерес в этой связи представляет разработка систем управления для беспилотных летательных аппаратов (БЛА) вертикального взлета и посадки, т.к. в отличие от других объектов управления (колесных БТС, БЛА самолетного типа) они одновременно обладают сложной нелинейной динамикой и функционируют в трехмерном пространстве. И именно с решением этой задачи – задачи автоматизации управления БЛА вертолетного типа – связывают существенный прогресс в области создания новых методов и алгоритмов распознавания образов, управления, планирования траектории, построения модели мира и др.

[1] Работа выполнена при финансовой поддержке Минобрнауки России по государственному контракту № 14.514.11.4081 в рамках ФЦП «Исследования и разработки по приоритетным направлениям развития научно-технологического комплекса России на 2007-2013 годы».

К настоящему моменту в научной литературе описано множество подходов к построению систем управления беспилотными аппаратами [1, 2]. Существуют определенные попытки стандартизации в этой области [3]. Традиционно в робототехнике, искусственном интеллекте и смежных дисциплинах под системами управления понимаются двух уровневые системы, состоящие из делиберативного и реактивного уровней управления [4]. На реактивном уровне решаются базовые задачи, связанные с выдерживанием заданных параметров объекта управления (скорость, угловое положение и т.д.). Эти задачи имеют наивысший приоритет, и обычно их решение производится в режиме реального времени непосредственно на бортовом вычислителе объекта управления (в рассматриваемом случае – на бортовом вычислителе беспилотного транспортного средства). При этом естественным образом реактивный уровень управления «замкнут» на исполнительные механизмы (актуаторы) БТС. Также именно на этом уровне производится первичная обработка информации, поступающей с датчиков БТС. Данные, необходимые непосредственно для решения текущих задач, используются программными модулями уровня управления, остальная информация передается на делиберативный уровень.

На делиберативном уровне решается множество задач - постановка целей, определение их приоритетов, прогнозирование, планирование и т.п. Именно разнообразием задач (в том числе и концептуальным) обуславливается необходимость дальнейшего разбиения делиберативного уровня на подуровни. Цитируя [5], в качестве примера можно привести задачу планирования, «одновременно решаемую на делиберативном уровнях в разных контекстах (и разными методами) - планирование поведения, планирование траектории».

В вышеупомянутой работе [5] обоснована целесообразность разработки иерархических трехуровневых систем управления для современных беспилотных летательных аппаратов, функции управления которых разбиты на:

1. стратегический (верхний) уровень управления, ответственный за постановку и выбор целей, прогнозирование, высокоуровневую обработку информации;

2. тактический (промежуточный) уровень управления, ответственный за распознавание образов, построение карты местности, планирование траектории;

3. уровень управления (нижний уровень), ответственный за выдерживание параметров управления исполнительными механизмами.

В данной работе принимается такая трехуровневая архитектура системы управления, и рассматривается ряд вопросов, связанных с непосредственно программной реализацией системы для решения конкретного типа задач. Для сокращения времени на разработку предлагается использовать свободно-распространяемое программное обеспечение (отрытое программное обеспечение, открытое ПО), в частности программные обеспечение Robotic Operating System (ROS) [6, 7]. Использование ROS позволяет существенно сократить временные и производственные издержки, а также повысить степень унификации разрабатываемой системы.

Дальнейшее изложение организовано следующим образом. В разделе 2 описывается в обобщенном виде постановка решаемой задачи (автоматизации управления БЛА вертикального взлета/посадки). В разделе 3 предлагается концептуальная архитектура системы управления для решения поставленной задачи. Раздел 4 посвящен аналитическому обзору функциональных возможностей ROS и способов применения этого открытого ПО для программной реализации системы управления. В разделе 5 представлено описание программного комплекса навигации и управления на базе ROS, разрабатываемого коллективом авторов.

1. Постановка задачи

Здесь и далее в качестве модельной задачи будет рассматриваться задача автоматизации управления БЛА (возможно – несколькими БЛА) вертикального взлета и посадки. Предполагается оснащение инерциальной системой навигации, позволяющей непосредственно оценивать следующие параметры движения (полета): высоту над подстилающей поверхностью, мгновенное ускорение (в 3 направлениях), изменение углов (по 3 осям) в системе координат БЛА, а также – набором дополнительных датчиков, требования к которому пока не заданы.

Кроме того, предполагается отсутствие возможности использования глобальных систем навигации (GPS, ГЛОНАСС) для определения местоположения БЛА в мировой системе координат, а также отсутствие в каком-либо виде априорной информации об окружающей среде (фактически речь идет об отсутствии цифровой модели (карты) местности). Считается, что задачи связанные с выдерживанием необходимых параметров для обеспечения заданных режимов движения (полет вперед/назад/влево/вправо, зависание, разворот на заданное число градусов) успешно решаются на бортовом вычислителе БЛА. Задачи, возникающие на верхних уровнях управления, решаются на внешнем вычислителе, который связан с БЛА надежным каналом связи. Цель считается заданной извне (например, поставленной оператором) причем в том виде, который «понятен» разрабатываемой системе управления БЛА. Примером такой цели может являться цель - «обнаружить в зоне видимости датчиков БЛА объект, заданный оператором с помощью обучающей выборки образцов».

Основные рассматриваемые проблемы:

1) концептуальная организация системы управления;

2) программная реализация модулей системы управления.

2. Концептуальная архитектура системы управления беспилотным средством

Анализируя поставленную задачу, можно сделать вывод о том, что разрабатываемая система управления не содержит верхнего – стратегического уровня управления, ответственного за целеполагание, концептуальное планирование, прогнозирование и т.д., так как глобальная цель известна и задается извне. Также напомним, что задачи, связанные с выдерживанием параметров, необходимых для обеспечения определенных режимов полета, считаются разрешенными. То есть нижний уровень управления считается реализованным: модули системы управления этого уровня исполняются на бортовом вычислителе БЛА. Таким образом, требуется разработка лишь модулей тактического уровня (уровень управления уже реализован, а стратегический уровень является вырожденным), исполняемых на внешнем по отношению к управляемому БЛА вычислителю.

Поскольку набор целей, задаваемых оператором, полностью не определен (известен лишь один пример постановки цели, связанный с обнаружением целевого объекта), то представляется целесообразным реализация тактического уровня управления таким образом, что оперативное расширение функциональности производится «на лету», без внесения существенных изменений в структуру и архитектуру системы. С точки зрения концептуального проектирования это означает, что описание процессов, выполняемых для решения функциональных задач тактического уровня управления, должно быть дано в максимально обобщенном виде, так что становится возможным встраивание конкретного функционала в работающую систему, без изменения структуры последней. Приведем такое описание.

Основными задачами, решаемыми системой управления БЛА на тактическом уровне являются задачи:

– построения модели мира (в виде метрической карты окружающего пространства);

– определения «места» БЛА в этой модели (определения географического местоположения БЛА на построенной карте);

– планирования (построения траектории БЛА в виде определённой последовательности местоположений, привязанных к построенной карте);

– оценивания свойств (обработки поступающей информации с целью выявления существенных характеристик, позволяющих решить поставленную оператором задачу – например, распознавание графических образов с целью обнаружения целевых объектов).

Наиболее проработанные методы решения первых двух задач предполагают их интеграцию в единую субзадачу – задачу одновременного построения карты и определения местоположения БЛА на этой карте. Общепринятое обозначение этой задачи – SLAM, от английского Simultaneous Location and Mapping [8]. При этом важнейшей подзадачей SLAM является определение расстояния до окружающих БЛА объектов (по информации от датчиков БЛА).

Выбор того или иного метода решения SLAM задачи (по существу являющегося композицией ряда алгоритмов решения частных задач) существенно зависит от объема и типа данных, поступающих от уровня управления (фактически – от датчиков БЛА). Можно утверждать, что состав и тип установленных на БЛА датчиков определяет выбор алгоритма решения подзадачи определения расстояния до объектов, который является ключевым алгоритмом решения SLAM задачи в целом. Так, если БЛА (помимо указанной выше инерциальной системы навигации) оснащен лазерным дальномером, то целесообразно применение алгоритмов сопоставления сканов (scan matching) [9]. Модификации этих алгоритмов также могут применяться тогда, когда информация о расстояниях до объектов окружающей среды доступна в явном виде (например, если БЛА оснащен не лазерным дальномером, но ультразвуковыми датчиками измерения расстояния). В противном случае применяются косвенные методы определения расстояний до объектов. Не вдаваясь в подробности, отметим здесь лишь существование большого числа алгоритмов решения SLAM задачи по видео-данным, опирающихся на методы вычислительной геометрии и компьютерного зрения и подходящих для решения задач управления и навигации БЛА, оснащенных лишь видео-камерами [10, 11].

Задача оценивания свойств – это задача интеллектуального анализа данных, целью которого в общем случае является выделение значимых (с точки зрения поставленных перед системой управления целей) зависимостей в данных, поступающих с нижнего уровня управления (от датчиков БТС). Поскольку рассматриваемая проблема (раздел 2) подразумевает возможность обнаружения целевых объектов по набору образцов, то для решения задачи оценивания свойств возникает необходимость применения методов машинного обучения [12]. Наиболее очевидный прогнозный вариант задачи обнаружения – задача анализа видеопотока БЛА на предмет обнаружения объектов, заданных обучающей выборкой фотообразцов. Для решения этой задачи целесообразным представляется использование хорошо зарекомендовавших в области распознавания образов модификаций алгоритма Виолы-Джонса, в частности модификации, основанной на использовании каскада классификаторов [13].

Задача планирования траектории в общем виде предусматривает необходимость построения модели окружающего БЛА пространства. В качестве исходных для построения такой модели данных могут выступать данные, являющиеся выходными при решении SLAM задачи. Целесообразным при этом представляется трансформация решения SLAM задачи в графовый вид, т.е. хранение карты окружающего пространства в виде взвешенного графа, вершинам которого соответствуют координаты центров некоторых областей, а ребрам – т.н. элементарные траектории (обычно – отрезки прямых), соединяющие эти центры определенным образом между собой. Достаточно подробный аналитический обзор графовых моделей, применяемых для решения задачи планирования траектории, приведен в [14]. Задача планирования траектории, рассматриваемая как задача поиска пути на графе, может решаться множеством известных способов. Применительно к рассматриваемой предметной области, наиболее целесообразным видится применения эвристических алгоритмов поиска, основанных на итерационном обходе вершин графа [15], а также декомпозиционных алгоритмов, осуществляющих иерархическое разбиение исходной задачи на ряд легкоразрешимых подзадач с последующей композицией локальных решений [16].

Подводя итог вышесказанному, целесообразной для решения поставленной задачи представляется концептуальная архитектура тактического уровня системы управления, представленная на Рис. 1.

Интерфейс с оператором, предоставляющий в общем случае возможность взаимодействия со всеми структурными блоками системы, необходим для предъявления целей. Так, например, если цель, как это было описано в разделе 2, состоит в обнаружении целевого объекта(ов), то эта цель передается в блок оценивания свойств, т.к. именно этот блок отвечает за решение задач такого рода.

Описанная система управления позволяет решить обобщённую задачу автоматизации

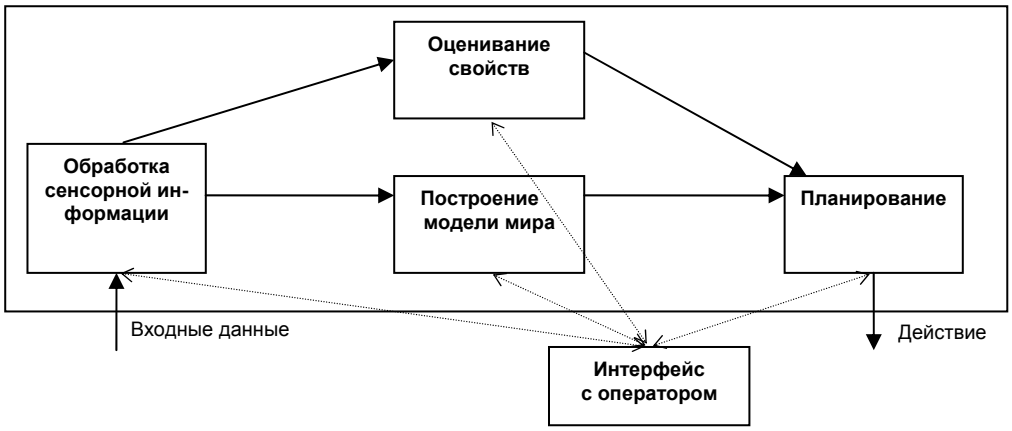

Рис. 1. Концептуальная архитектура тактического уровня системы управления БЛА

управления БЛА, рассматриваемую в работе (постановка в разделе 2) и при этом, в случае если обобщенная задача будет уточнена (например, если будет явно сформирован перечень возможных целей), предусматривает возможность детализации отдельных процессов на этапе программной реализации. Таким образом, встаёт вопрос разработки системы управления, предусматривающей на программном уровне возможность расширения и встраивания новых модулей «на лету». Очевидно, что для этого необходимо использование некоторой стандартизированной платформы с открытыми интерфейсами межмодульного взаимодействия. Предлагается в качестве такой платформы использовать свободно-распространяемую платформу Robotics Operationg System (ROS).

3. ROS – открытая программная платформа для создания систем управления

ROS (Robot Operating System) – это программная платформа (framework) разработки систем управления БТС, представляющая исследователям и разработчикам разнообразный функционал для организации распределенной работы модулей, управляющих движением БТС и обрабатывающих сигналы его датчиков. ROS реализует стандартные службы операционной системы, такие как: аппаратную абстракцию, низкоуровневый контроль устройств, реализацию часто используемых функций, передачу сообщений между процессами, управление пакетами, но с формальной точки зрения операционной системой не является.

Платформа ориентирована преимущественно на использование в UNIX-системах, для которых она поставляется как в виде уже собранных бинарных пакетов, так и пакетов исходных кодов для сборки. Существует также версия под операционные системы семейства Windows. Программное обеспечение ROS выпускается в соответствии с условиями BSD-лицензии [17] и с открытым исходным кодом, что обуславливает его широкое распространение и повсеместное использование в исследовательских проектах. В качестве примеров таких проектов можно привести:

- проект по разработке системы управления коллективом БЛА sFly, Государственный институт технологии Швейцарии, Цюрих [18];
- проекты по автоматизации управления БЛА, лаборатория Nimbus Университета Небраски-Линкольна, США [19];
- проект по разработке навигационного ПО для робота Pioneer 3-DX [20].

Платформа нашла поддержку и у производителей роботов. В [21] приведен перечень из более чем 50 робототехнических устройств, поддерживающих ROS. Среди них можно выделить такие известные проекты как Parrot AR.Drone, iRobot, Willow Garage Texai, Willow Garager PR2, Fraunhofer Care-o-bot 3, Pioneer 3-DX, Lego NXT.

Основными преимуществами платформы ROS по сравнению с аналогичным ПО являются:

1. открытый исходный код;

2. архитектура, поддерживающая параллельное асинхронное и синхронное взаимодействие программные компонент;

3. библиотека, состоящая из более, чем 2000 программных пакетов, каждый из которых направлен на решение одной или нескольких частных задача управления интеллектуальными автономными системами;

4. наличие инструментальных утилит для управления компонентами и их связями во время исполнения.

Таким образом, представляется целесообразным, во-первых, использовать платформу ROS для программной реализации системы управления БЛА, направленной на решение рассматриваемой в работе задачи (разделы 2, 3), во-вторых, более подробно рассмотреть принципы использования ROS для создания систем управления.

Основная единица ROS – пакет, который может быть объединен с другими пакетами в метапакет, структура которого аналогична пакету. Основой любого пакета являются файл, содержащий инструкции для сборки и установки пакета в конкретном операционном окружении, а также файл, содержащий метаданные пакета, такие как название, версия, разработчик, зависимости от других пакетов (могут быть обозначены как зависимости времени компиляции, так и времени исполнения). В дальнейшем в директорию, содержащую пакет добавляются файлы с исходным кодом, файлы, описывающие сообщения и сервисы и другие файлы, которые потребуются в процессе разработки данного пакета. Количество исполняемых файлов для пакета не ограничено.

Каждый из исполняемых файлов пакета представляет собой узел (node), который «общается» с другими узлами (не обязательно из одного и того же пакета), что проявляется в отсылке сообщений и обеспечении сервисов. Необходимо отметить, что запускаемый узел ROS всегда связывается с центральным узлом системы – ROS Core. Центральный узел обеспечивает мониторинг и управление другими узлами системы, которые могут быть запущены на различном вычислительном оборудовании, объединенном в единую сеть.

Таким образом, основной ROS является структурный подход к функциональной декомпозиции, параллельное и распределенное исполнение библиотечных модулей и возможность синхронного (посредством сервисов) и асинхронного (посредством тем) взаимодействия между ними.

При применении ROS для разработки программного комплекса навигации и управления БТС разработчик реализует основные функциональные модули комплекса как пакеты ROS, выделяет среди них те, которые должны работать распределено и параллельно и реализует протоколы взаимодействия между ними на базе задокументированных механизмов взаимодействия узлов в ROS. Отдельно стоит подчеркнуть возможность повторного использования существующих в библиотеке платформы узлов, реализующих различные алгоритмы навигации и управления БТС.

Покажем как описанная выше концептуальная архитектура тактического уровня системы управления (Рис. 1) может быть транслирована в схему системы управления на базе платформы ROS. На Рис. 2 показаны узлы и пакеты, которые можно выделить в программной реализации СУ БЛА и связи между ними: экспорт пакетов на уровне исходных кодов и взаимодействие между параллельно запущенными узлами на уровне посылки сообщений и вызова сервисов (на рисунке это отличие не отображено).

Приведем соответствие компонент тактического уровня системы управления (Рис. 1) и узлов и пакетов программной реализации СУ БЛА (Рис. 2).

Так компоненту «Обработка сенсорной информации» соответствуют узлы «Драйвер БПЛА», предоставляющий низкоуровневый интерфейс для управления БПЛА, «Захват видеопотока с камер БПЛА» и пакет «OpenCV», предоставляющий доступ к библиотеке функций компьютерного зрения OpenCV. Компонент «Оценивание свойств» соответствует узел «Обнаружение целевых объектов», а компоненту «Построение модели мира» - узел «Определение направления движения», поскольку данный узел строит модель окружающей сцены по данным оптического потока. Компонент «Планирование» отсутствует в представленной схеме

Рис. 2. Программная реализация системы управления (тактический уровень) на базе ROS

программной реализации, а компоненту «Интерфейс с оператором» соответствует узел «Интерфейс управления» и пакет «Библиотека графических компонентов».

Приведенная схема позволяет также продемонстрировать такое преимущество ROS как гибкость уже разработанная системы при внесении изменений. Эта гибкость обеспечивается за счет того, что все зависимости узлов друг от друга соответствуют четко определенным интерфейсам. Так, например, не составляет труда заменить узел «Определение направления движения», реализующий определение направления движения на основе алгоритма оптического потока по данным видеокамеры БЛА, на узел, реализующий более сложный алгоритм SLAM. При этом SLAM-узел сможет подписаться на сообщения узла драйвера БЛА для получения и последующего использования дополнительной информации от бортовых датчиков. Таким образом, выполняется требование, сформулированное в разделе 3 работы, – предусматривается на программном уровне возможность расширения и встраивания новых модулей «на лету», без необходимости внесения существенных изменений в существующую программную структуру системы управления.

4. Программный комплекс навигации и управления беспилотным летательным аппаратом на базе ROS

Авторы статьи в рамках выполнения НИР по государственному контракту № 14.514.11.4081 (Заказчик – Министерство образования и науки РФ) реализуют экспериментальный образец программного комплекса навигации и управления БЛА на базе свободно-распространяемого программного обеспечения. Основной особенностью разрабатываемого комплекса является возможность удаленного управления одновременно несколькими БЛА. При этом реализуемый принцип удаленного управления фактически основывается на предоставлении «управления» как облачного сервиса. В качестве БЛА используются квадрокоптеры Ar.Drone 2.0 [22]. Разработка программного комплекса ведется на базе описанной в разделе 4 платформы ROS. Опишем разрабатываемый комплекс более подробно.

Выделяются следующие слои распределенной архитектуры комплекса управления БЛА с программной точки зрения:

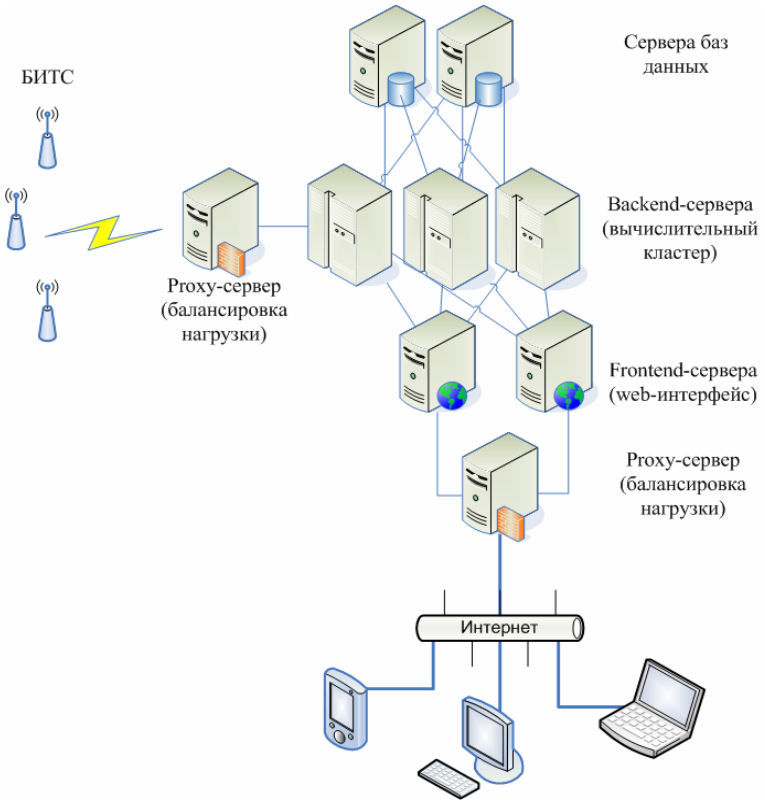

Рис. 3. Аппаратная архитектура распределенной облачной системы управления БЛА

1. слой хранения данных, ответственный за асинхронный прием данных от других слоев комплекса и сохранение их в базе данных, а также отправку запрашиваемых данных;

2. слой вычислительных модулей, ответственный за масштабируемую реализацию алгоритмов обработки данных от БЛА;

3. коммуникационный слой с БЛА, ответственный за реализацию интерфейсов и протоколов управления БЛА;

4. интерфейсный слой, ответственный предоставление пользователю удаленного доступа к комплексу управления БЛА через web-интерфейс;

5. слой, реализующий функции балансировки нагрузки, ответственный за распределение запросов пользователей между серверами управления БПЛА.

На Рис. 3 показана архитектура распределенной облачной системы управления БПЛА с указанием серверов каждого слоя. Каждый сервер, работая под управлением платформы ROS, реализует требуемую функциональность за счет множества специализированных узлов (node в терминологии ROS) и взаимодействия между ними по средствам асинхронной посылки сообщений (в терминологии ROS – topics) и синхронных вызовов (в терминологии ROS – service).

Взаимосвязь программных компонентов распределенной облачной системы управления БЛА показана на UML-диаграмме размещения компонент (Рис. 4).

Указанные на Рис. 4 компоненты выполняют функции соответствующих слоев, перечисленных выше. Компонент «Менеджер узлов ROS» используется как для распределения запросов пользователей к узлам ROS для управления БЛА, так и для мониторинга и управления узлами в ручном режиме при администрировании системы. Отметим, что на рисунке показаны типовые узлы системы. В процессе работы системы может быть запущено несколько экземпляров узлов, при этом компоненты, названия которых выделены на Рис. 4 курсивом, запускают только один раз (на первом из запускаемых узлов).

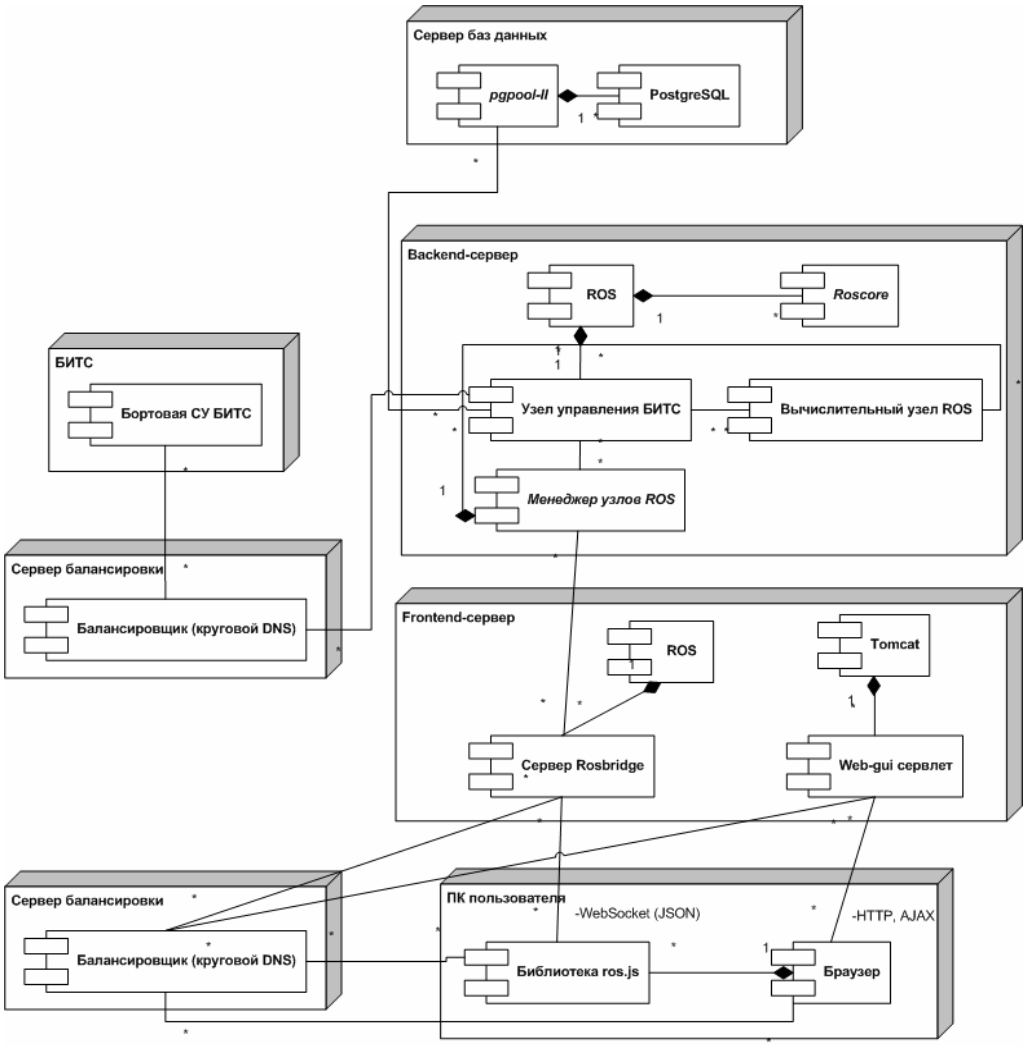

Рис. 4. UML-диаграмма размещения программных компонент распределенной облачной системы управления БЛА

Особенностью представленной архитектуры является технология построения Frontend-серверов слоя пользовательского интерфейса, которые реализуют web-интерфейс для удаленного взаимодействия пользователя с системой управления БЛА. В качестве web-сервера на Frontend-серверах может быть использован любой из распространенных web-серверов, например, Apache Tomcat. Web-сервер обслуживает запросы клиентских браузеров по протоколам HTTP и AJAX и генерирует HTML файлы с JavaScript и CSS-ресурсами. Эти файлы представляют собой web-интерфейс тонкого клиента системы управления БЛА. Взаимодействие тонкого клиента, полученного браузером пользователя от Frontend-серверов, и ROS-узлов вычислительных серверов осуществляется на основе технологии Rosbridge [23].

Поскольку узлы ROS работают параллельно, и предусматривается возможность программного создания, контроля процесса исполнения и удаления узлов ROS, то разрабатываемая программная архитектура фактически реализует принцип облачного доступа к услугам как сервисам. В описываемом случае услугой является удаленное управление БЛА (в виде постановки и контроля исполнения заданий через web-интерфейс).

В реализуемом программном комплексе предполагается наличие двух видов Proxy-серверов балансировки нагрузки. Во-первых, это сервер, балансирующий нагрузку, посту-

пающую на Frontend и Backend сервера со стороны пользователей комплекса. Во-вторых, это сервер, балансирующий нагрузку, поступающую на Backend сервера со стороны БПЛА.

Frontend-сервер генерирует по запросу пользователя страницы web-интерфейса, предоставляя ему удаленный доступ к комплексу, который заключается в реализации следующих стандартных для облачных сервисов функций: регистрация, авторизация пользователя в комплексе, функция выделения единицы БПЛА из пула свободных БПЛА, функция просмотра телеметрических данных о состоянии БПЛА, функция постановки задания в виде набора текстовых команд для исполнения БПЛА, и другие.

Интерфейсная часть данных функции (доступная через web-интерфейс) может быть реализована на одном из языков web-программирования (PHP, JavaScript, Ruby, Java и другие). Взаимодействие с узлами-серверами вычислительного кластера реализуется за счет библиотеки пакета Rosbridge. Данный пакет позволяет представить события, произошедшие в браузере пользователя, как сообщения, отправляемые удаленным ROS узлам в формате сообщений принятом в ROS. Также возможно обратное представление – ответные сообщения от ROS узлов транслируются в вызовы соответствующих обработчиков событий в web-интерфейсе.

Функция просмотра телеметрических данных о состоянии БЛА реализуется за счет асинхронных по отношению к Frontend-серверу запросов данных, хранящихся на серверах баз данных. При этом для отображения запрошенных телеметрических данных на странице пользователя целесообразно использовать технологию AJAX.

Функция постановки задания может быть реализована посредством трансляции набора текстовых команд в собственный предметно ориентированных язык комплекса, либо в программу на каком-либо скриптовом языке (например, языке Python).

Остановимся подобнее на функциях выделения и освобождения единицы БПЛА. Реализация этих функций может быть основана на различных принципах, так в качестве наиболее простого принципа может служить следующая стратегия выделения вычислительных ресурсов по запросу пользователя:

1. при поступлении запроса пользователя о выделении БЛА, формируется запрос менеджеру узлов ROS, реализованному в комплексе;

2. менеджер узлов по запросу о выделении БЛА проверяет наличие свободных БЛА после чего посылает сообщение сервису управления платформы ROS (на рисунке 4 сервис обозначен как «roscore») о создании и запуске на кластере вычислительных серверов нового узла ROS управления БЛА с соответствующими параметрами, которые позволяют связать созданный узел и БЛА;

3. после создания и запуска узла менеджер узлов сообщает уникальный идентификатор канала сообщения созданного узла узлу обработки команд пользователя и соответственно пользователь может наблюдать через web-интерфейс факт выделения в его управление БЛА.

Такой способ фактически предполагает крупногранулярный параллелизм со структурной декомпозицией. Другими вариантами организации параллельного управления БЛА и, соответственно, организации параллелизма вычислительных ресурсов в комплексе управления БЛА являются мелкозернистый параллелизм (на уровне отдельных потоков внутри узлов ROS) и параллелизм с функциональной декомпозицией. Последний означает, что вычислительные ресурсы выделяются не под новую единицу БЛА, доступ к которой предоставляется пользователю, а для выполнения новых вычислительных функций, например, алгоритмов обработки данных от БЛА или алгоритмов управления БЛА.

При относительно небольшом количестве управляемых БЛА (до 100 штук) целесообразно использовать крупногранулярный параллелизм со структурной декомпозицией, поскольку данный способ упрощает структуру комплекса, делает её более надежной с точки зрения локализации возможных ошибок одним узлом ROS. Однако, при это вычислительная система несет накладные расходы по производительности, связанные с дублированием параллельно запущенных вычислительных модулей, реализующих одну и ту же функциональность.

Функция контроля выполнения задания реализуется отдельным узлом ROS, запущенным для каждого БЛА на сервере вычислительного кластера. Этот узел вызывает необходимые алгоритмы обработки данных, поступающих от видеокамеры БЛА и бортовых датчиков и алгоритмы выработки команд управления БЛА согласно пользовательскому заданию. В случае не возможности выполнения пользовательского задания (например, отсутствия найденного на видеотоке целевого объекта или отсутствия возможности измерения расстояния до объектов сцены в данной позиции БЛА), узел сообщает о сложившейся ошибочной ситуации узлам Frontend сервера, которые в свою очередь уведомляют пользователя.

Таким образом, основные алгоритмы удаленного взаимодействия пользователя с комплексом управления БЛА реализуются на основе механизмов взаимодействия узлов ROS друг с другом и внешними системами. При этом использование платформы ROS позволяет: упростить разработку системы (за счет использования множества готовых компонент), её сопровождение (за счет готовой архитектуры и утилит поддержки), её расширение (за счет поддержки функциональной декомпозиции и декомпозиции на распределенные, параллельно функционирующие узлы, зависящие друг от друга по четко специфицированным интерфейсам).

Заключение

В настоящее время наблюдается значительный интерес со стороны исследовательского сообщества (в таких областях как робототехника, искусственный интеллект, теория автоматического управления и др.) к созданию беспилотных транспортных систем (средств) нового поколения, а именно – к созданию беспилотных транспортных средств, способных к автономному функционированию в окружающей среде. Создание систем управления такими средствами подразумевает необходимость исследования методов и подходов к решению ряда вопросов связанных как с концептуальной организацией архитектуры таких систем, так и непосредственно с их программной реализацией. Именно этим вопросам и посвящена данная работа.

По результатам проведенного исследования в работе предложена концептуальная модель архитектуры тактического уровня управления беспилотным транспортным средством (беспилотным летательным аппаратом), а также способ программной реализации этой модели, основанный на использовании открытого программного обеспечения ROS (Robotics operating system). Предложенное решение (архитектура и ее программная реализация) подразумевает возможность расширять функционал системы «на лету», без внесения существенных изменений в разработанную концептуальную архитектуру и программную структуру, что, в конечном итоге, позволяет использовать описанные методы и подходы для решения широкого круга как теоретических, так и практических задач в области интеллектуального управления беспилотными транспортными средствами.

Литература

1. Albus J. et al. 4D/Real-time Control System (4D/RCS): A Reference Model Architecture for Unmanned Vehicle Systems v2.0, NIST, NISTIR 6910, 2002
2. Jameson S., Franke J., Szczerba R., Stockdale S. Collaborative Autonomy for Manned/Unmanned Teams. AHS International Forum 61. Grapevine. TX. 2005
3. Осипов Г.С., Тихомиров И.А., Хачумов В.М., Яковлев К.С. Интеллектуальное управление транспортными средствами: стандарты, проекты, реализации // Авиакосмическое приборостроение. 2009. № 6. с. 34–43.
4. Осипов Г.С. Методы искусственного интеллекта и задачи управления. Пленарные доклады международной мультиконференции «Теория и системы управления». М.: ИПУ РАН. 2009. с. 161–170.
5. Яковлев К.С., Макаров Д.А., Панов А.И., Зубарев Д.В. Принципы построения многоуровневых архитектур систем управления беспилотными летательными аппаратами // Авиакосмическое приборостроение, 4, 2013. с. 10-28.
6. Documentation – ROS Wiki [Электронный ресурс] // Официальный сайт программного обеспечения Robotics Operating System. URL: http://www.ros.org/. (Дата обращения: 20.07.2013)
7. Quigley M., Conley K., Gerkey B., Faust J., Foote T., Leibs J., Ng A. Y. ROS: an open-source Robot Operating System. ICRA workshop on open source software (Vol. 3, No. 3.2). 2009.
8. Durrant-Whyte H., Bailey T. Simultaneous localization and mapping: part I // Robotics & Automation Magazine, IEEE, 13(2), 2006. pp. 99-110.
9. Borrmann D., Elseberg J., Lingemann K., Nuchter A., Hertzberg J. Globally consistent 3D mapping with scan

matching // Robotics and Autonomous Systems. 2008. Vol. 56, p. 130–142.
10. Steder B., Grisetti G., Stachniss C., Burgard W. Visual SLAM for flying vehicles. IEEE Transactions on Robotics, 24(5). 2008. pp. 1088-1093.
11. Huang A. S., Bachrach A., Henry P., Krainin M., Maturana D., Fox D., Roy N. Visual odometry and mapping for autonomous flight using an RGB-D camera // In Int. Symposium on Robotics Research (ISRR),(Flagstaff, Arizona, USA), 2011.
12. Bishop C. M., Nasrabadi, N. M. Pattern recognition and machine learning, (Vol. 1, p. 740). 2006. New York: Springer.
13. Viola P., Jones M. Robust real-time object detection // International Journal of Computer Vision, 4, 2001.
14. Яковлев К.С., Баскин Е.С. Графовые модели в задаче планирования траектории на плоскости // Искусственный интеллект и принятие решений, 1, 2013. С.5-12.
15. Korf R.E., Artificial intelligence search algorithms, CRC Handbook of Algorithms and Theory of Computation, M.J. Atallah (Ed.). Boca Raton, FL: CRC Press, 1998.
16. Яковлев К.С., Макаров Д.А. Интеллектуальные технологии и методы управления беспилотными летательными аппаратами. Труды конференции «Системный анализ и информационные технологии САИТ-2013». Принято к печати.
17. BSD licenses [Электронный ресурс] // Список и разъяснения по актуальным лицензиям BSD. URL: http://en.wikipedia.org/wiki/BSD_licenses. (Дата обращения: 20.07.2013)
18. Start – sFly: Swarm of micro FLYing robots // Официальный сайт проекта sFly: Swarm of micro FLYing robots. URL: http://www.sfly.ethz.ch/. (Дата обращения: 20.07.2013)
19. Projects | Nimbus Lab // Официальный сайт проектов лаборатории Nimbus. URL: http://nimbus.unl.edu/projects/. (Дата обращения: 20.07.2013)
20. Zaman S., Slany W., Steinbauer G. ROS-based mapping, localization and autonomous navigation using a pioneer 3-DX robot and their relevant issues. In Electronics, Communications and Photonics Conference (SIECPC), 2011 Saudi International. 2011. pp. 1-5.
21. Documentation – ROS Wiki [Электронный ресурс] // Список роботизированных платформ, использующих ROS. URL: http://www.ros.org/wiki/Robots. (Дата обращения: 20.07.2013)
22. Bristeau P.-J., Callou F., Vissiere D., Petit N. The Navigation and Control Technology Inside the AR.Drone Micro UAV // Preprints of the 18th IFAC World Congress (Milano, Italy). Milano: IFAC. 2011. Vol. 18. Part 1. p. 1477–1484.
23. Rosbridge // Официальный сайт проекта Rosbridge. URL: http://rosbridge.org/. (Дата обращения: 20.07.2013).

Яковлев Константин Сергеевич. Научный сотрудник ООО "Технологии системного анализа", старший научный сотрудник лаборатории интеллектуальных динамических систем ИСА РАН. Автор 20 научных работ. Область научных интересов: искусственный интеллект, робототехника, когнитивные науки, интеллектуальные динамические системы, интеллектуальные системы управления, планирование, планирование траектории, моделирование целенаправленного поведения, эвристический поиск. E-mail: yakovlev@isa.ru

Петров Александр Викторович. Генеральный директор ООО "НПП САТЭК плюс", ведущий инженер-программист РГАТУ имени П.А.Соловьева. Автор 10 научных работ. Область научных интересов: искусственный интеллект, робототехника, когнитивные науки, интеллектуальные динамические системы, интеллектуальные системы. E-mail: gmdidro@gmail.com

Хитьков Всеволод Владимирович. Аспирант РГАТУ имени П.А. Соловьева. Область научных интересов: цифровая обработка изображений, распознавание образов, интеллектуальные динамические системы. E-mail: vskhitkov@gmail.com