МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук.

РАЗРАБОТКА ПРОТОТИПА АВТОПИЛОТА ДЛЯ ЛЕТАТЕЛЬНОГО АППАРАТА С ИСПОЛЬЗОВАНИЕМ СИМУЛЯТОРА

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

Студентки 4 курса 451 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Дорохиной Анастасии Сергеевны

Научный руководитель
доцент, к. ф.-м. н. _____ И. А. Батраева

Заведующий кафедрой
доцент, к. ф.-м. н. _____ С. В. Миронов

Саратов 2026

Выпускная квалификационная работа посвящена разработке прототипа системы автономного управления беспилотным летательным аппаратом в симуляционной среде. Актуальность работы обусловлена широким распространением беспилотных технологий в различных отраслях промышленности, логистики, мониторинга территорий и автоматизированного контроля объектов. Современные беспилотные системы требуют высокого уровня автономности, что предполагает способность аппарата самостоятельно выполнять взлёт, навигацию по заданному маршруту, обнаруживать препятствия и безопасно завершать миссию без постоянного участия оператора.

Практическая реализация подобных систем на реальных летательных аппаратах связана со значительными материальными затратами и рисками повреждения оборудования. По этой причине важным этапом разработки является создание программных решений в виртуальной среде моделирования, позволяющей тестировать алгоритмы управления в условиях, максимально приближённых к реальным.

Целью работы являлась разработка программного комплекса автономного управления беспилотным летательным аппаратом, способного выполнять полный цикл миссии: от взлёта до посадки, используя алгоритмы компьютерного зрения и интеллектуального принятия решений.

В рамках достижения поставленной цели были реализованы следующие задачи:

- интеграция симулятора с программным автопилотом;
- разработка системы управления полётом;
- создание механизма построения маршрута;
- реализация алгоритмов обнаружения препятствий;
- разработка алгоритма обхода препятствий;
- внедрение визуальной навигации на основе ArUco-маркеров;
- создание пользовательского интерфейса управления;
- проведение испытаний разработанного программного комплекса.

Разработанная система представляет собой набор взаимодействующих программных компонентов, каждый из которых отвечает за выполнение определённой части общей задачи. Архитектура построена по модульному принципу, что обеспечивает возможность дальнейшего расширения функциональности и адаптации под реальные условия эксплуатации (см. рис. 1).

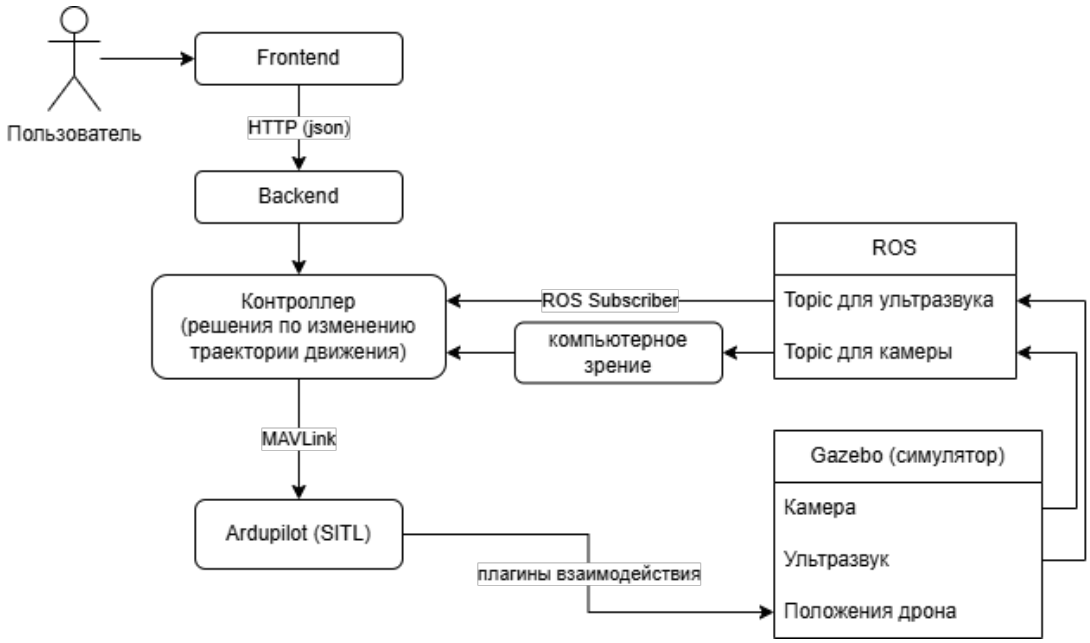

Рисунок 1 – Архитектура программного комплекса

В качестве основы программного комплекса использовались симулятор Gazebo, система автопилота ArduPilot в режиме Software-in-the-Loop (SITL), библиотека MAVLink для обмена данными, программная платформа ROS для передачи сообщений между компонентами системы, библиотека OpenCV для обработки изображений и модель компьютерного зрения YOLO для обнаружения препятствий. Пользовательское взаимодействие реализовано посредством веб-интерфейса.

Разработка велась в операционной системе Linux с использованием виртуальной машины. В процессе подготовки среды была выполнена настройка программного окружения, включающего Gazebo, ArduPilot SITL, ROS, Python, FastAPI и React. В ходе тестирования было установлено, что для стабильной работы симуляции необходимы не менее шести виртуальных процессорных ядер и не менее 10 ГБ оперативной памяти. Данные параметры обеспечивали корректную работу физического движка Gazebo и программного автопилота без потери устойчивости модели квадрокоптера.

Первым этапом практической реализации стало создание виртуальной среды моделирования. В разработанной конфигурации Gazebo выполняет несколько функций одновременно: отвечает за расчёт динамики движения квадрокоптера, обеспечивает работу виртуальных датчиков, формирует видеопоток с установленных на аппарате камер. В Gazebo был сформирован полигон, содержащий объекты, которые могли использоваться как препятствия на маршруте

полёта. Для снижения вычислительной нагрузки использовался ограниченный набор моделей: здания, деревья и геометрические примитивы. Каждый объект описывался набором физических характеристик, включая положение в пространстве, параметры коллизии, визуальное представление и массу. Такой подход позволил обеспечить реалистичное взаимодействие беспилотного аппарата с окружающей средой и проводить испытания алгоритмов автономного управления в условиях, приближенных к реальным.

Особое внимание уделялось обеспечению реалистичности условий испытаний. При размещении объектов учитывались размеры беспилотного аппарата, возможные траектории движения и условия работы алгоритмов компьютерного зрения. Это позволило получать результаты, максимально приближённые к поведению системы в реальной эксплуатации.

В процессе настройки среды были выявлены определённые аппаратные ограничения. Симуляция одновременно использует физический движок, систему компьютерного зрения и программный автопилот. При недостатке вычислительных ресурсов возникали ошибки моделирования и нестабильность поведения дрона. Для решения данной проблемы была проведена оптимизация параметров виртуальной машины и выделены дополнительные ресурсы центрального процессора и оперативной памяти.

После настройки симулятора была реализована интеграция симулятора с ArduPilot. В работе использовался режим SITL, позволяющий запускать полноценный экземпляр автопилота без использования реального полётного контроллера.В этом режиме все функции реального полётного контроллера выполняются программным обеспечением, работающим непосредственно на компьютере. Связь между Gazebo и ArduPilot была организована посредством протокола MAVLink. Данный протокол является отраслевым стандартом обмена данными между наземными станциями управления, автопилотами и дополнительными вычислительными модулями. Через MAVLink передаются команды управления, телеметрическая информация, диагностические сообщения и параметры состояния летательного аппарата.

Для связи с автопилотом был разработан программный модуль на языке Python с использованием библиотеки pymavlink. После подключения к UDP-порту автопилота программа ожидала получение heartbeat-сообщения, подтверждающего готовность системы к работе. После установления соединения систе-

ма получает возможность изменять режимы полёта, выполнять взлёт и посадку, задавать скорости движения и получать данные о положении аппарата..

Особое внимание было уделено организации двустороннего обмена данными. Система поддерживает передачу команд изменения режима полёта, взлёта, посадки, изменения ориентации и задания линейных скоростей. Одновременно осуществляется непрерывное получение телеметрии, включающей координаты, ориентацию аппарата, GPS-данные, диагностические сообщения и информацию о состоянии системы. Полученные данные используются всеми компонентами программного комплекса для принятия решений в реальном времени.

Центральным элементом проекта является модуль управления полётом. Для организации логики работы был реализован конечный автомат состояний. Такой подход позволил разделить работу системы на отдельные этапы и обеспечить строгое управление переходами между ними.

В разработанном автомате состояний используются следующие основные режимы работы:

– ожидание запуска миссии;

– поиск стартовой визуальной метки;

– взлёт;

– выполнение маршрута;

– обход препятствия;

– посадка;

– завершение миссии.

В начальном состоянии система находится в режиме ожидания запуска. После получения команды от пользователя выполняется переход к этапу поиска стартовой визуальной метки. Если требуемая метка обнаружена, система автоматически инициирует процедуру взлёта. После завершения набора высоты происходит загрузка маршрута и переход к выполнению миссии. В процессе движения система непрерывно отслеживает наличие препятствий. При возникновении потенциально опасной ситуации происходит переход в режим обхода препятствия. После завершения манёвра управление возвращается к выполнению основной миссии. Завершающим этапом является обнаружение посадочной метки и выполнение автоматической посадки.

На этапе взлёта выполняется перевод ArduPilot в режим GUIDED. По-

сле успешного изменения режима производится активация двигателей и набор заданной высоты. После стабилизации аппарата система автоматически переходит к выполнению основной миссии.

Маршрут полёта задаётся пользователем через веб-интерфейс и передаётся в систему в формате JSON. Каждая точка маршрута содержит координаты в локальной системе координат симулятора. После получения миссии точки загружаются в навигационный контроллер и используются для формирования траектории движения.

Для управления навигацией был реализован отдельный класс контроля TrajectoryController. В его обязанности входит хранение списка контрольных точек, отслеживание текущей цели, анализ положения аппарата и формирование управляющих воздействий.

Одной из особенностей разработанной системы является использование промежуточных точек. Вместо движения напрямую к удалённой цели маршрут разбивается на последовательность более коротких участков. Такой подход обеспечивает более плавное движение аппарата, уменьшает вероятность значительных отклонений и повышает устойчивость управления.

Для построения промежуточных точек используется алгоритм линейной интерполяции между текущим положением дрона и целевой координатой. Количество промежуточных точек определяется длиной участка маршрута. В результате формируется последовательность небольших перемещений, которые последовательно выполняются автопилотом.

В процессе полёта система непрерывно получает текущие координаты аппарата из телеметрии. На основании этих данных вычисляется расстояние до целевой точки и определяется факт её достижения. Для повышения устойчивости предусмотрен допустимый радиус ошибки, учитывающий особенности моделирования и возможные задержки передачи данных.

Для компенсации отклонений от маршрута используется пропорциональный регулятор. Ошибка между текущим положением и требуемой координатой преобразуется в управляющие скорости по осям. Чем больше отклонение, тем более интенсивная команда формируется системой. Дополнительно реализован механизм сглаживания изменения скоростей, позволяющий избежать резких манёвров и рывков.

Для корректной работы системы компьютерного зрения необходимо обес-

печить постоянное направление передней камеры вдоль траектории движения. С этой целью реализован алгоритм автоматического вычисления угла рыскания. На основании направления движения определяется требуемая ориентация аппарата, после чего автопилоту передаётся команда изменения курса.

Одной из наиболее важных функций разработанного комплекса является обнаружение препятствий. Для решения данной задачи использовалась модель глубокого обучения YOLOv8n. Выбор облегчённой версии модели обусловлен необходимостью работы в условиях виртуальной машины без использования производительной графической подсистемы.

Видеопоток с фронтальной камеры передаётся через ROS и поступает в модуль компьютерного зрения. После предварительной обработки изображение подаётся на вход нейронной сети. Модель выполняет детекцию объектов и возвращает ограничивающие прямоугольники, соответствующие найденным препятствиям.

Для каждого обнаруженного объекта вычисляются координаты центра, размеры ограничивающего прямоугольника и его площадь. Площадь используется как приближённая оценка расстояния до препятствия. Чем больше объект занимает места в кадре, тем ближе он находится к аппарату.

Для исключения ложных срабатываний система игнорирует объекты, размеры которых не превышают установленный порог. Такой подход позволяет сосредоточить внимание только на действительно опасных препятствиях, способных повлиять на безопасность полёта.

После обнаружения препятствия выполняется анализ его положения относительно центра изображения. Кадр условно разделяется на три зоны. Если объект находится слева, система выбирает обход справа. Если препятствие расположено справа, выполняется обход слева. При нахождении объекта по центру формируется дополнительная траектория смещения.

Для реализации обхода была введена двухуровневая система навигации. Первый уровень представляет глобальный маршрут, сформированный пользователем. Второй уровень отвечает за локальную коррекцию траектории при возникновении препятствий.

При обнаружении препятствия создаётся временная точка обхода. После достижения данной точки система строит траекторию возврата к исходному маршруту. Благодаря такому подходу сохраняется возможность продолжения

миссии без полного перестроения маршрута.

Дополнительно реализован механизм памяти препятствий. На практике возможны ситуации, когда объект временно исчезает из кадра вследствие помех или кратковременных ошибок обработки видеопотока. Для предотвращения преждевременного возврата на исходную траекторию система продолжает учитывать препятствие ещё несколько секунд после потери детекции. Это существенно повышает устойчивость работы алгоритма обхода.

Важной частью проекта стала реализация визуальной навигации с использованием ArUco-маркеров. В системе используются две различные метки: стартовая и посадочная. Каждая метка имеет уникальный идентификатор, который позволяет системе определить требуемое действие.

Для генерации маркеров использовались средства библиотеки OpenCV и словарь DICT_4X4_50. Сгенерированные изображения были размещены в виртуальной среде Gazebo в виде текстур специальных площадок.

Распознавание ArUco-маркеров выполняется с помощью нижней камеры дрона. Полученный видеопоток передаётся через ROS в специализированный модуль обнаружения. Перед обработкой изображение переводится в оттенки серого, после чего запускается алгоритм поиска маркеров.

При успешном распознавании OpenCV возвращает идентификатор маркера и координаты его углов. На основании идентификатора система определяет дальнейшее действие. Если обнаружена стартовая метка, начинается процедура взлёта. Если обнаружена посадочная метка, система переходит к выполнению посадки.

Алгоритм посадки состоит из нескольких этапов. Сначала производится обнаружение посадочной площадки. Затем вычисляется отклонение центра маркера относительно центра изображения. На основании полученной ошибки формируются корректирующие команды движения.

Для обеспечения точности посадки используется пропорциональное управление. Чем сильнее маркер смещён относительно центра изображения, тем больше корректирующее воздействие. После выравнивания положения начинается плавное снижение аппарата.

Во время снижения система продолжает анализировать видеопоток и корректировать положение дрона. Такой подход позволяет компенсировать возникающие отклонения и обеспечивает более точное приземление.

Рисунок 2 – Внешний вид интерфейса

В процессе испытаний были выявлены случаи временной потери изображения с камеры во время снижения. Для повышения надёжности алгоритма было принято решение продолжать посадку даже при кратковременной потере маркера после его первоначенного обнаружения. Это позволило избежать ложных прерываний процедуры посадки.

Одним из ключевых компонентов разработанного программного комплекса является пользовательский интерфейс (см.рис. 2). Его основное назначение заключается в обеспечении удобного взаимодействия оператора с системой автономного управления беспилотным летательным аппаратом. Интерфейс позволяет выполнять полный цикл работы с миссией: от формирования маршрута полёта до наблюдения за состоянием дрона во время выполнения поставленной задачи.

В отличие от консольных решений и специализированных программ управления автопилотом, разработанный интерфейс ориентирован на использование в браузере и не требует установки дополнительного программного обеспечения на компьютер пользователя. Такой подход упрощает эксплуатацию системы и позволяет запускать интерфейс на различных устройствах независимо от установленной операционной системы.

Для реализации пользовательской части приложения использовалась биб-

лиотека React. Выбор данной технологии обусловлен её высокой производительностью, компонентным подходом к разработке и возможностью динамического обновления отображаемой информации без перезагрузки страницы. Интерфейс построен как одностраничное приложение, в котором взаимодействие пользователя с системой происходит в реальном времени.

Архитектура интерфейса реализована по клиент-серверной модели. Клиентская часть отвечает за отображение информации и обработку действий пользователя, тогда как серверная часть обеспечивает связь с модулем управления дроном, обработку маршрутов и передачу телеметрических данных. Обмен информацией между клиентом и сервером осуществляется посредством HTTP-запросов и REST API.

Основным элементом пользовательского интерфейса является интерактивная карта. Для её реализации использовалась библиотека pigeon-maps. Карта позволяет пользователю визуально формировать маршрут полёта путём размещения контрольных точек непосредственно на рабочем поле. Такой способ задания маршрута является более наглядным по сравнению с ручным вводом координат и снижает вероятность ошибок при планировании миссии.

При добавлении новой точки маршрута пользователь выполняет щелчок мышью по требуемой области карты. Координаты выбранной позиции автоматически сохраняются в структуре маршрута. Каждая созданная точка отображается специальным маркером, а соседние точки соединяются линиями, образуя предполагаемую траекторию движения беспилотного летательного аппарата. Благодаря этому пользователь получает визуальное представление будущего маршрута ещё до начала выполнения миссии.

После завершения формирования маршрута пользователь может инициировать передачу миссии в систему управления. Для этого используется специальная кнопка запуска миссии. При её нажатии список координат преобразуется в формат JSON и отправляется на серверную часть приложения посредством HTTP POST-запроса. Сервер принимает полученные данные, выполняет их проверку и передаёт сформированный маршрут модулю управления полётом.

Серверная часть интерфейса реализована на языке Python с использованием фреймворка FastAPI. Данный инструмент обеспечивает высокую производительность обработки запросов и удобные средства построения REST API. Сервер выполняет роль промежуточного звена между пользовательским интер-

фейсом и системой управления дроном.

Одной из основных задач серверной части является приём маршрутов, сформированных пользователем. После получения миссии сервер сохраняет её во внутреннем представлении и передаёт соответствующему модулю управления. Кроме того, сервер отвечает за получение телеметрических данных от автопилота и их последующую передачу в клиентское приложение.

Во время выполнения миссии интерфейс предоставляет пользователю возможность наблюдать за состоянием летательного аппарата в режиме реального времени. Для этого реализован отдельный блок отображения телеметрической информации. Телеметрия представляет собой совокупность параметров, характеризующих текущее состояние дрона и ход выполнения полёта.

В интерфейсе отображаются текущие координаты летательного аппарата, получаемые из навигационной системы. Эти данные используются как для текстового отображения, так и для обновления положения маркера дрона на карте. Благодаря этому пользователь может визуально наблюдать перемещение аппарата вдоль заданного маршрута.

Отдельное внимание уделено отображению состояния системы. В интерфейсе предусмотрены текстовые уведомления, информирующие пользователя о ключевых событиях работы программного комплекса. К таким событиям относятся успешное подключение к автопилоту, обнаружение стартовой метки, начало выполнения миссии, обнаружение препятствия, переход в режим обхода, обнаружение посадочной площадки и завершение полёта.

Для отображения местоположения дрона на карте используется отдельный маркер, положение которого непрерывно обновляется в соответствии с поступающими координатами. В результате пользователь может наблюдать движение аппарата практически в режиме реального времени. Одновременно отображается построенный маршрут, что позволяет сравнивать фактическую траекторию полёта с запланированной.

В процессе разработки особое внимание уделялось простоте пользовательского взаимодействия. Интерфейс содержит минимальное количество элементов управления, необходимых для выполнения поставленных задач. Благодаря этому оператор может быстро освоить работу с программным комплексом без необходимости изучения большого количества функций и настроек.

Дополнительным преимуществом разработанного интерфейса является

его независимость от конкретной платформы. Поскольку доступ к системе осуществляется через веб-браузер, пользователь может работать с приложением на различных компьютерах без установки специализированного программного обеспечения. Это делает систему более универсальной и удобной для дальнейшего внедрения.

Таким образом, разработанный пользовательский интерфейс обеспечивает полный набор функций, необходимых для управления автономным полётом беспилотного летательного аппарата. Он объединяет средства планирования маршрута, мониторинга телеметрии, визуального контроля положения дрона и взаимодействия с системой управления, что существенно повышает удобство эксплуатации разработанного программного комплекса.

Разработанный программный комплекс представляет собой полноценный прототип интеллектуального автопилота, объединяющий средства моделирования, навигации, компьютерного зрения и пользовательского взаимодействия. Реализованная архитектура допускает дальнейшее расширение функциональности, подключение дополнительных алгоритмов планирования маршрутов и перенос отдельных компонентов на реальные беспилотные летательные аппараты.