УДК 004.3

В. И. Кияев
Профессор кафедры информатики
ФГБОУ ВО «Санкт-Петербургский государственный экономический университет»

К. А. Дятлов
Выпускник бакалавриата кафедры информатики
ФГБОУ ВО «Санкт-Петербургский государственный университет»

ИСПОЛЬЗОВАНИЕ IoT-ТЕХНОЛОГИЙ ДЛЯ МОНИТОРИНГОВЫХ СИСТЕМ В ХОЗЯЙСТВЕННОЙ ДЕЯТЕЛЬНОСТИ

Аннотация. Современные IoT-технологии могут успешно использоваться при формировании управляющих и мониторинговых систем в хозяйственной деятельности. Построение подобных систем на основе исполнительных робототехнических устройств – в частности, на основе применения беспилотных летательных аппаратов (БПЛА) – связано с определёнными трудностями, в случаях их использования в закрытых пространственных объёмах, имеющих сложную конфигурацию. В статье излагается методика построения симулятора, который позволяет построить модель системы управления такого БПЛА.

Ключевые слова: IoT-технологии, управляющие и мониторинговые системы, беспилотный летательный аппарат.

Введение

В настоящее время технологии Интернета вещей (IoT) всё более широко используются в адаптивных системах управления и мониторинга. Это в большой степени касается вопросов управления сложными распределенными хозяйственными объектами и экологического мониторинга. В качестве примера можно привести текущий мониторинг целостности и работоспособности наземных и подводных нефтяных и газотранспортных систем, опор плавучих буровых установок, поиск очагов лесных пожаров, слежение за состоянием шельфов и береговых линий морей и крупных озёр. В качестве исполнительных механизмов подобных управляющих и мониторинговых систем часто используются различные исполнительные робототехнические устройства: наземные колёсные и гусеничные самоходные устройства (роверы), миниатюрные подводные лодки и беспилотные летательные аппараты (БПЛА) – это небольшие самолёты и вертолеты с несколькими несущими винтами (квадрокоптеры, дроны) [1].

Опыт использования БПЛА показал, что они могут быть использованы для широкого круга мониторинговых задач, связанных с экологией и безопасностью. А это требует наличия гибкой, желательно адаптивной системы управления действиями БПЛА. Адаптивность обуславливается тем, что БПЛА выполняют задачи часто в группах и в сложных рельефных и метеорологических условиях, то есть уровень неопределенности может быть достаточно высок, что может вызывать нежелательные инциденты. Для эффективного обнаружения и исключения инцидентов требуется удобное решение, автоматизирующее управление процессами управления БПЛА, которые происходят в режиме реального времени – с учётом того, что маршруты (траектории) БПЛА могут пересекаться как в пространстве, так и во времени. Несмотря на то, что управление перемещением таких объектов достаточно строго регламентировано, недостаточная автоматизация процессов управления, а также влияние внешней среды и человеческого фактора неоднократно, как показывает статистика, приводило к нежелательным последствиям.

Отметим ещё одно немаловажное обстоятельство. Положение БПЛА – пространственно-временные координаты {X, Y, Z, t} – в открытом воздушном пространстве с высокой степенью точности можно определять при помощи системы GPS-локации. Однако, если летательный аппарат выполняет задачу наблюдения или поиска в лесу ниже уровня верхушек деревьев, в сильно пересеченной местности или в помещениях сложной пространственной конфигурации, то локационные GPS-сигналы могут не достигать

приёмника БПЛА, а управленческие сигналы могут многократно отражаться от окружающих поверхностей, что приводит к их рассеиванию и ослаблению. Это может привести к невыполнение поставленной задачи или, в худшем случае, поломке летательного аппарата.

Гипотеза

Современные средства управления исполнительными робототехническими устройствами в рамках применения технологии Интернета вещей позволяют успешно решать большой спектр задач в области управления сложными распределенными хозяйственными объектами и экологического мониторинга.

Методы

На сегодняшний день большая часть беспилотных аппаратов управляется непосредственно человеком. Существующие системы автопилотирования в подавляющем большинстве случаев используют для навигации и геолокации спутниковую систему GPS. Зачастую этого достаточно, чтобы беспилотник долетел в заданное место назначения и выполнил поставленную задачу, однако такое решение не подходит, как было сказано выше, для задач управления полетом БПЛА в помещениях сложной пространственной конфигурации (шахты, тоннели, пещеры и т. д.), в лесу и на пересечённой местности, где получение сигнала с GPS-спутников затруднено препятствиями. В данном случае целесообразно будет использовать ориентирование с помощью камеры, установленной на БПЛА. Однако, возникает проблема: при разработке алгоритма ориентирования в пространстве по камере необходимо проведение большого числа тестов в реальных помещениях с использованием реального беспилотника. Каждое неправильное срабатывание алгоритма может привести к повреждениям или полной поломке устройства. Эту проблему можно решить созданием виртуальной модели физического летательного аппарата и исследованием его поведения в различных условиях, близких к реальным.

Из всего многообразия беспилотников в нашем случае был исследован наиболее распространенный вид БПЛА – квадрокоптер, так как он сочетает в себе маневренность, компактность, простоту управления и доступность рядовому потребителю. Основой исследования являются изучение физики полета квадрокоптера, принципов работы модулей аппарата, алгоритма системы управления и стабилизации квадрокоптера для их последующей реализации в симуляторе.

Квадрокоптер – беспилотный летательный аппарат, построенный по вертолетной схеме, с четырьмя несущими винтами, у которого два противоположных винта вращаются в одном направлении, и два других – в обратном, при этом маневры осуществляются путем изменения скорости вращения винтов. Управление квадрокоптером осуществляется либо дистанционно человеком при помощи пульта управления, либо автономно на основе данных навигационной системы. Система автопилота может быть расположена как на борту квадрокоптера, так и на земной станции, использующей переданные изображения с камеры беспилотника для вычисления текущих координат и построения оптимального маршрута полета в реальном времени. Предлагаемый симулятор должен обеспечивать возможность тестирования всех трех видов систем управления. Более того, необходимо предусмотреть тот факт, что физическая модель квадрокоптера может быть установлена на бортовой компьютер беспилотника для последующего расчета и предсказания траектории облета препятствий прямо во время выполнения задачи. Вычислительная мощность микрокомпьютеров, устанавливаемых на борт БПЛА, сильно уступает производительности стационарных ПК, объем RAM не превышает 2GB, а объем постоянной памяти ограничен емкостью флеш-карты. Это накладывает ощутимые ограничения на потребляемые симулятором ресурсы процессора и оперативной памяти, а также на занимаемый объём скомпилированной и собранной реализации.

В основе ряда алгоритмов уклонения от столкновения с объектами лежит подход автоматического дифференцирования. Следовательно, в реализации симулятора дифференциальные уравнения, описывающие физику полета квадрокоптера, должны быть легко доступны для понимания, изменения и оптимизации. В противном случае, пользователь симулятора будет вынужден потратить большой объём времени на изучение и понимание принципов реализации физики вместо того, чтобы исследовать и тестировать алгоритмы облета препятствий.

В результате проведенного исследования были обобщены требования, выдвигающиеся к прототипу симуляции:

- невысокий порог вхождения для понимания принципов реализации физики полёта квадрокоптера;
- возможность моделирования физики полёта квадрокоптера;
- возможность изменять любые уравнения и параметры физики в реализации полёта;
- невысокая требовательность к ресурсам вычислительной машины;
- возможность реализации визуализации физической модели и окружения.

Прототип симулятора, решающий поставленные задачи и удовлетворяющий указанным требованиям, возможно реализовать как с нуля, так и на основе сторонней программной библиотеки, подходящей прототипу симулятора по своим инструментам и возможностям. В работе были рассмотрены существующие решения, с помощью которых реализованы похожие задачи:

- Microsoft AirSim[1] – кроссплатформенный симулятор для БПЛА с открытым исходным кодом, реализованный на базе игрового движка Unreal Engine[2]. В нем достаточно точно реализована реалистичная физика полета БПЛА, он предоставляет большие возможности по настройке модулей беспилотников, а также располагает качественной визуальной составляющей. Недостатком данного решения является внушительная требовательность к ресурсам ПК, крайне высокий порог вхождения и низкая возможность изменения уравнений физики модели квадрокоптера.
- Unity3D[3] – кроссплатформенный игровой движок, имеющий в своем арсенале множество заготовок для быстрого создания игры с реалистичной физикой. Обладает красивой графикой и менее высоким порогом вхождения, чем Unreal Engine. К недостаткам относятся высокая требовательность к ресурсам процессора и памяти, невозможность изменения физических уравнений библиотеки и сложность в интеграции с библиотекой компьютерного зрения OpenCV[4].
- Webots[5] – мультиплатформенное приложение с открытым исходным кодом для симуляции различных робототехнических систем. Предоставляет удобную среду разработки и качественную визуализацию для моделирования, программирования и симуляции роботов. Главным недостатком этого решения является его трудность в установке на бортовой компьютер квадрокоптера, и требовательность к ресурсам процессора.
- ReactPhysics3D[6] – библиотека физических примитивов на языке C++ для использования в трехмерных симуляциях и играх. Предоставляет реализованную физику твердого тела, расчет коллизий, различные методы связи твердых тел: шарниры, рычаг, пружина и др. Вследствие наличия только базовых, но достаточно мощных инструментов, в рассматриваемой библиотеке возможно изменение любых уравнений физики, а порог вхождения стороннего пользователя минимален.

[1] https://github.com/microsoft/AirSim/
[2] https://unrealengine.com/
[3] https://unity.com/
[4] https://opencv.org/
[5] https://cyberbotics.com/
[6] https://www.reactphysics3d.com/

Для визуализации в библиотеке используется графический фреймворк OpenGL[7]. Он довольно прост в освоении, не требователен к вычислительным ресурсам при невысоком числе рисуемых объектов, и позволяет получить видеопоток хорошего качества и детализации для работы алгоритмов компьютерного зрения.

В таблице 1 показаны сравнительные характеристики соответствия рассматриваемых решений поставленным требованиям. Оценки выставлены от 0 до 10, где 0 означает несоответствие требованию.

Таблица 1. Сравнительные характеристики решений на основе существующих библиотек.

Требование	Microsoft AirSim	Unity3D	Webots	ReactPhysics3D
Реализация модели квадрокоптера	10	10	10	10
Изменение физических уравнений	3	1	5	9
Ресурсы ВМ	1	2	4	8
Визуализация	10	10	10	10
Интеграция с OpenCV	1	0	4	7

По результатам сравнительного анализа было принято решение в качестве базового инструмента использовать библиотеку ReactPhysics3D.

Результаты и обсуждение

В прототипе симулятора в каждый момент виртуального времени t имеется необходимость нахождения углов *pitch, roll, yaw* поворота модели квадрокоптера по главным осям относительно инерциальной системы отсчета (рис. 1).

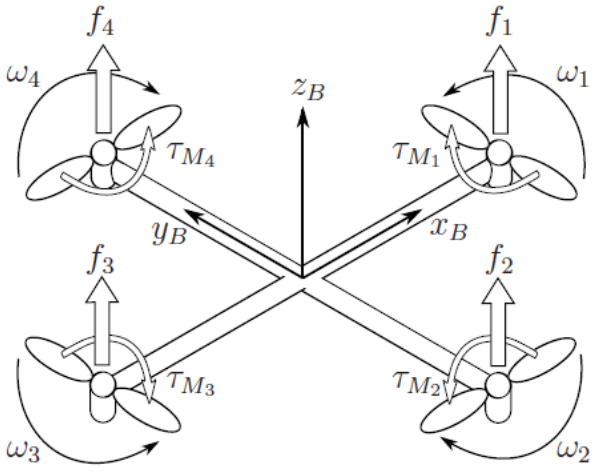

Рисунок 1. Главные оси ориентации квадрокоптера в инерциальной системе координат

Представление ориентации модели при помощи стандартной матрицы поворота [2, 6, 8] несет в себе ряд существенных недостатков:

- Для вычисления отклонения квадрокоптера по трем осям в некоторый момент времени, необходимо провести множество сложных операций, связанных с расчетом углов Эйлера по матрице поворота. Вдобавок, если объект вращается с некоторой угловой скоростью $\vec{w}$, то работа с матрицами поворота становится весьма трудоемкой задачей [7].

[7] https://www.opengl.org/

- Существует вероятность возникновения эффекта складывания рамок [5], при котором теряется возможность вращения по одной из осей в 3-х мерном пространстве. Эффект возникает из-за использования тригонометрических функций в расчете матрицы поворота. Если в произвольный момент полета некоторый угол Эйлера φ становится равным $\pm\frac{\pi}{2}$, то матрица поворота, вычисленная по этим трем углам, во всех последующих итерациях пересчета будет одинаково реагировать как на изменение угла ψ, так и на изменение угла θ. Поворот по данной матрице будет производиться вокруг одной и той же оси, несмотря на изменение значений разных углов. Выходом из представленной ситуации может послужить только изменение значения угла φ.

Для обхода перечисленных недостатков в нашей работе был использован другой способ описания ориентации в пространстве с использованием кватернионов [9]. Кватернионы – это система гиперкомплексных чисел, образующая векторное пространство размерностью четыре над полем вещественных чисел $\mathbb{R}$. В трёхмерном пространстве они представляют собой пару вектора и скаляра. Рассмотрим представление поворота на угол α в трехмерном пространстве с использованием кватернионов.

Пусть u – это единичный вектор (ось вращения), а $q = \cos\frac{\alpha}{2} + u\sin\frac{\alpha}{2}$ кватернион. Тогда мы получим:

$$v' = qvq^{-1} = \left(\cos\frac{\alpha}{2} + u\sin\frac{\alpha}{2}\right) v \left(\cos\frac{\alpha}{2} - u\sin\frac{\alpha}{2}\right) \quad (1)$$

Распишем метод расчета углов отклонения модели квадрокоптера от осей инерциальной системы отсчета в случае использования кватерниона ориентации[7].

При помощи уравнения (1) из кватерниона $q = [q_0 \quad q_1 \quad q_2 \quad q_3]^T$ текущей ориентации находим искомые углы Эйлера.

$$\begin{bmatrix} pitch \\ roll \\ yaw \end{bmatrix} = \begin{bmatrix} \text{atan2}(2(q_0q_1 + q_2q_3), 1 - 2(q_1^2 + q_2^2)) \\ \arcsin\left(2(q_0q_2 - q_3q_1)\right) \\ \text{atan2}(2(q_0q_3 + q_1q_2), 1 - 2(q_2^2 + q_3^2)) \end{bmatrix} \quad (2)$$

Следовательно, вместо матрицы поворота, вычисляемой по трем углам Эйлера, достаточно рассматривать кватернион ориентации, отвечающий за результирующий поворот модели квадрокоптера вокруг искомой оси на требуемый угол. Применение кватернионов упрощает вычисление ориентации объекта и решает проблему скрещивания рамок. В библиотеке ReactPhysics3D, выбранной как основы прототипа симулятора квадрокоптера, реализованы оба способа представления ориентации объекта: как с помощью матриц поворота, так и при помощи кватернионов.

Для решения проблемы стабилизации положения квадрокоптера в пространстве в системе управления используется ряд пропорционально интегрально-дифференцирующих (ПИД) регуляторов. Необходимость их применения мотивировано следующими причинами:

- двигатели реального квадрокоптера не являются полностью идентичными по выдаваемой мощности;
- наличие у любого физического тела инерции;
- неоднородность окружающей среды: порывы ветра, выпадение осадков, наличие различных непредусмотренных препятствий.

Для построения системы управления квадрокоптером используются три ПИД-регулятора с целью обеспечить стабильный полет аппарата в трех плоскостях. Каждый ПИД-регулятор состоит из трёх составляющих: пропорциональной, интегральной и дифференцирующей – при этом:

- пропорциональная составляющая производит выходной сигнал, противодействующий отклонению регулируемой величины от желаемого

значения, полученного в текущий момент времени. Его величина пропорциональна значению отклонения.

- интегрирующая составляющая пропорциональна интегралу по времени от отклонения регулируемой величины. Используется с целью устранения статической ошибки и позволяет регулятору с течением времени учесть эту ошибку.
- дифференцирующая составляющая пропорциональна темпу изменения отклонения регулируемой величины и предназначена для противодействия отклонениям от желаемого значения, прогнозируемым в будущем.

На рисунке 2 показана схема устройства ПИД-регулятора в математическом смысле.

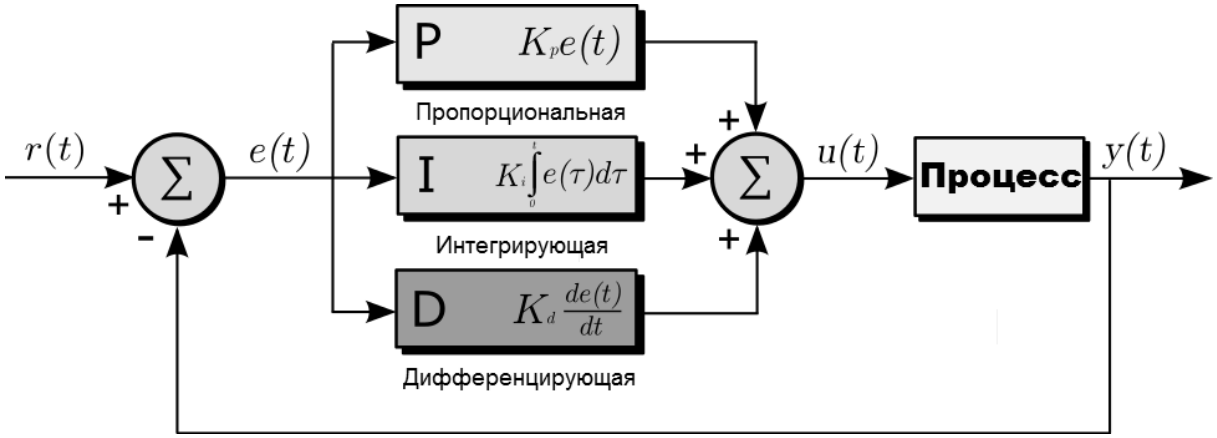

Рисунок 2. Общая схема ПИД-регулятора

В общем случае выходной сигнал ПИД-регулятора вычисляется с помощью выражения [4]:

$$u(t) = P + I + D = K_p e(t) + K_i \int_0^t e^{\tau} d\tau + K_d \frac{de}{dt},$$

где K_p, K_i, K_d – коэффициенты усиления пропорциональной, интегрирующей и дифференцирующей составляющих регулятора соответственно.

В дискретной реализации метода расчета выходного сигнала уравнение принимает следующую форму:

$$U(n) = K_p E(n) + K_p K_{ip} T \sum_{k=0}^{n} E(k) + \frac{K_p K_{dp}}{T} (E(n) - E(n-1)),$$

где T – время дискретизации.

ПИД-регулятор встроен в систему управления квадрокоптером согласно прилагаемой схеме на рис. 3.

В основе работы алгоритма системы стабилизации квадрокоптера лежат управляющие сигналы, диктующие желаемые значения ориентации и положения модели квадрокоптера в пространстве. При полете с использованием исключительно пульта управления, роль навигационной системы играет человек, он отвечает за формирование желаемых управляющих сигналов, а система стабилизации переводит их в требуемые значения напряжений на двигателях. В режиме автопилота команды с пульта дистанционного управления используются для корректировки аппарата относительно стабильного положения в пространстве, рассчитываемого на основе данных навигационной системы.

Система стабилизации каждый физический такт симуляции опрашивает гироскоп с целью получить актуальные значения углов Эйлера отклонения от инерциальной системы отсчета. Размерность углов – радианы.

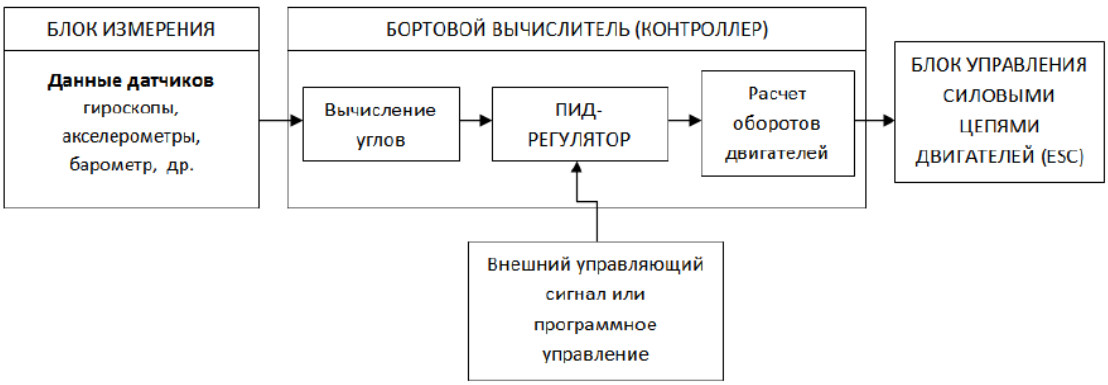

Рисунок 3. Схема системы управления квадрокоптером

Стабилизатор принимает на вход четыре внешних сигнала управления: *throttle, pitch, roll, yaw*. Значения этих сигналов лежат в интервале от 900 до 2100. Путем применения масштабирующих функций величины управляющих сигналов приводятся к промежутку $[-\pi;\pi]$, понятному для работы ПИД-регуляторов.

Полученные значения с гироскопа вместе с входными отмасштабированными сигналами передаем соответствующим ПИД-регуляторам, в которых вычисляются значения желаемых тяг по трем осям. Затем, используя уравнения (2), производится расчет значений тяги на четырех двигателях квадрокоптера.

$$\begin{bmatrix} \text{PWM\_FR} \\ \text{PWM\_FL} \\ \text{PWM\_BR} \\ \text{PWM\_BL} \end{bmatrix} = \begin{bmatrix} \text{throttle} + \text{thrustPitch} - \text{thrustRoll} + \text{thrustYaw} \\ \text{throttle} + \text{thrustPitch} + \text{thrustRoll} - \text{thrustYaw} \\ \text{throttle} - \text{thrustPitch} - \text{thrustRoll} - \text{thrustYaw} \\ \text{throttle} - \text{thrustPitch} + \text{thrustRoll} + \text{thrustYaw} \end{bmatrix} \quad (2)$$

При помощи графического фреймворка OpenGL были реализованы сцена и пользовательский интерфейс, предоставляющие пользователю возможности:

- запускать симулятор в реальном времени,
- выполнять пересчет динамики объектов покадрово,
- ставить симулятор на паузу и возобновлять его,
- выполнить перезапуск всей симуляции с исходными данными.

Также возможно включение/отключение влияния гравитации и изменение частоты пересчета физических уравнений модели.

На рисунке 4 представлена диаграмма классов реализованной модели квадрокоптера. В модели были воплощены все основные модули квадрокоптера: инерциальные измерительные датчики, камера квадрокоптера, двигатели и центральный модуль с системой управления и стабилизации квадрокоптера.

В реализованном симуляторе есть возможность настройки следующих параметров физической модели для тестирования систем автономной навигации:

- масса двигателей;
- масса центрального модуля;
- размер несущей рамы квадрокоптера;
- размер несущих винтов квадрокоптера;
- значения коэффициентов ПИД-регуляторов;
- величины коэффициентов тяги двигателей;

- величины коэффициентов вращательных моментов винтов;
- включение/отключение любого из датчиков;
- настройка масштабирующих функций управляющих сигналов.

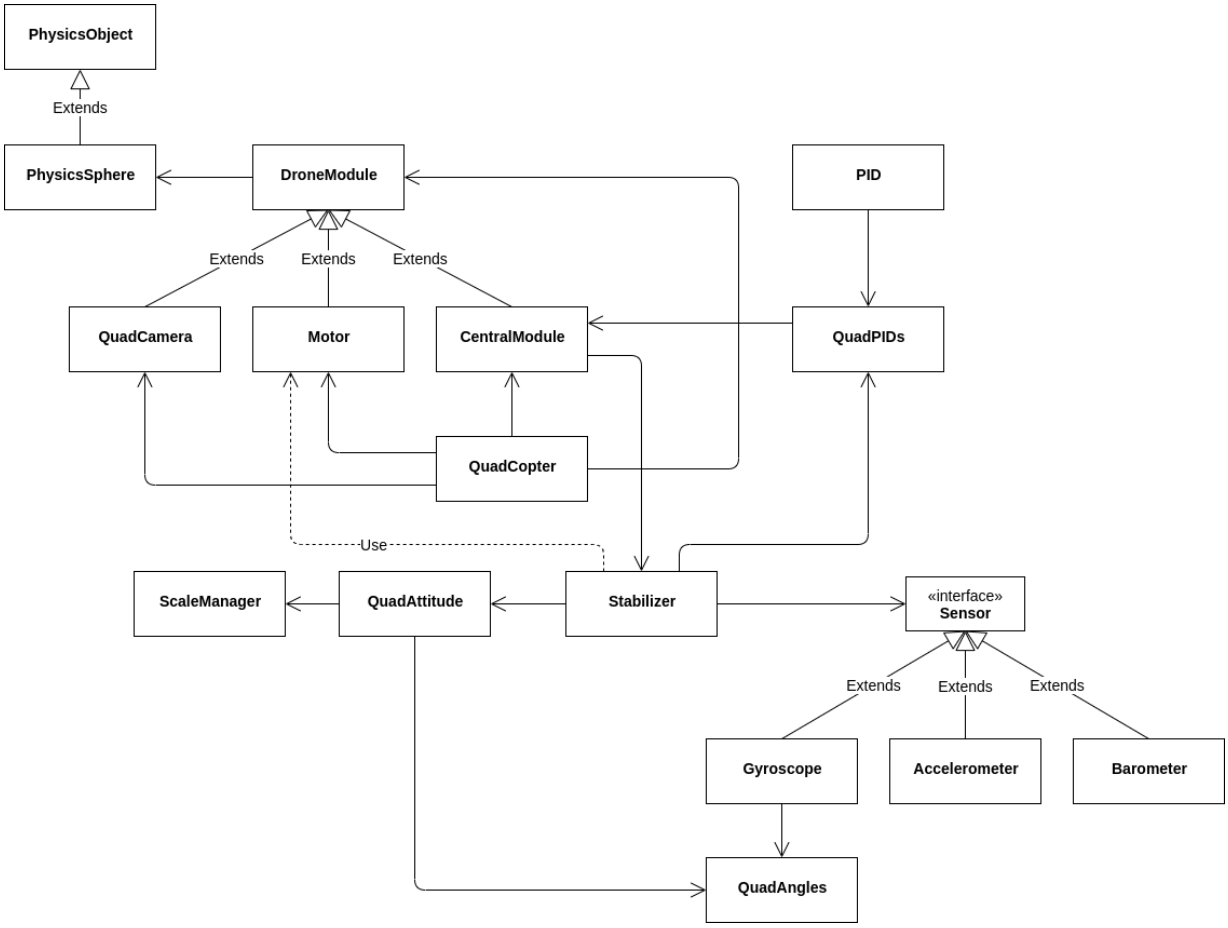

Рисунок 4. Диаграмма классов реализованной модели квадрокоптера

На рисунке 5 показана реализованная модель квадрокоптера в виртуальной сцене, и два возможных способа наблюдения за симуляцией: с камеры, свободно перемещающейся по сцене, и с камеры, находящейся на борту модели квадрокоптера.

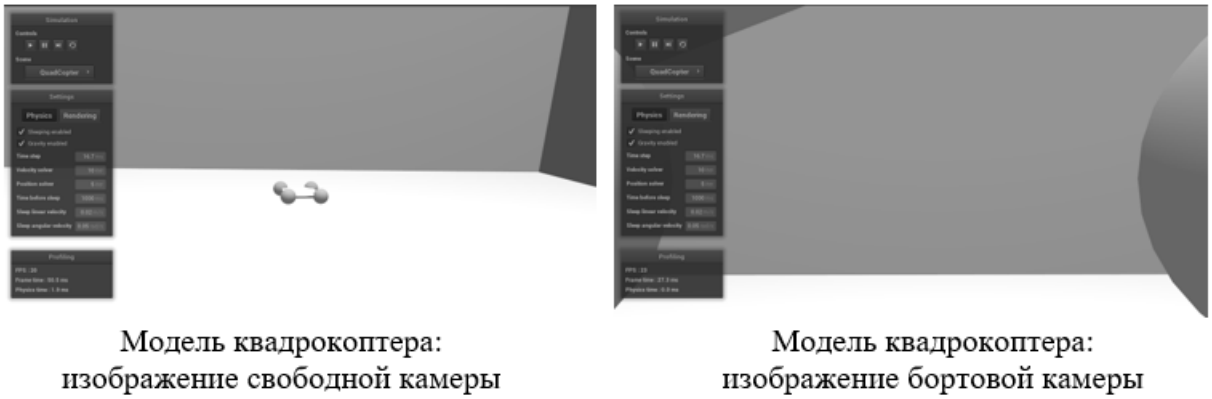

Рисунок 5. Экранные формы для представления положения квадрокоптера в виртуальной сцене пространственной конфигурации

Таким образом, на основе выполненного исследования были выбраны наиболее подходящие инструменты для реализации симуляции и разработан прототип симулятора

квадрокоптера, позволяющий использовать тестирование автономной навигационной системы на основе визуальной одометрии как при полёте дрона в реальном мире, так и симуляцию в сгенерированном под нужды конкретной задачи окружении. Построенный симулятор позволяет при необходимости добавлять в модель полёта квадрокоптера различные внешние возмущения, что делает возможным моделировать и исследовать его поведение в сложных пространственных конфигурациях. Такая расширенная функциональность с набором различных физических датчиков, установленных на квадрокоптере (камеры слежения, датчики ориентировки и визуального сближения) позволит использовать симулятор для эффективного тестирования систем ориентирования в условиях, где системы GPS практически не применимы.

Список литературы

1. Амелин К. С., Амелина Н.О., Граничин О. Н., В. И. Кияев. Разработка приложений для мобильных интеллектуальных систем на платформе Intel Atom (монография). − Санкт-Петербург: BBM, 2012. − 211 С.
2. Гурьянов А.Е. Моделирование управления квадрокоптером // Инженерный вестник. – 2014. № 8.
3. Andrew Witkin, David Baraff Michael Kass. Physically Based Modelling, SIGGRAPH 2001 Course Notes 25. − 2001.
4. Жмудь В.А., Заворин А.Н., О.Д. Ядрышников. Неаналитические методы расчета ПИД-регуляторов: учебное пособие // Изд-во НГТУ. – 2013.
5. Mitchell EEL, Rogers AE. Quaternion parameters in the simulation of a spinning rigid body // Simulation. – 1965. – Vol.4, no. 6. – pp. 390-396.
6. Teppo Luukkonen. Modelling and control of quadcopter. // Independent research project in applied mathematics, Espoo. – 2011. – Vol. 22.
7. James Diebel. Representing attitude: Euler angles, unit quaternions, and rotation vectors // Matrix. – 2006. – Vol. 58, no. 15-16. – pp. 1-35.
8. Beji L, Abichou A, Slim R. Stabilization with motion planning of a four-rotor mini rotorcraft for terrain missions // Fourth International Conference on Intelligent Systems Design and Applications (ISDA). – 2004. − pp. 335-340.
9. Jack CK Chou. Quaternion kinematic and dynamic differential equations // IEEE Transactions on robotics and automation – 1992. – Vol.8, no. 1. – pp. 53-64.