УДК 681.5.01
DOI 10.25205/1818-7900-2020-18-3-19-33

Разработка архитектуры полетного контроллера для квадрокоптера на базе одноплатного компьютера Raspberry Pi

Д. А. Липовый [1], **А. С. Мальцев** [2]

[1] *Новосибирский государственный университет*
Новосибирск, Россия
[2] *Институт автоматики и электрометрии СО РАН*
Новосибирск, Россия

Аннотация

Данная работа посвящена разработке модульной архитектуры полётного контроллера для квадрокоптера. В качестве аппаратной части контроллера применяется одноплатный компьютер Raspberry Pi, при разработке программной части используется фреймворк Robot Operating System (ROS). В работе представлено описание разработанной архитектуры, использованных алгоритмов управления, результаты численных полетных экспериментов в физическом симуляторе Gazebo.

Ключевые слова

квадрокоптер, полётный контроллер, Robot Operating System (ROS)

Для цитирования

Липовый Д. А., Мальцев А. С. Разработка архитектуры полетного контроллера для квадрокоптера на базе одноплатного компьютера Raspberry Pi // Вестник НГУ. Серия: Информационные технологии. 2020. Т. 18, № 3. С. 19–33. DOI 10.25205/1818-7900-2020-18-3-19-33

Development Quadcopter Flight Controller Architecture Based on a Single-Board Computer Raspberry Pi

D. A. Lipoviy [1], **A. S. Maltsev** [2]

[1] *Novosibirsk State University*
Novosibirsk, Russian Federation
[2] *Institute of Automation and Electrometry SB RAS*
Novosibirsk, Russian Federation

Abstract

This work is devoted to development of a modular flight controller architecture for a quadcopter. The hardware part of the controller is a single-Board raspberry Pi computer, for developing the software part Robot Operating System (ROS) framework was used. The paper describes the developed architecture, control algorithms, results of flight experiments in the Gazebo physical simulator.

Keywords

quadcopter, flight controller, Robot Operating System (ROS)

For citation

Lipoviy D. A., Maltsev A. S. Development Quadcopter Flight Controller Architecture Based on a Single-Board Computer Raspberry Pi. *Vestnik NSU. Series: Information Technologies*, 2020, vol. 18, no. 3, p. 19–33. (in Russ.) DOI 10.25205/1818-7900-2020-18-3-19-33

© Д. А. Липовый, А. С. Мальцев, 2020

Введение

Во всём мире использование беспилотных летательных аппаратов (БПЛА) получило широкое распространение благодаря большому количеству решаемых задач. Особой популярностью пользуются квадрокоптеры, благодаря простоте конструкции, легкости управления и низкой цене основных компонент. Они применяются в задачах наблюдения и мониторинга на большой территории, картографирования и проведения геологических и геодезических измерений, транспортирования грузов, отработки и демонстрации технологий группового управления [1–4].

Главным составляющим БПЛА является полётный контроллер – электронное устройство, содержащее набор определённых элементов и датчиков. Его задача – выполнение базовых команд управления летательным аппаратом, таких как стабилизация углов ориентации, навигация, автоматический полёт по путевым точкам на заданной высоте.

Существующие на данный момент квадрокоптеры настроены и оптимизированы под решение различных задач, с которыми они хорошо справляются, однако из-за узкой специализации они имеют ограниченную расширяемость и неполную документацию. Это ограничивает область их применения в научных исследованиях, поскольку их модификация связана с дополнительными трудозатратами. Для проведения исследовательских работ необходим полётный контроллер, который обладает запасом производительности, возможностью лёгкой модификации программного обеспечения для тестирования новых алгоритмов полёта и решения нетиповых задач. Также он должен обеспечивать удобные для разработчика инструменты создания и отладки программного обеспечения с широким внедрением моделирования в процесс разработки и тестирования программных модулей.

Целью данной работы является разработка модульной архитектуры, явно разграничивающей функции полётного контроллера, которая позволяет легко модифицировать его отдельные части. Для разработки используются фреймворк Robot Operating System (ROS) и одноплатный компьютер Raspberry Pi. В статье предложена архитектура полётного контроллера, представлено описание программных компонент и алгоритмов, полученные результаты экспериментов с инерциальной навигацией и управлением полета квадрокоптера в виртуальном окружении.

Характеристика существующих решений

В настоящее время существуют разнообразные коммерческие и открытые полётные контроллеры, обладающие различными функциями и характеристиками. Рассмотрим некоторые популярные открытые решения: *mRo Pixhawk, ArduPilot Mega, CC3D* [5].

Полётный контроллер *mRo Pixhawk* является аппаратной частью проекта *Dronecode* и имеет 32-битный процессор STM32F427 Cortex M4 (тактовая частота 180 МГц), 256 Кбайт ОЗУ. Вторую частью этого проекта составляют POSIX совместимая операционная система NuttX с программным обеспечением *PX4 autopilot*, реализующим функции полётного контроллера. Архитектура данного ПО состоит из набора драйверов и отдельных модулей, взаимодействующих при помощи обмена сообщениями uORB и реализующие функции навигации, связи по протоколу mavlink с наземным пунктом, управления ориентацией и полетом [6]. К недостаткам можно отнести невозможность использования единой структуры программного обеспечения полетного контроллера и наземного пункта управления, сложность с модификацией прошивки под нестандартные задачи и ограниченные вычислительные ресурсы.

ArduPilot Mega 2.8 (APM) – Arduino совместимый полётный контроллер, разработанный сообществом DIY Drones. Он способен управлять автономными мультикоптерами, традиционными вертолетами, наземными колёсными роботами. В качестве бортового ПО данный контроллер использует *Ardupilot*. Недостатком данной системы является отсутствие ярко вы-

раженной модульной архитектуры и схемы взаимодействия процессов. В результате сильно увеличивается сложность модификации исходного кода и настройка под решение новой практической задачи. В качестве основного микроконтроллера используется 8-битный Atmel ATmega2560.

CC3D – полётный контроллер, имеющий 32-битный процессор STM32F405RGT6 ARM Cortex M4 (тактовая частота 168 МГц), 1 Мбайт флеш-памяти и работающий под управлением операционной системы FreeRTOS и ПО *LibrePilot*. *LibrePilot* – проект с открытым исходным кодом, основанный на *OpenPilot*, и реализующий функции полётного контроллера для квадрокоптера.

Перечисленные полётные контроллеры имеют недостатки в виде слабых технических характеристик и сложности модификации исходного кода.

Далее, рассмотрим работы, посвящённые разработке архитектуры квадрокоптера. В работе [7] представлено описание алгоритма полёта и архитектуры программной и аппаратной части полётного контроллера, с использованием фреймворка ROS. Предложенная в работе архитектура программного обеспечения состоит из двух узлов. Первый узел запускается на наземной станции и используется для отправки команд управления. Второй узел работает на самом квадрокоптере и выполняет функции полётного контроллера. Внутри узла запускается два потока, один из которых занят получением команд управления, а второй их исполнением. Использование многопоточности и отсутствие явно разграниченных модулей архитектуры будут создавать дополнительные сложности при модификации данного полётного контроллера.

Работа [8] посвящена разработке квадрокоптера, который имеет широкие возможности расширения архитектуры. Авторы представили подробное описание аппаратной части. В качестве полётного контроллера используется *Pixhawk*, который выполняет функции автопилота, и подключенная к нему плата *Odroid U3*, на которой авторы предлагают запускать более ресурсоемкие алгоритмы полёта (например, с использованием показаний видеокамеры). Команды управления с платы *Odroid U3* поступают *в Pixhawk* по адаптированному для ROS протоколу *mavlink – mavros*. Однако, для изменения алгоритма одной из базовых функций полётного контроллера (например, алгоритма стабилизации по углам крена / тангажа / рысканья) необходимо модифицировать ПО автопилота *Pixhawk*.

Авторы работы [9] предлагают описание концептуальной системы управления БПЛА с использованием ROS. Представленная в работе архитектура включает в себя не только квадрокоптер, но и сервер баз данных, backend-сервер, наземную станцию управления и другие части. Она описывает взаимосвязь программных компонентов, показывает, какое ПО должно быть установлено и запущено на физических машинах. Предложенные авторами архитектурные модули используются для решения обобщённых высокоуровневых задач, но архитектура самого полётного контроллера в работе не представлена.

В работе [10] представлено описание эксперимента по управлению квадрокоптером при транспортировке груза на подвесе. В качестве полётного контроллера квадрокоптера использовался *Pixhawk*, а фреймворк ROS применялся для разработки наземной станции, которая следит за траекторией движения БПЛА и посылает автопилоту команды управления по протоколу *mavros*.

Постановка задачи

Таким образом, создаваемый полетный контроллер должен обеспечивать решение основных задач получения и обработки сигналов с датчиков летательного аппарата, формирования команд для драйверов двигателей в соответствии с алгоритмом управления, обеспечивать запуск универсальных операционных систем и тем самым предоставлять удобные механизмы для разработки, модификации, отладки и конфигурирования программного обеспечения под меняющиеся задачи. Чтобы избежать дополнительных потерь аппаратов, возникающих

при тестировании новых алгоритмов управления и компонентов ПО, необходимо обеспечить возможность проведения программной (SIL) и программно-аппаратной (HIL) симуляции полёта, когда реальный контроллер работает в контуре управления с математической моделью движения квадрокоптера.

В качестве аппаратной части полётного контроллера решено использовать одноплатный компьютер Raspberry Pi, поскольку он обладает хорошими техническими характеристиками и невысокой стоимостью. Данное устройство выпускается в нескольких версиях, в данной работе используется версия Raspberry Pi Model 3 B, имеющая 64-битный процессор ARM Cortex-A53 (тактовая частота 1,2 ГГц) и 1 ГБ ОЗУ. В качестве операционной системы используется Ubuntu 18.04.

Чтобы контроллер мог эффективно управлять полётом БПЛА, ему необходимо определить текущую ориентацию устройства в пространстве. Для этого используется инерциальное измерительное устройство (IMU) MPU-6050. Микросхема включает в себя акселерометр и гироскоп, встроенный фильтр нижних частот, и подключается к Raspberry Pi по шине I2C.

Для разработки программного обеспечения использовался фреймворк ROS (Robot Operating System), в качестве языка программирования выбран C++. Использование ROS позволяет легко реализовать модульную архитектуру полетного контроллера, благодаря своей концепции узлов и тем.

Узел (node) – это отдельная программа, написанная на Python или C++, которая решает некоторую задачу и может взаимодействовать с другими узлами посредством ROS-тем. Разделение сложных систем на изолированные узлы дает определенные преимущества: понижается связанность кода, облегчается внесение модификаций, повышаются возможности повторного использования программного кода.

Тема (topic) – это именованный канал связи, по которому узлы обмениваются сообщениями. Любой узел может опубликовать сообщение в произвольную тему, а также получать сообщение из любой темы.

Основным преимуществом ROS является то, что данный фреймворк обеспечивает построения программных систем в виде набора независимых программ, взаимодействие между которыми осуществляется через отправку и доставку сообщений между узлами, даже в случае распределённой системы, когда узлы запущены на разных машинах, находящихся в одной сети. Это дает возможность гибко разбивать задачи на отдельные узкоспециализированные программы, использовать удобные средства разработки и языки программирования, конфигурировать требуемую функциональность системы за счет комбинации отдельных программ. В результате, все базовые функции полетного контроллера должны реализовываться в виде отдельных узлов ROS.

Архитектура программного и аппаратного обеспечения

Архитектура разработанного полётного контроллера показана на рис. 1 и состоит из четырёх основных узлов:

1) драйвер IMU (IMU_driver_node);
2) узел определения ориентации (orientation_node);
3) узел управления ориентацией (controller_node).
4) узел управления позицией (position_controller_node).

Драйвер IMU – узел, который выполняет сбор данных с датчика MPU-6050, подключенного по шине I2C к плате. Получаемые данные содержат показания акселерометра и гироскопа. Данный узел решает следующие задачи: доступ к регистрам, содержащим показания MPU-6050, преобразование полученных данных к нужным единицам измерения (м/с^2 для акселерометра, рад/с для гироскопа), формирование и отправка сообщения в тему, на которую подписан узел определения ориентации. Снятие показаний датчиков осуществляется с частотой 100 Гц.

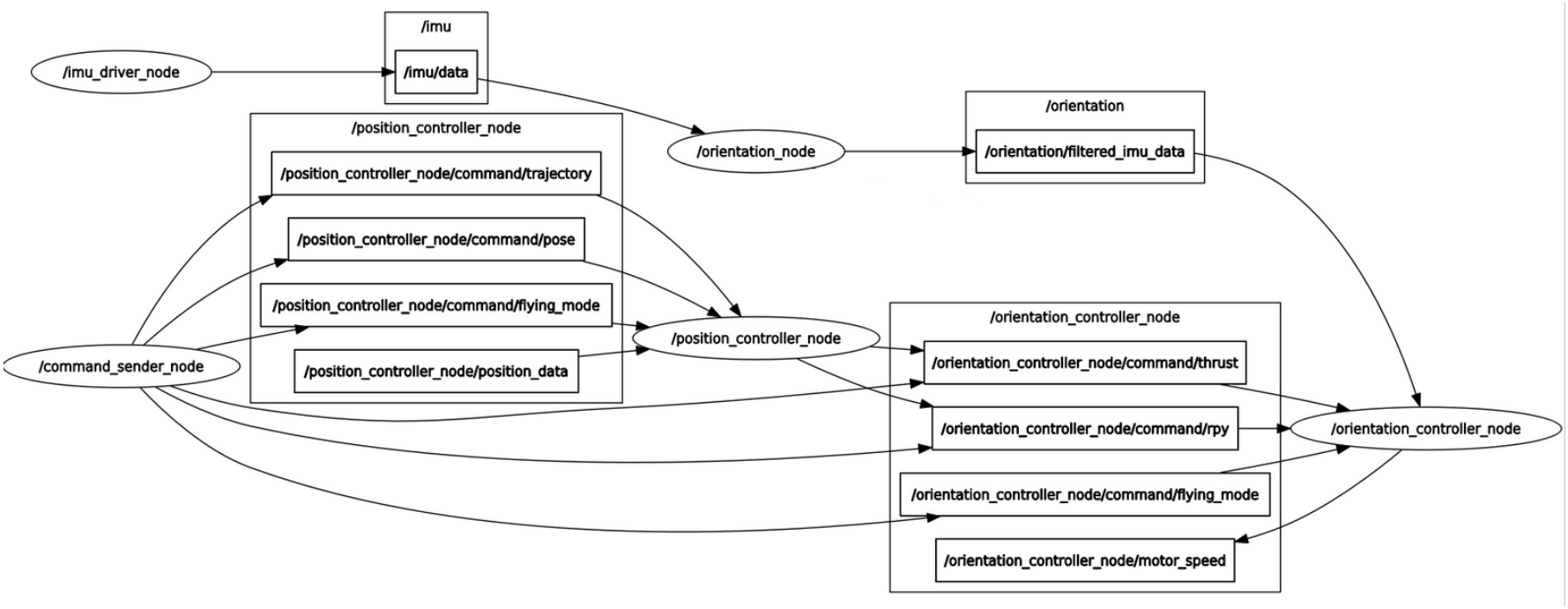

Рис. 1. Программная архитектура
Fig. 1. Software architecture

Узел определения ориентации получает показания акселерометра, гироскопа и на основе этих данных рассчитывает углы крена, тангажа и скорости рыскания (угол рыскания нельзя определить по показаниям акселерометра, а при интегрировании значений гироскопа будет накапливаться ошибка). В результате, формируется сообщение содержащее информацию об ориентации квадрокоптера и угловых скоростях крена, тангажа, рыскания) и публикуется в тему, на которую подписан узел управления.

Для определения ориентации используется фильтр Маджвика [11]. Этот фильтр работает с данными инерциальных и магнитных датчиков и существует в двух вариантах. Первый применим к системам с акселерометром и гироскопом, второй используется, когда в системе дополнительно присутствует магнитометр. В качестве реализации использовался исходный код на C, написанный самим автором фильтра [1]. Данная реализация была адаптирована для использования совместно с ROS.

Узел управления ориентацией квадрокоптера получает информацию о текущей ориентации и угловых скоростях, команды управления, и на основе этих данных рассчитывает необходимое управляющее воздействие на моторы. Данный узел, в зависимости от используемого режима полёта, управляет текущими углами крена, тангажа квадрокоптера или угловыми скоростями крена, тангажа, рыскания.

Узел управления позицией получает команды управления и информацию о текущей позиции квадрокоптера. В данном случае командой управления является либо точка, которую необходимо удерживать, либо массив точек, которые необходимо посетить. Для удержания позиции используется ПИД регулятор, результатом работы которого являются желаемые углы крена и тангажа, которые необходимо принять квадрокоптеру, чтобы начать двигаться по направлению к заданной точке. Далее эти данные передаются в качестве команды управления в узел управления ориентацией.

В результате реализованы четыре режима полета.

1. Стабилизация по скоростям крена, тангажа, рыскания – командой управления являются желаемые угловые скорости вращения вокруг осей x, y, z квадрокоптера и скорость вращения моторов.
2. Стабилизация по углам крена, тангажа, скорости рыскания – командой управления являются желаемые углы крена, тангажа, скорость рыскания квадрокоптера и скорость вращения моторов.
3. Удержания позиции – командой управления является точка с координатами (x, y, z), которую квадрокоптер будет удерживать.
4. Полет по траектории – командой управления является массив точек с координатами (x, y, z), по которым будет пролетать квадрокоптер.

Также реализована функция удержания высоты. Она может быть активирована в режимах стабилизации по скоростям или углам крена, тангажа, скорости рыскания.

Алгоритм управления

После того, как узел управления получил данные о состоянии квадрокоптера, необходимо применить поправку к скоростям вращения моторов для выполнения текущей команды управления. Для этой цели используется каскад ПИД регуляторов [12] (рис. 2).

ПИД регулятор – алгоритм управления, который на основе отклонения от величины, в которой необходимо стабилизироваться, выдает поправку на соответствующие моторы. Формула

$$u(t) = K_p e(t) + K_I \int_0^t e(x)dx + K_D \frac{d}{dt} e(t)$$

[1] https://x-io.co.uk/open-source-imu-and-ahrs-algorithms/

выражает необходимую поправку в момент времени t, при отклонении, равном $e(t)$. K_p, K_I, K_D, – пропорциональный, интегральный и дифференциальный коэффициенты соответственно.

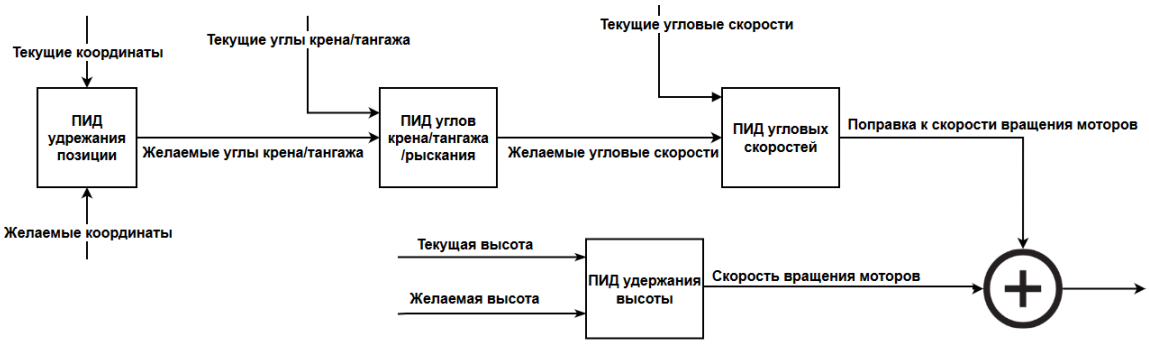

Рис. 2. Схема ПИД регуляторов
Fig. 2. Diagram of PID controllers

В режиме стабилизации по скоростям крена, тангажа, рыскания для каждой оси используется отдельный ПИД регулятор. Входными параметрами являются текущая и желаемая скорости вращения вокруг одной из осей, а результатом является поправка к скорости вращения моторов. В результате, управляющее воздействие на моторы выглядит следующим образом (нумерация моторов начинается с левого верхнего по часовой стрелке):

$$\begin{cases} motor_speed[0] = desired_thrust - pid_roll_rate + pid_pitch_rate + pid_yaw_rate \\ motor_speed[1] = desired_thrust + pid_roll_rate + pid_pitch_rate - pid_yaw_rate \\ motor_speed[2] = desired_thrust + pid_roll_rate - pid_pitch_rate + pid_yaw_rate \\ motor_speed[3] = desired_thrust - pid_roll_rate - pid_pitch_rate - pid_yaw_rate \end{cases}$$

Здесь *pid_roll_rate*, *pid_pitch_rate*, *pid_yaw_rate* – выходы соответствующих ПИД регуляторов скорости крена, тангажа, рыскания, *desired_thrust* – базовая скорость вращения моторов, полученная с командой управления (или посчитанная ПИД регулятором удержания высоты, если данный режим включен).

В режиме стабилизации по углам на вход соответствующих ПИД регуляторов подается текущий и желаемый угол крена, тангажа, а выходным параметром является угловая скорость вращения вокруг оси, которую должен развить квадрокоптер. В результате, выходы данных ПИД регуляторов подаются на входы ПИД регуляторов режима стабилизации по скоростям, в качестве одного из параметров.

В режиме удержания позиции входными параметрами ПИД регулятора являются текущая и желаемая точки с координатами (x, y, z), а выходами является желаемый угол крена, тангажа (для координат x и y). Для стабилизации по координате z используется ПИД регулятор удержания высоты.

В режиме полёта по траектории на вход подается массив точек с координатами (x, y, z), по которым должен пролететь квадрокоптер. Для перемещения из одной точки в другую строится прямая между текущей и желаемой позицией квадрокоптера. Полученная прямая разбивается на массив точек с некоторым заданным интервалом, после чего квадрокоптер последовательно пролетает по точкам из полученного массива в режиме удержания позиции. Считается, что точка была пройдена, если квадрокоптер оказался в некотором заданном радиусе от неё. При посещении последней точки траектории включается режим удержания текущей позиции. Псевдокод данного алгоритма представлен на рис. 3.

```
Начальные данные:
trajectory_points - массив точек, которые должен посетить
квадрокоптер
trajectory_line - массив точек на прямой линии (изначально пустой)
if trajectory_points not empty OR trajectory_line not empty then
    if trajectory_line not empty then
        desired_x = trajectory_line.front.x
        desired_y = trajectory_line.front.y
        desired_z = trajectory_line.front.z
        distance =
         √((current_x − desired_x)² + (current_y − desired_y)² + (current_z − desired_z)²)
        if distance < radius then
            trajectory_line.pop_front()
        end
    else
        end = trajectory_points.pop_front()
        star = current_point
        i = 0
        while i < sampling_step do
            new_point.x = start.x + i*(end.x-start.x) ÷sampling_step
            new_point.y = start.y + i*(end.y-start.y) ÷sampling_step
            new_point.z = start.z + i*(end.z-start.z) ÷sampling_step
            trajectory_line.push(new_point)
            i += 1
        end
    end
end
```

Рис. 3. Псевдокод алгоритма
Fig. 3. Algorithm pseudocode

Результаты экспериментов

Для проверки оценки ориентации, посчитанной по показаниям MPU-6050 был разработан плагин к программе визуализации rviz, поставляющейся вместе с ROS. Из ROS-темы он получает углы крена и тангажа и в реальном времени рисует прямоугольник, повернутый на данные углы (рис. 4).

Для начальной оценки работоспособности алгоритмов управления тестовые полёты проводились в физическом симуляторе Gazebo, который присутствует в составе ROS. Для симуляции различных видов роботов в Gazebo существует множество стандартных пакетов. Если же существующие пакеты по какой-либо причине не устраивают разработчика, то он может написать собственную программу симуляции физической среды, разработать и загрузить свою модель робота.

Поскольку симулятор Gazebo является частью ROS, то он использует ту же концепцию узлов и тем. Это очень удобно, поскольку алгоритм управления может быть разработан и протестирован в симуляторе и перенесён на реального робота без изменений в программе. Для тестирования разработанной архитектуры был использован пакет rotors_simulator [13]. В качестве модели квадрокоптера была выбрана встроенная модель Ardrone со следующими параметрами: масса 1,5 кг, моменты инерции I_{xx} = 0,0458929 кг·м^2, I_{yy} = 0,0458929 кг·м^2, I_{xx} = 0,0977 кг·м^2. Для отправки команд управления был разработан специальный узел command_sender_node.

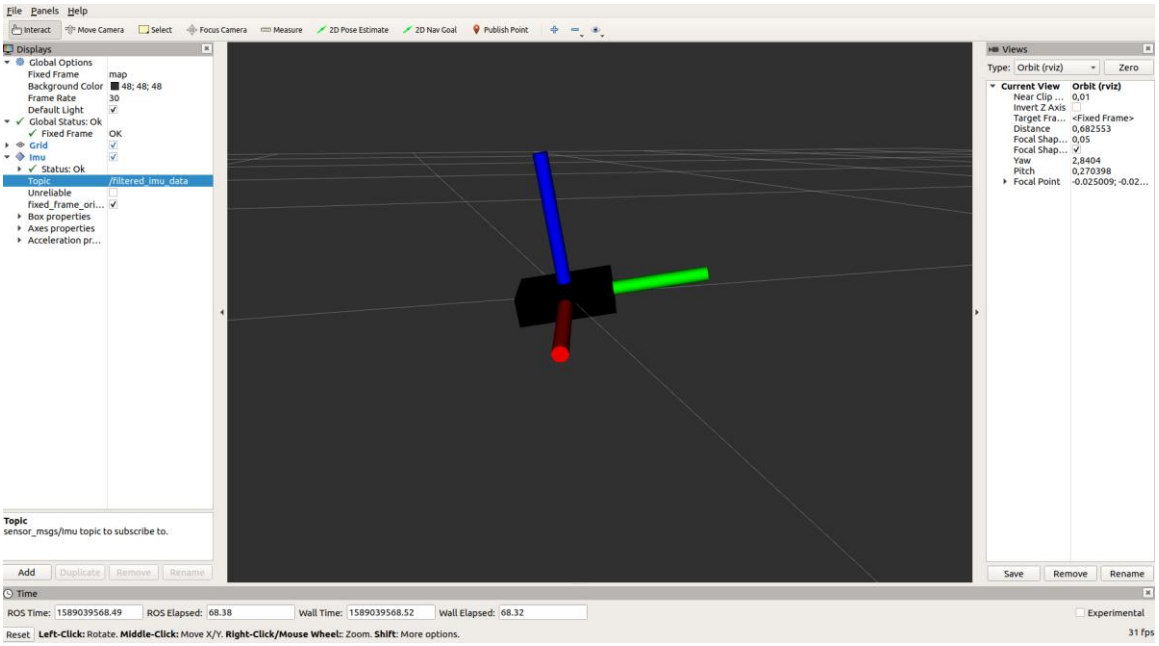

Рис. 4. Интерфейс плагина rviz
Fig. 4. Rviz plugin interface

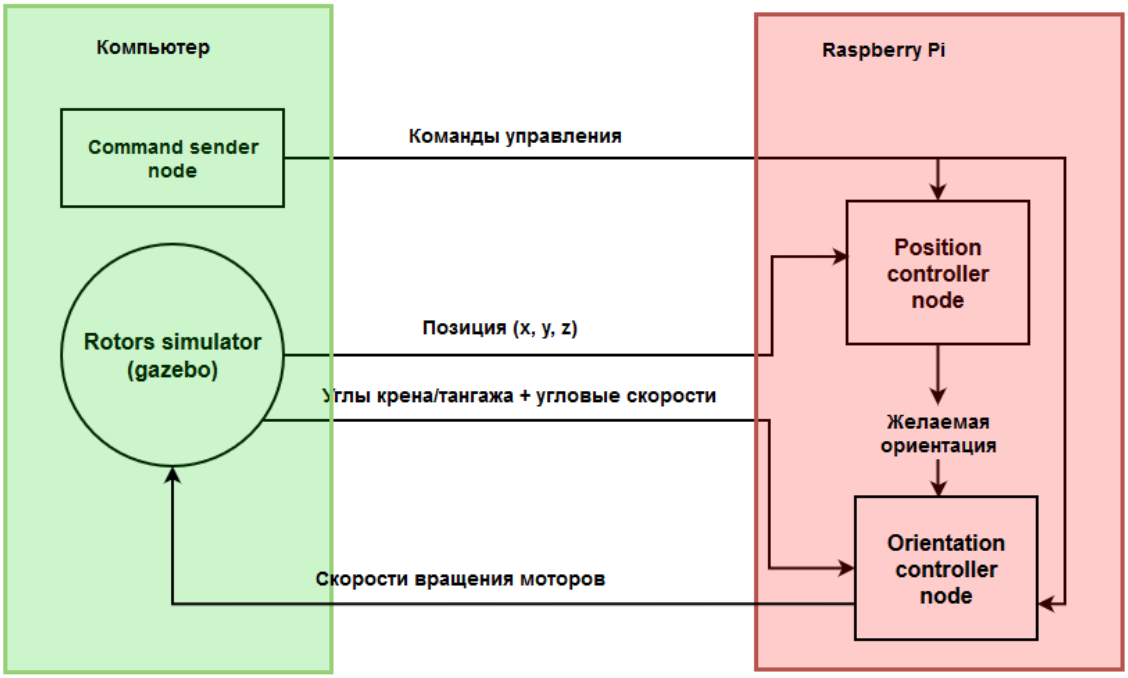

Рис. 5. Архитектура системы с использованием физического симулятора
Fig. 5. System architecture using a physical simulator

Поскольку при полётах в симуляторе нельзя воспользоваться реальными датчиками, информация об ориентации квадрокоптера поступает в полётный контроллер из симулятора. Поскольку ROS позволяет узлам обмениваться сообщениями в случае распределённой системы, то физический симулятор и узел отправки команд были запущены на персональном компьютере, а узел управления на Raspberry Pi (рис. 5). Параметры настроенных ПИД регуляторов указаны в таблице (значения по крену и тангажу совпадают, поскольку квадрокоптер считается симметричным).

На рис. 6, 7 показаны переходные процессы в режимах стабилизации по скоростям или углам крена. На рис. 8 представлены графики зависимости координат в режиме удержания позиции. Квадрокоптер стартует в точке (0, 0) и старается удержать позицию в точке (2, 2). На рис. 9 изображена траектория в режиме полёта по точкам [(2, 2) (–2, 2) (–2, –2) (2, –2) (2, 2)], начало движения в точке (0, 0). На рис. 10 показан переходный процесс в режиме удержания высоты.

Коэффициенты ПИД регуляторов
PID coefficients

Название	K_P	K_I	K_D
ПИД угловой скорости крена	1,5	0,1	0,05
ПИД угловой скорости тангажа	1,5	0,1	0,05
ПИД угловой скорости рыскания	4,5	0	0
ПИД угла крена	2,5	0	0
ПИД угла тангажа	2,5	0	0
ПИД координаты x	1,5	0	4
ПИД координаты y	1,5	0	4
ПИД координаты z	1000	3,5	300

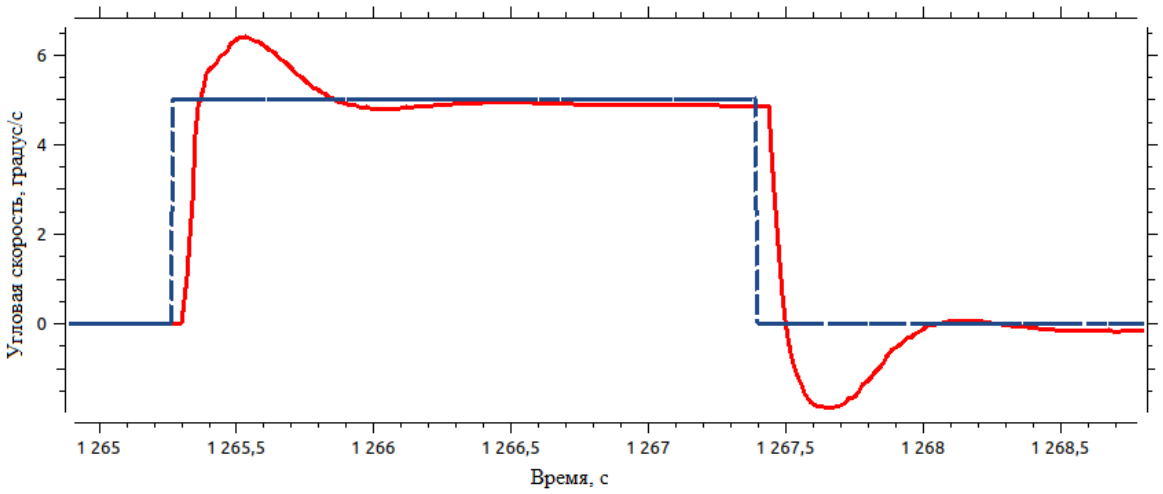

Рис. 6. Стабилизация по скорости крена
Fig. 6. Roll rate stabilization

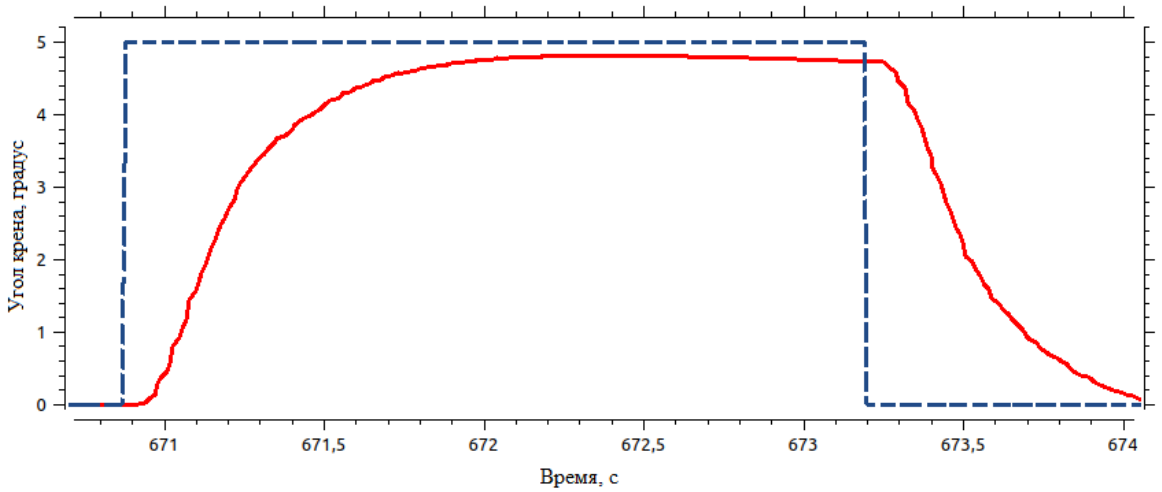

Рис. 7. Стабилизация по углу крена
Fig. 7. Roll angle stabilization

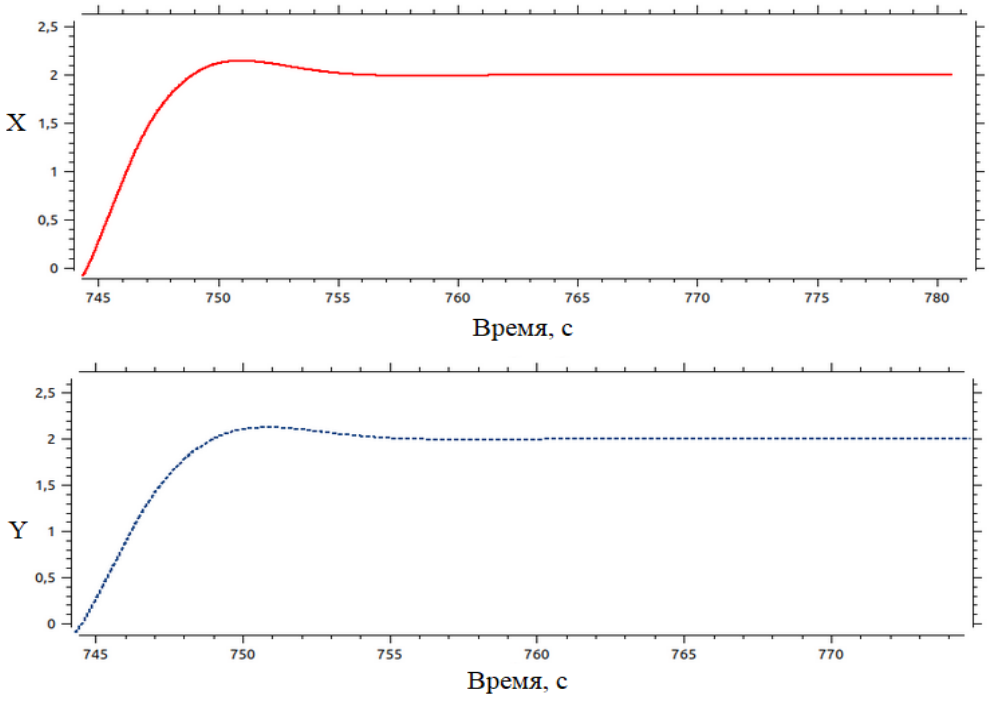

Рис. 8. Режим удержания позиции
Fig. 8. Position hold mode

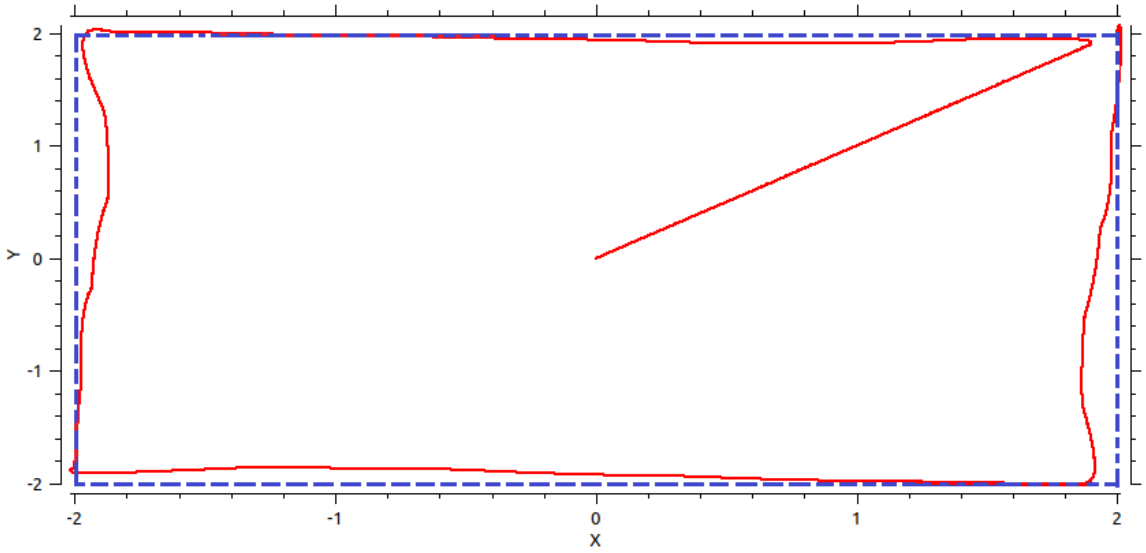

Рис. 9. Полёт по точкам
Fig. 9. Flying through points

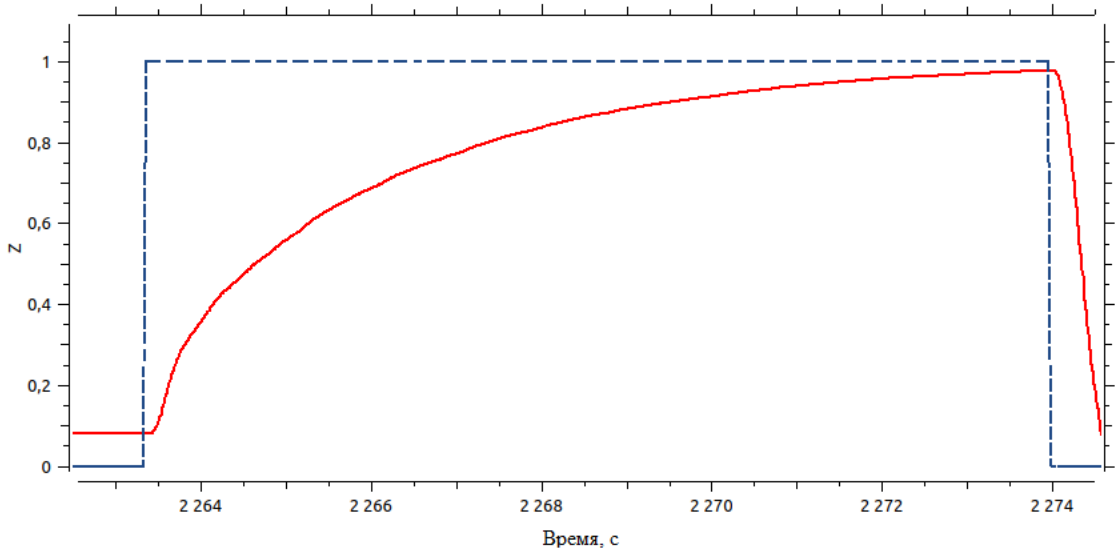

Рис. 10. Режим удержания высоты
Fig. 10. Altitude Hold Mode

Заключение

Предложена и реализована модульная архитектура полётного контроллера, отличающаяся использованием фреймворка Robot Operating System и обеспечивающая явное разделение функций на независимые программы, такие как сбор показаний датчиков, инерциальная на-

вигация, управление ориентацией, управление позицией, которые реализованы в виде отдельных узлов, на основе одноплатного компьютера Raspberry Pi.

Приведены результаты тестовых полётов в физическом симуляторе Gazebo. Благодаря использованию ROS основные функции полётного контроллера в явном виде разграничены. Единственное что связывает их между собой это типы сообщений и названия тем, откуда они поступают, поэтому архитектура имеет модульную структуру и легко поддается модификации. Например, если в дальнейшем новому узлу понадобятся данные об ориентации, то необходимо только подписаться на соответствующее сообщение, и при этом нет необходимости изменять код узла, публикующего эти данные.

По результатам численных экспериментов в симуляторе Gazebo полётный контроллер с реализованными алгоритмами управления обеспечивает стабилизацию угловой скорости с погрешностью не более 0,2 градуса/с, углов ориентации - 0,35 градуса, координат и высоты – 0,4 м и 0,05 м соответственно.

Развитием работы является использование представленного полетного контроллера для управления полетом реального аппарата и проведение серии летных экспериментов для оценки характеристик алгоритмов управления и обработки сигналов с датчиков с целью доработки системы по результатам этих экспериментов.

Список литературы

1. **Petrovic T., Haus T., Arbanas B., Orsag M., Bogdan S.** Towards heterogeneous aerial-ground cooperative robot system for complex aerial manipulation tasks. In: Proceedings of the 12th International Conference on Informatics in Control, Automation and Robotics (ICINCO, 21–23 July, 2015). Colmar, Alsace, 2015, vol. 1, p. 238–245. DOI 10.5220/0005548302 380245
2. **Hausman K., Muller J., Hariharan A., Ayanian N., Sukhatme G.** Cooperative multi-robot control for target tracking with onboard sensing. *The International Journal of Robotics Research*, 2015. vol. 34, p. 1660–1677. DOI 10.1007/978-3-319-23778-7_58
3. **Ogren P., Fiorelli E., Ehrich Leonard N.** Cooperative control of mobile sensor networks: Adaptive gradient climbing in a distributed environment. *IEEE Trans. on automatic control*, 2004, vol. 49, no. 8, p. 1292–1302. DOI 10.1109/TAC.2004.832203
4. **Nathan M., Jonathan F., Vijay K.** Cooperative manipulation and transportation with aerial robots. *Autonomous Robots*, 2011, vol. 30, no. 1, p. 73–86. DOI 10.1007/s10514-010-9205-0
5. **Ebeid E., Skriver M., Terkildsen K. H., Jensen K., Schultz U.** A survey of open-source UAV flight controllers and flight simulators. *Microprocessors and Microsystems*, 2018, vol. 61, p. 11–20. DOI 10.1016/j.micpro.2018.05.002
6. **Meier L., Honegger D., Pollefeys M.** PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms. *IEEE international conference on robotics and automation (ICRA)*, 2015, p. 6235–6240. DOI 10.1109/ICRA.2015.7140074
7. **Majumdar A., Gamez N., Benavidez P., Jamshidi M.** Development of robot operating system (ROS) compatible open source quadcopter flight controller and interface. In: 12th System of Systems Engineering Conference (SoSE). Waikoloa, HI, 2017, p. 1–6. DOI 10.1109/ SYSOSE.2017.7994980
8. **Abeywardena D., Pounds P., Hunt D., Dissanayake G.** Design and development of ReCOPTER: An open source ROS-based multi-rotor platform for research. In: Australasian Conference on Robotics and Automation, ACRA, 2015.
9. **Яковлев К. С., Петров А. В., Хитьков В. В.** Программный комплекс навигации и управления беспилотными транспортными средствами // Информационные технологии и вычислительные системы. 2013. № 3. С. 72–83.
10. **Димова А. С., Котов К. Ю., Мальцев А. С., Семенюк Е. Д., Соболев М. А.** Управление траекторным движением квадрокоптера при транспортировке груза на подвесе: экс-

перимент // Вестник НГУ. Серия: Информационные технологии. 2019. Т. 17, № 4. С. 46–56. DOI 10.25205/1818-7900-2019-17-4-5-46-56

11. **Madgwick S.** An efficient orientation filter for inertial and inertial/magnetic sensor arrays. *Report x-io and University of Bristol (UK)*, 2010, vol. 25, p. 113–118.
12. **Bao N., Ran X., Wu Z., Xue Y., Wang K.** Research on attitude controller of quadcopter based on cascade PID control algorithm. In: IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), 2017, p. 1493–1497. DOI 10.1109/ITNEC.2017.8285044
13. **Furrer F., Burri M., Achtelik M., Siegwart R.** RotorS – A modular Gazebo MAV simulator framework. Robot Operating System (ROS). Springer, Cham, 2016, p. 595–625. DOI 10.1007/978-3-319-26054-9_23

References

1. **Petrovic T., Haus T., Arbanas B., Orsag M., Bogdan S.** Towards heterogeneous aerial-ground cooperative robot system for complex aerial manipulation tasks. In: Proceedings of the 12th International Conference on Informatics in Control, Automation and Robotics (ICINCO, 21–23 July, 2015). Colmar, Alsace, 2015, vol. 1, p. 238–245. DOI 10.5220/0005548302380245
2. **Hausman K., Muller J., Hariharan A., Ayanian N., Sukhatme G.** Cooperative multi-robot control for target tracking with onboard sensing. *The International Journal of Robotics Research*, 2015. vol. 34, p. 1660–1677. DOI 10.1007/978-3-319-23778-7_58
3. **Ogren P., Fiorelli E., Ehrich Leonard N.** Cooperative control of mobile sensor networks: Adaptive gradient climbing in a distributed environment. *IEEE Trans. on automatic control*, 2004, vol. 49, no. 8, p. 1292–1302. DOI 10.1109/TAC.2004.832203
4. **Nathan M., Jonathan F., Vijay K.** Cooperative manipulation and transportation with aerial robots. *Autonomous Robots*, 2011, vol. 30, no. 1, p. 73–86. DOI 10.1007/s10514-010-9205-0
5. **Ebeid E., Skriver M., Terkildsen K. H., Jensen K., Schultz U.** A survey of open-source UAV flight controllers and flight simulators. *Microprocessors and Microsystems*, 2018, vol. 61, p. 11–20. DOI 10.1016/j.micpro.2018.05.002
6. **Meier L., Honegger D., Pollefeys M.** PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms. *IEEE international conference on robotics and automation (ICRA)*, 2015, p. 6235–6240. DOI 10.1109/ICRA.2015.7140074
7. **Majumdar A., Gamez N., Benavidez P., Jamshidi M.** Development of robot operating system (ROS) compatible open source quadcopter flight controller and interface. In: 12th System of Systems Engineering Conference (SoSE). Waikoloa, HI, 2017, p. 1–6. DOI 10.1109/SYSOSE.2017.7994980
8. **Abeywardena D., Pounds P., Hunt D., Dissanayake G.** Design and development of ReCOPTER: An open source ROS-based multi-rotor platform for research. In: Australasian Conference on Robotics and Automation, ACRA, 2015.
9. **Yakovlev K. S., Petrov A. V., Khitkov V. V.** Programmnyi kompleks navigatsii i upravleniya bespilotnymi transportnymi sredstvami [Software package for navigation and control of unmanned vehicles]. *Informatsionnye tekhnologii i vychislitel'nye sistemy*, 2013, no. 3, p. 72–83. (in Russ.)
10. **Dimova A. S., Kotov K. Yu., Maltsev A. S., Semenyuk E. D., Sobolev M. A.** Upravlenie traektornym dvizheniem kvadrokoptera pri transportirovke gruza na podvese: eksperiment [Controlling the trajectory of a quadrocopter when transporting cargo on a suspension: experiment]. *Vestnik NSU. Series: Information Technologies*, 2019. vol. 17, no. 4, p. 46–56. (in Russ.) DOI 10.25205/1818-7900-2019-17-4-5-46-56
11. **Madgwick S.** An efficient orientation filter for inertial and inertial/magnetic sensor arrays. *Report x-io and University of Bristol (UK)*, 2010, vol. 25, p. 113–118.

12. **Bao N., Ran X., Wu Z., Xue Y., Wang K.** Research on attitude controller of quadcopter based on cascade PID control algorithm. In: IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), 2017, p. 1493–1497. DOI 10.1109/ITNEC.2017.8285044
13. **Furrer F., Burri M., Achtelik M., Siegwart R.** RotorS – A modular Gazebo MAV simulator framework. Robot Operating System (ROS). Springer, Cham, 2016, p. 595–625. DOI 10.1007/978-3-319-26054-9_23

Материал поступил в редколлегию
Received
15.05.2020

Сведения об авторах

Липовый Дмитрий Анатольевич, магистрант факультета информационных технологий, Новосибирский государственный университет (Новосибирск, Россия)

DimanqqV2.0@gmail.com

Мальцев Александр Сергеевич, кандидат технических наук, научный сотрудник, Институт автоматики и электрометрии СО РАН (Новосибирск, Россия)

alexandr@idisys.iae.nsk.su

Information about the Authors

Dmitriy A. Lipoviy, Master's Student, Faculty of Information Technologies, Novosibirsk State University (Novosibirsk, Russian Federation)

DimanqqV2.0@gmail.com

Aleksandr S. Maltsev, Candidate of Science (Techniques), Researcher, Institute of Automation and Electrometry SB RAS (Novosibirsk, Russian Federation)

alexandr@idisys.iae.nsk.su