МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по научно-исследовательской работе

Тема: Исследование алгоритмов планирования траектории квадрокоптера на основе Moveit

Студентка гр. 3304 _____ Диогенова К.А.

Руководитель _____ Филатов А.Ю.

Санкт-Петербург

2018

ЗАДАНИЕ

НА НАУЧНО-ИССЛЕДОВАТЕЛЬСКУЮ РАБОТУ

Студентка Диогенова К.А.

Группа 3304

Тема НИР: Исследование алгоритмов планирования траектории квадрокоптера на основе MoveIt!

Задание на НИР:

Определить принцип работы (полета) квадракоптера. Определить и изучить инструменты для разработки алгоритмов планирования алгоритмов. Найти и провести обзор существующих решений поставленной задачи.

Сроки выполнения НИР: 01.10.2018 – 20.12.2018

Дата сдачи отчета: 19.12.2018

Дата защиты отчета: 20.12.2018

Студентка гр. 3304 _____ Диогенова К.А.

Руководитель _____ Филатов А.Ю.

АННОТАЦИЯ

Основной целью научной исследовательской работы является разработка алгоритма планирования траектории полета квадракоптера. В настоящей работе приводится обзор инструментов с помощью которых будет происходить планирование траекторий, а также симуляция работы.

SUMMARY

The main goal of the scientific research work is the development of the algorithm for planning the trajectory of the flight of a quadcopter. This paper provides an overview of the tools by which trajectory planning will take place, as well as a simulation of the work.

ВВЕДЕНИЕ

Специальные мобильные роботы являются самыми распространенными видами роботов, способными свободно перемещаться и адаптироваться к условиям окружающей среды. Однако для этого необходимо получать информацию об окружающем мире и сопоставлять её с данными о собственном положении при планировании траектории, с целью определения последующего шага движения. Это задача называется планированием траектории.

Современные беспилотные летательные аппараты (БПЛА) способны не только выполнить команды, поступающие из центра управления, регламентирующие параметры движения (азимут, скорость, ускорение), но и самостоятельно планировать траекторию достижения целевого состояния, учитывая необходимость также решать задачи обхода препятствий. При этом ориентация в пространстве обеспечивается навигационной системой, использующей GPS/ГЛОНАСС, дополненной средствами системы ориентации: технического зрения, дальномером, инфракрасными датчиками и т.д. Достижение целевого положения осуществляется по некоторой траектории, для предварительного планирования которой необходима информация о внешней среде/пространстве, в котором будет производиться движение. Эти сведения доставляет карта местности, которая может быть получена из различных источников. Однако такая карта не может учитывать возможные изменения, обусловленные перемещением мобильных устройств (транспорт, люди), природных явлений (осадки), антропогенного воздействия на ландшафт, сезонных изменений природы и др. Тем не менее общее представление о местности, доставляемое картой, вполне достаточное, для осуществления обобщённого (предварительного) планирования траектории движения.

1. МАТЕМАТИЧЕСКИЙ АППАРАТ

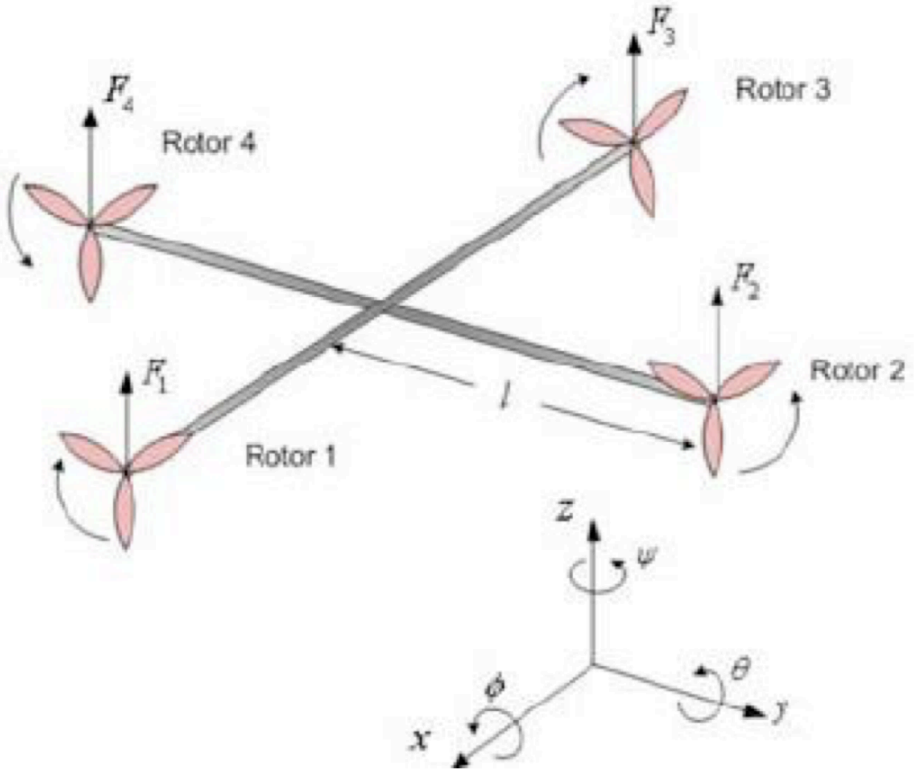

Рисунок 1 – «Квадракоптер»

Каждый из двигателей создает тягу и момент вращения. Для того чтобы скомпенсировать моменты вращения, пара двигателей 1 и 3 вращаются по часовой стрелке, а роторы 2 и 4 – против часовой. Тем самым моменты, создаваемые парой двигателей 1 и 3, компенсируются моментами второй пары – 2 и 4.

Вращения квадрокоптера производится путем изменения скоростей вращения двигателей. Для изменения крена необходимо увеличить угловую скорость вращения двигателя 2 и уменьшить скорость вращения 4. Аналогичным образом осуществляется изменение тангажа. Изменения рысканья производится сложнее: увеличивается скорость вращения двигателей 1-3 и уменьшается скорость вращения 2-4. Изменение угловой скорости двух винтов (пар винтов) необходимо для того, чтобы сохранять общую тягу, создаваемую 4-мя винтами неизменной.

Принятые допущения:

- квадрокоптер симметричен относительно осей x и y;
- рама квадрокоптера и его винты абсолютно жесткие;
- каждый двигатель располагается на конце стержня;
- тяга, создаваемая каждым винтом, перпендикулярна плоскости x-y.

2. ИНСТРУМЕНТ: ROBOT OPERATING SYSTEM

2.1. Справочная информация

ROS (Robot Operating System) представляет собой надстройку над операционной системой, которая позволяет разрабатывать системы управления роботами. По сути, ROS — это набор из различных библиотек, таких как:

- OpenCV — библиотека, содержащая алгоритмы компьютерного зрения и обработки изображений;
- PCL- библиотека для работы с облаками 3D-точек;
- Ogre — объектно-ориентированный графический движок с открытым исходным кодом;
- Orocos — библиотека для управления роботами (например, расчет кинематики).

Также в ROS входят драйвера для различных манипуляторов и сенсоров (включая MS Kinect). Основополагающим преимуществом ROS является клиент-серверная архитектура — разработчики реализовали механизм пересылки сообщений между различными объектами, возможность построения распределенных систем, предоставление bridge'ей я языками C++, Python.

Сегодня ROS стабильно устанавливается и работает только на Ubuntu версии от 10 и выше

ROS обеспечивает стандартные службы операционной системы:

- аппаратную абстракцию,
- низкоуровневый контроль устройств,
- реализацию часто используемых функций,
- передачу сообщений между процессами,
- управление пакетами.

ROS, также предоставляет инструменты и библиотеки для получения, построения, написания и выполнения кода на нескольких компьютерах. ROS основан на архитектуре графов, где обработка данных происходит в узлах, которые могут получать и передавать сообщения между собой.

2.2. Основные преимущества ROS

ROS представляет собой распределенную сеть процессов (Узлов — Nodes), что позволяет разрабатывать их индивидуально.

Эти процессы могут быть сгруппированы в Пакеты и Стеки (Packages and Stacks), которые можно легко распространять.

Также, ROS поддерживает федеративные системы репозиториев кода.

Такой дизайн, от уровня файловой системы на уровень сообществ, позволяет возможность независимо принимать решения о разработке и реализации, но все они могут быть объединены вместе, средствами инфраструктуры ROS.

Существует ещё несколько других целей в рамках ROS:

- Тонкий: ROS должен быть как можно тоньше — чтобы код, написанный для ROS мог быть использован и в других программных системах. Следствием этого является то, что ROS легко интегрируется с другими системами программного обеспечения роботов: ROS уже интегрирован с OpenRAVE, Orocos и Player.
- Библиотеки: предпочтительной моделью развития является написание библиотек с чистым функциональным интерфейсом.
- Независимость от языка: структуру ROS легко реализовать на любом современном языке программирования. ROS уже реализован на: Python, C++, Lisp и экспериментальные библиотеки на Java и Lua.

- Простое тестирование: ROS имеет встроенный фреймворк для тестирования — rostest, что позволяет легко тестировать приложения.
- Масштабирование: ROS подходит для больших систем выполнения и для большого процесса разработки.

2.3. Основные понятия

Пакетом (package) называется наименьшая единица. Представляет собой директорию, содержащую в себе какие-либо данные, библиотеки, исполняемые и конфигурационные файлы и т.д. и т.п., логически объединенные в какой-то полезный модуль. Цель такого структурирования совершенно прозрачна — повышение пользовательского использования и возможности повторного использования.

Структура пакета выглядит следующим образом:

- bin/: скомпилированные бинарники
- include/package_name: заголовочные файлы для C++ (обязательно должны описываться в manifest.xml)
- msg/: типы сообщений
- src/package_name/: исходный код на C++ и скрипты на Python'e, экспортируемые в другие пакеты
- srv/: типы сервисов, предоставляемых пакетом
- scripts/: скрипты на Python'e
- CMakeLists.txt: CMake файл для сборки пакета
- manifest.xml: манифест пакета
- mainpage.dox: Doxygen-документация

В свою очередь, пакеты объединяются в стэки. На рис. 2 представлен пример такой структуры.

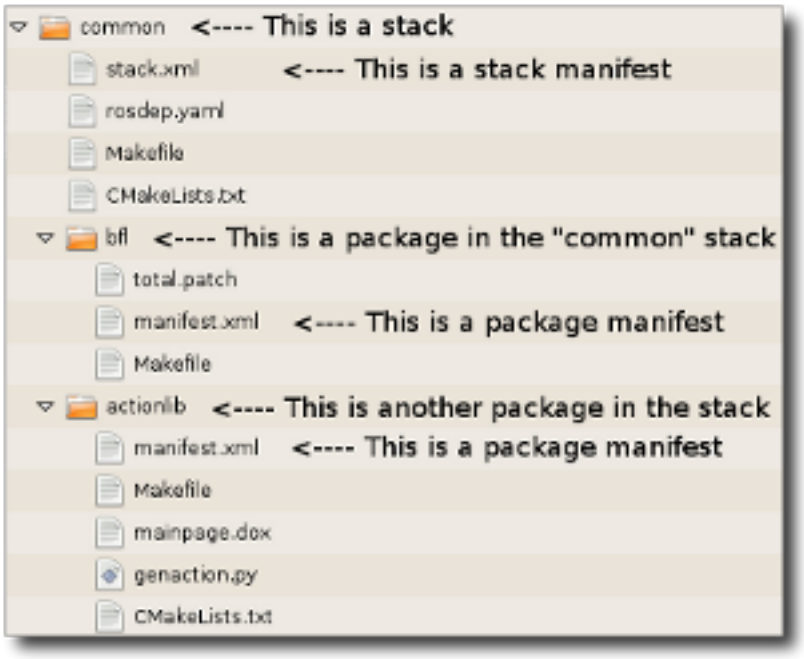

Рисунок 2 – «Стэк»

Динамическая составляющая в ROS описывается узлами(node) и шинами(topic). Узел — это запущенный процесс, который умеет общаться с другими процессами. Шина — именованный канал, соединяющая различные узлы. Узлы и шины формируют асинхронный механизм обмена данными.

3. MOVEIT – ИНСТРУМЕНТ ПЛАНИРОВАНИЯ ТРАЕКТОРИЙ ПОЛЕТА КВАДРАКОПТЕРА

MoveIt! представляет собой современное программное обеспечение для мобильных манипуляций, включающее последние достижения в планировании движения, манипуляциях, восприятии 3D, кинематике, управлении и навигации. Он представляет собой простую в использовании платформу для разработки современных приложений для робототехники, оценки новых конструкций роботов и создания интегрированных продуктов робототехники для промышленных, коммерческих, научно-исследовательских и других областей.

Основной узел, используемый в MoveIt! является узлом move_group. Как показано на рис. 3 системной архитектуры, узел move_group объединяет все внешние узлы, чтобы обеспечить набор действий и сервисов ROS для пользователей. Эти внешние узлы включают в себя датчики и контроллеры квадрокоптера, данные с сервера параметров и пользовательский интерфейс.

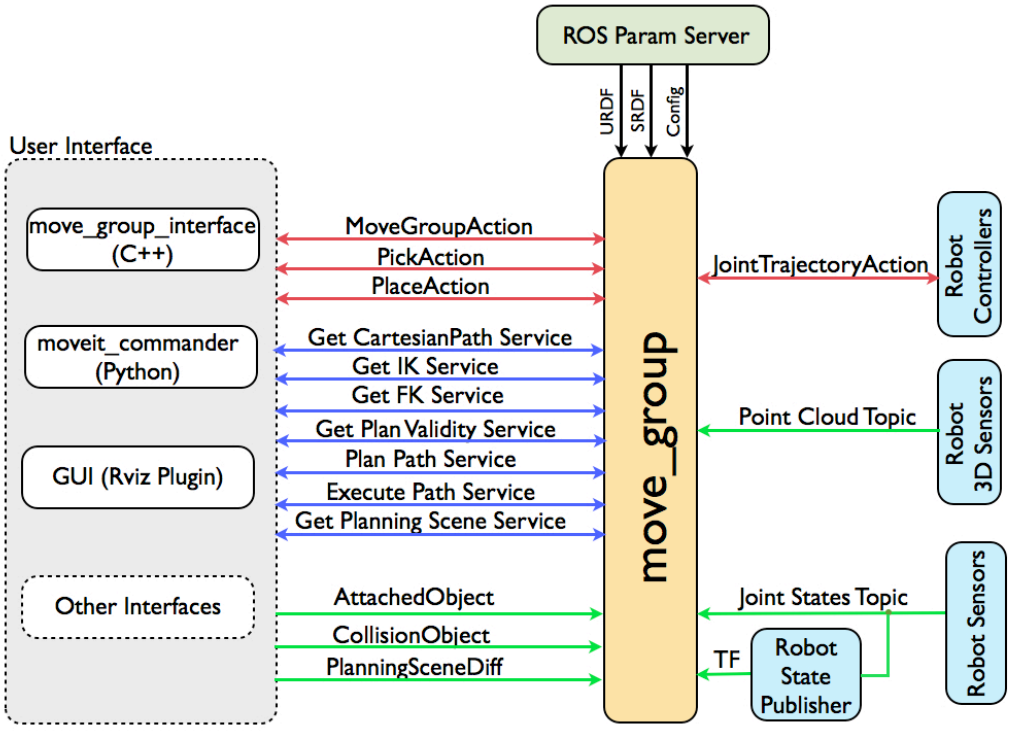

Рисунок 3 – «Системная архитектура»

Общая концепция MoveIt! заключается в определении групп узлов и других элементов для выполнения перемещающихся действий с использованием алгоритмов планирования движения. Эти алгоритмы рассматривают сцену с объектами для взаимодействия и совместными характеристиками группы. MoveIt! предоставляет три основных интерфейса для доступа к действиям и службам, предоставляемым узлом move_group.

Пакет move_group_interface предоставляет интерфейс C++. Пакет moveit_commander поддерживает Python, и плагин Motion Planning Rviz предоставляет графический интерфейс. Узел move_group связывается с квадрокоптером, используя темы и действия ROS. Узел получает текущую информацию о состоянии, такую как положение и ориентацию квадрокоптера, прослушивая joint_states topic. Поэтому необходимо запустить узел joint_state_publisher для трансляции состояния квадрокоптера. Узел move_group также получает глобальную информацию о нахождении квадрокоптера, используя библиотеку ROS TF.

TF обеспечивает преобразование между базовым фреймом робота и картой. Для публикации этой информации выполняется узел robot_state_publisher. Узел move_group взаимодействует с системой управления квадрокоптреа через интерфейс FollowJointTrajectoryAction. На квадрокоптере должен обслуживать клиента move_group, чтобы получать управляющие команды, выводимые узлом move_group. Действие FollowJointTrajectoryAction будет изменено для соответствия динамике квадрополя с несколькими степенями свободы (DOF). Наконец, узел move_group поддерживает сцену планирования, используя датчики квадрокоптера в качестве входных данных. Сцена планирования является представителем текущего состояния квадрокоптера и его наблюдаемой среды.

Наконец, узел move_group использует сервер параметров ROS для получения информации о конфигурации. Это включает в себя URDF и SRDF квадрокоптера, а также другую конфигурационную информацию, включая границы узлов, переменные планирования движения и информацию о датчике.

SRDF, а также файлы конфигурации автоматически генерируются с помощью MoveIt! Интерфейсы плагина обеспечивают доступ к нескольким библиотекам планирования движения, в основном OMPL, доступным через действие или службу ROS.

OMPL (Open Motion Planning Library) - библиотека планирования движения с открытым исходным кодом, которая в основном реализует рандомизированные планировщики движения. MoveIt! использует планировщиков движения из этой библиотеки в качестве своего стандартного набора планировщиков. Пользователь инициирует запрос плана движения, устанавливая желаемую цель для квадрокоптера, и планировщик движения вычисляет траекторию. Узел move_group будет генерировать желаемую траекторию, которая перемещает квадрокоптер в желаемую позицию без сбоев в любые препятствия, идентифицированные датчиками или нарушающие ограничения скорости и ускорения модели квадрокоптера.

3D восприятие в MoveIt! обрабатывается монитором карты занятости. Монитор карты занятости использует архитектуру плагина для обработки входных сигналов различных типов, как показано на рис. 4. В частности, MoveIt! имеет встроенную поддержку для обработки двух видов входов:

- Point Cloud Topic: обрабатываются плагином обновления плагина облачного облака;
- Depth Image Topic: обработанные плагином обновления изображения глубины изображения.

Также можно добавлять собственные типы обновлений в качестве плагина к монитору карты занятости.

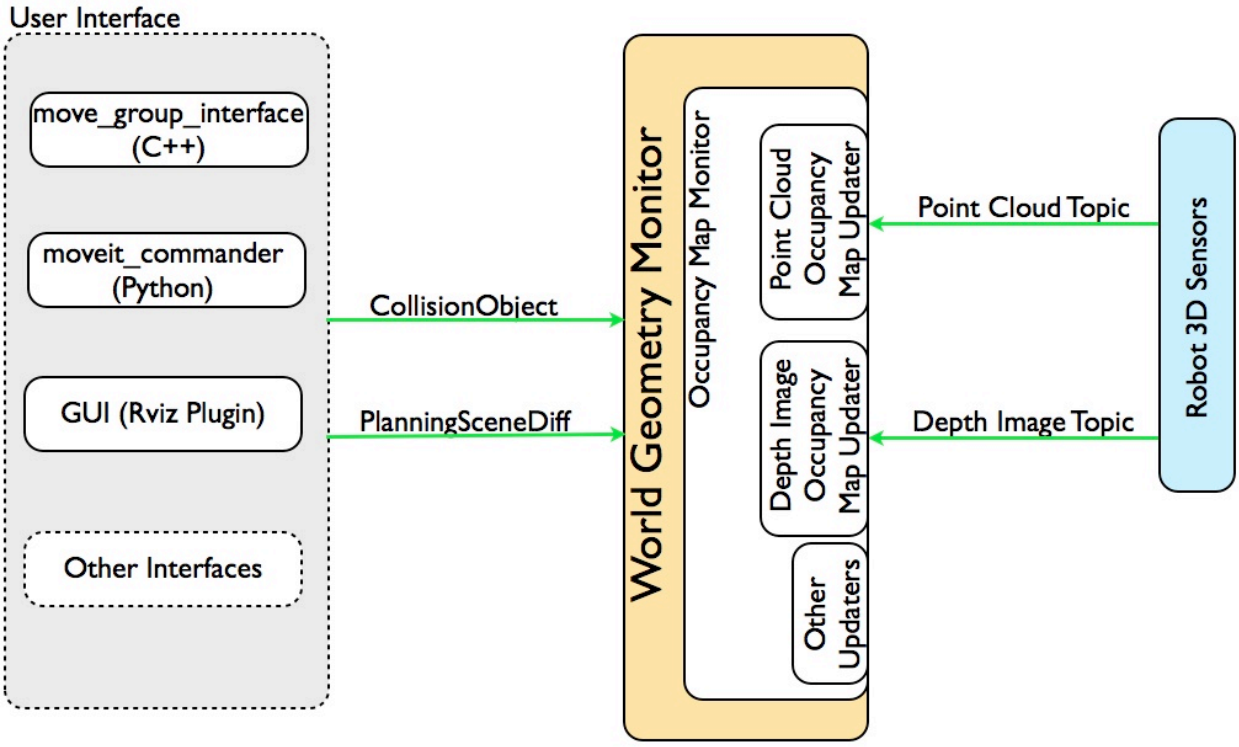

Рисунок 4 – «3D восприятие»

MoveIt! – первоначально пакет для манипуляций с роботами. Поэтому программное обеспечение по умолчанию поддерживает манипуляции с захватом и совместные траектории и предполагает, что у робота есть контроллеры, которые могут успешно принимать эти формы управляющих входов. Поскольку мы адаптируем программное обеспечение MoveIt! для квадротора с несколькими степенями свободы, нам нужно будет внести некоторые незначительные дополнения в MoveIt!. Это включает в себя изменение MoveIt! Simple Controller Manager и добавление сервера действий для взаимодействия между MoveIt! и контроллеры положения квадротора. ROS использует службы в ситуациях, когда узлам необходимо обмениваться запросами простых задач и обратной связью между собой. Для более сложных ситуаций ROS использует пакет actionlib для создания серверов, выполняющих долговременные задачи. Это полезно для задач, которые могут занять много времени. Это позволяет пользователю или узлу отменить запрос задачи во время выполнения или получить периодическую обратную связь о том, как продвигается запрос. Эта платформа клиент-сервер используется в MoveIt! для предзапускаемых задач, таких как захваты рулевого управления. Узел

move_group использует интерфейс действия для связи с контроллерами на квадрокоптере. Узел move_group предоставляет клиенту действия сообщение с сервером действий контроллера на квадрокоптере.

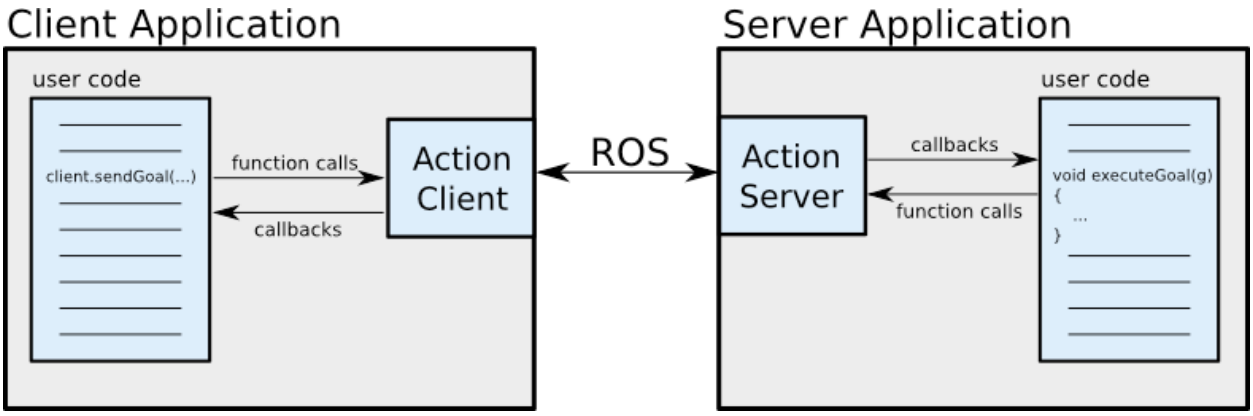

Рисунок 5 – «Взаимодействие клиента и сервера»

ROS использует предопределенный протокол действия для обеспечения взаимодействия между клиентом и сервером действия. Этот протокол действия зависит от нескольких тем ROS в указанном пространстве имен ROS для передачи сообщений. Основная тема использует автогенерированное сообщение ActionGoal для отправки новых задач на сервер. В теме отмены используются сообщения actionlib_msgs / GoalID, позволяющие клиентам отправлять запросы на отправку действий на сервер. В теме статуса используется actionlib_msgs / GoalStatusArray, чтобы клиентские действия могли получать информацию о состоянии каждой цели, которая в настоящее время отслеживается сервером. В разделе отзывов используется автоматическое сообщение ActionFeedback, чтобы предоставить серверам возможность отправлять периодические обновления клиентам во время обработки цели. В теме результата используется автоматическое сообщение ActionResult для предоставления серверам способа отправки информации клиентам действий по завершении цели.

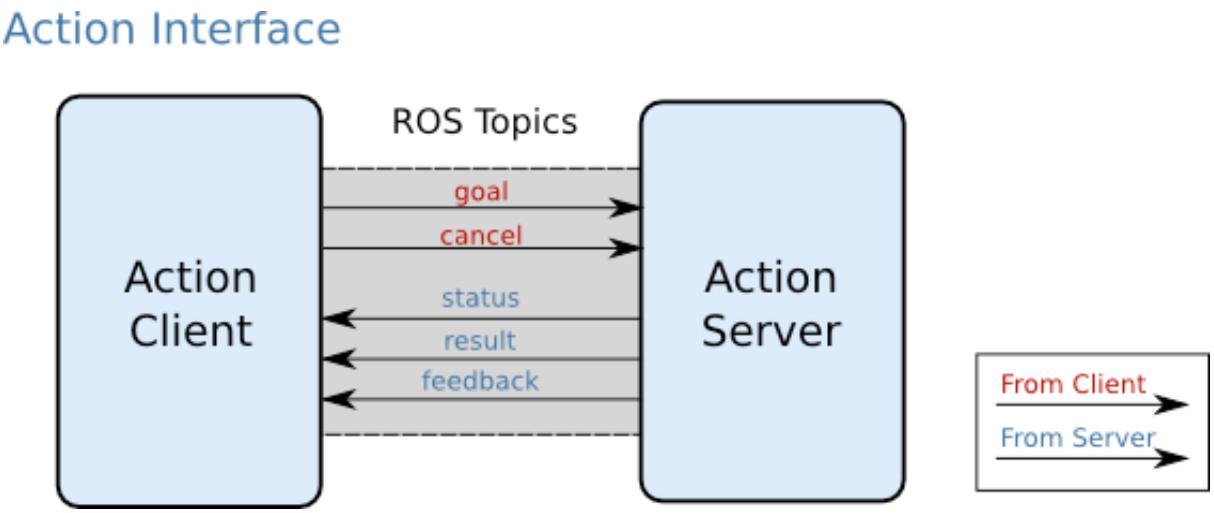

Рисунок 6 – «Интерфейс взаимодействия»

4. СУЩЕСТВУЮЩИЕ АНАЛОГИ

Сегодня существует ряд методов планирования траектории. Основными являются следующие подходы: планирование с использованием энкодеров; применение инерционных датчиков; подходы на базе системы технического зрения и генерирование траектории на основе данных навигационных и локационных систем ГЛОНАСС или GPS.

Эти подходы были успешно использованы для планирования и генерирования траектории движения робота. Однако в зависимости от вида робота и миссии движения каждый из этих подходов показал различные результаты по универсальности использования.

Когда речь идет о планировании траектории воздушных специальных роботов, то есть о беспилотных летательных аппаратах, то все эти методы используются в зависимости от миссии полета, степени автономности движения и состояния окружающей среды. Как показывает анализ литературы, отсутствует конкретная схема соотношения БПЛА с выбранной миссией полета. Это привело к появлению многих подклассов БПЛА с различными степенями автономности движения, аэродинамических характеристик и управляемости.

Вопрос определения и классификации БПЛА, как разновидности летательных аппаратов, изменялся параллельно с процессами развития техники и технологии. Существуют разные мнения по определению БПЛА, но основные из них два: классификация по функциональности и системности.

ЗАКЛЮЧЕНИЕ

В ходе выполнения научно-исследовательской работы были сформирована тема исследования, определены цели, задачи и составлена программа для ее выполнения. Также была изучена предметная область и инструменты, которые будут использованы для реализации алгоритмов планирования траекторий полетов квадракоптера.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Ардентов А.А., Бесчастный И.Ю., Маштаков А.П. и др. Алгоритм вычисления положения БПЛА с использованием системы машинного зрения, Программные системы: теория и приложения, Электронный журнал. 2012. №3(12). С. 23-29.
2. Макаров И.М., Лохин В.М., Манько С.В. и др., Интеллектуальные системы управления беспилотными летательными аппаратами на основе комплексного применения технологий нечеткой логики и ассоциативной памяти., Авиакосмическое приборостроение., 2002.
3. Осипов Г.С., Тихомиров И.А., Хачумов В.М., Яковлев К.С., Интеллектуальные системы управления автономными транспортными средствами: стандарты, проекты, реализация., Авиакосмическое приборостроение, 2009.
4. Степанов Д.Н., Тищенко И.П. Задача моделирования полета беспилотного летательного аппарата на основе системы машинного зрения., Программные системы: теория и приложения. Электронный журнал. 2011, №4(8).
5. О.В. Даринцев, А.Б. Мигранов, Использование нейронной карты для планирования траектории мобильного робота, Искусственный интеллект. 2009. № 3.
6. Ю.В.Чернухин, С.Н. Писаренко, А. А. Приемко, Нейросетевая система навигационной безопасности транспортных объектов в наземной, подводной, надводной и воздушной средах, Искусственный интеллект., 2006. № 3.
7. Santos O., Romero H., Salazar S, Lozano R. Real-time Stabilization of a Quadrotor UAV: Nonlinear Optimal and Suboptimal Control, Journal of Intelligent & Robotic Systems. 2013. Vol. 70, iss. 1-4. P. 79-91. DOI: 10.1007/s10846-012-9711-8

8. Li T., Zhang Y., Gordon B.W. Passive and active nonlinear fault tolerant control of a quadrotor unmanned aerial vehicle based on the sliding mode control technique, Journal of Systems and Control Engineering. 2013. Vol. 227. P. 12-23. DOI: 10.1177/0959651812455293
9. Dierks T, Jagannathan S. Output feedback control of a quadrotor UAV using neural net- works // IEEE Transactions on Neural Networks. 2010. Vol. 21, no. 1. P. 50-66. DOI: 10.1109/TNN.2009.2034145
10. Управление и наведение беспилотных маневренных летательных аппаратов на основе современных информационных технологий. / Под ред. М.Н. Красильникова, Т.Г. Серебрякова. М.: Физматлит, 2005. 280 с.
11. Гэн К., Чулин Н. А., Алгоритмы стабилизации для автоматического управления траекторным движением квадрокоптера, Наука и образование МГТУ им. Н.Э.Баумана, электронный журнал, 2015, №05.