УТВЕРЖДАЮ

Директор

АО «Петрозаводскмаш»

_____ /_____

« » _____ 2025 г.

УТВЕРЖДАЮ

Директор

АО ЦПР «ЭФЭР»

_____ /_____

« » _____ 2025 г.

АВТОНОМНАЯ СИСТЕМА МОНИТОРИНГА ПОЖАРНОЙ БЕЗОПАСНОСТИ НА БАЗЕ БПЛА (АСМПБ «ЭФЭР-ГАРДИАН»)

ОПИСАНИЕ ПРОГРАММЫ

Версия 1.0.0

ЛИСТ УТВЕРЖДЕНИЯ

СОГЛАСОВАНО

Начальник ОИТ-Главный инженер

_____ Беспилотный А.А.

« » _____ 2025 г.

РАЗРАБОТЧИКИ

_____ Дронов В.В.

_____ Летучий К.К.

« » _____ 2025 г.

2025

АВТОНОМНАЯ СИСТЕМА МОНИТОРИНГА ПОЖАРНОЙ БЕЗОПАСНОСТИ НА БАЗЕ БПЛА (АСМПБ «ЭФЭР-ГАРДИАН»)

ОПИСАНИЕ ПРОГРАММЫ

Версия 1.0.0

Листов [20]

2025

АННОТАЦИЯ

Настоящее описание программы распространяется на программное обеспечение (далее – ПО) «Автономная система мониторинга пожарной безопасности на базе БПЛА (АСМПБ «ЭФЭР-ГАРДИАН»)». Описание программы содержит общие сведения о программном комплексе, его функциональное назначение, описание логической структуры, требования к техническим средствам, а также характеристики входных и выходных данных.

СОДЕРЖАНИЕ

1. ОБЩИЕ СВЕДЕНИЯ

1.1 Обозначение и наименование программы

- Полное наименование: Автономная система мониторинга пожарной безопасности на базе БПЛА (АСМПБ «ЭФЭР-ГАРДИАН»).
- Краткое наименование: АСМПБ «ЭФЭР-ГАРДИАН».
- Обозначение: FR-ASFMS-SW-1.0.0.

1.2 ПО, необходимое для функционирования

Для функционирования бортового программного комплекса БПЛА требуется следующее базовое ПО:

- Операционная система бортового вычислителя: Ubuntu Linux 22.04 LTS (или новее) или специализированная ОС на базе Linux, совместимая с ROS 2 Humble. Рекомендуется использовать минимальную серверную установку (Ubuntu Server) или специализированный дистрибутив для робототехники (например, ROS 2 base image). Python входит в дистрибутив ОС.
- Промежуточное ПО для робототехники: ROS 2 Humble Hawksbill (или актуальная LTS-версия) с фреймворком `nav2`, пакетами для SLAM (`slam_toolbox`, `cartographer`) и драйверами устройств.
- Среда выполнения для моделей ИИ: TensorFlow Lite Runtime (версия 2.13.0 и выше).
- Среда автопилота: Прошивка ArduPilot Copter (последняя стабильная версия) на полетном контроллере.
- Шлюз связи: Пакет `ros2_mavros` (или `micro-ROS` для более тесной интеграции) для связи между ROS 2 и автопилотом по протоколу MAVLink.
- Сторонние библиотеки и зависимости:
    - OpenCV (версия 4.8.0+ с поддержкой Python): Для обработки изображений и видео.
    - PCL (Point Cloud Library): Для работы с данными лидара.
    - Eigen, Boost: Математические библиотеки.
    - Python 3.10 или 3.11: Указанные версии являются родными для Ubuntu 22.04 LTS и официально поддерживаются ROS 2 Humble. Использование Python 3.8 не рекомендуется для данной сборки.
- Драйверы и специфичное ПО:
    - Драйверы для камер (например, `librealsense` для камер Intel, `v4l-utils`).

    - Драйверы и SDK для лидара (зависит от модели, например, для RPLIDAR).
    - Драйверы для Google Coral Edge TPU (`libedgetpu`).

1.3 Языки программирования

Основное ПО разработано с использованием следующих языков программирования:

- C++ (17 стандарт и выше): Для реализации высокопроизводительных модулей навигации (SLAM), управления, связи и ядра системы.
- Python (3.10 и выше): Для реализации модулей компьютерного зрения (работа с нейросетевой моделью), логики высокого уровня, REST API клиента/сервера, сценариев автоматизации и тестирования.
- C: Для низкоуровневых компонентов и работы с аппаратным обеспечением в среде автопилота (ArduPilot).

2. ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ

ПО АСМПБ «ЭФЭР-ГАРДИАН» предназначено для обеспечения полностью автономной работы роя БПЛА по непрерывному круглосуточному мониторингу крупногабаритных помещений (ангаров, цехов, складов) с целью раннего обнаружения признаков пожара (открытого пламени и задымления) и оперативного оповещения пожарных служб предприятия.

Основные решаемые задачи:

1. Автономная навигация и картография: Построение карты помещения (SLAM) и автономный полет по заданному маршруту с обходом статических и динамических препятствий.
2. Детекция угроз в реальном времени: Анализ видеопотоков с дневной камеры и тепловизора с использованием нейросетевой модели YOLOv11n для обнаружения классов «пламя» и «дым» с минимальной задержкой.
3. Реакция на инцидент: Автоматическое сближение с обнаруженным очагом, стабилизация, наведение камер и передача сигнала тревоги с координатами и визуальным подтверждением в пожарную сеть предприятия.
4. Управление жизненным циклом: Автоматический возврат на зарядную станцию при низком заряде батареи, точная посадка и инициация процесса зарядки.
5. Обеспечение отказоустойчивости: Реализация сценариев `fail-safe` при потере связи, отказе датчиков или критическом разряде батареи для обеспечения безопасного завершения миссии.
6. Интеграция с инфраструктурой: Двусторонний обмен данными с наземной станцией управления (GCS) и пожарной сетью предприятия по защищенным каналам.

Функциональные ограничения:

- Система предназначена для работы строго внутри заданных геозон (границ контролируемого помещения).
- Эффективность детекции зависит от условий видимости (сильная задымленность может затруднить визуальное обнаружение, повышая роль тепловизора).
- Автономность ограничена емкостью аккумуляторных батарей и временем их зарядки.
- Непрерывность мониторинга зависит от количества дронов в рое.

- Ручное управление через RC-пульт имеет наивысший приоритет и полностью отменяет автономный режим.

3. ОПИСАНИЕ ЛОГИЧЕСКОЙ СТРУКТУРЫ

3.1 Алгоритм программы

Основной алгоритм работы системы представляет собой конечный автомат (State Machine) со следующими ключевыми состояниями и переходами:

1. `STANDBY` (Ожидание): БПЛА на зарядной станции, системы инициализированы, ожидается команда «Взлет» от GCS или диспетчера.
2. `TAKEOFF` (Взлет): Взлет на заданную безопасную высоту.
3. `PATROL` (Патрулирование): Основной режим. Автономное движение по заданным точкам маршрута. Параллельно выполняется обработка видеопотока нейросетью. При обнаружении пламени/дыма – переход в `FIRE_DETECTED`.
4. `FIRE_DETECTED` (Обнаружение пожара): Прерывание маршрута. Сближение с целью, стабилизация и точное наведение камер. Отправка сигнала тревоги с координатами и видео. Ожидание команды «Отбой тревоги» для возврата в `PATROL` или перехода в `RTL` при низком заряде.
5. `RTL` (Возврат на базу): Активируется автоматически при низком заряде батареи или потере связи. БПЛА строит оптимальный маршрут к зарядной станции.
6. `LANDING` (Посадка на зарядку): Точная посадка на зарядную станцию с использованием данных дальномеров и системы визуального позиционирования.
7. `CHARGING` (Зарядка): Процесс зарядки, мониторинг состояния батареи.
8. `FAILSAFE` (Аварийный режим): Универсальное состояние для обработки нештатных ситуаций (отказ датчика, потеря вычислителя). Цель – безопасное завершение полета (например, экстренная посадка на месте).

3.2 Используемые методы

- Компьютерное зрение: Сверточные нейронные сети (CNN), модель YOLOv11n (You Only Look Once) для детекции объектов в реальном времени. Используется оптимизация для edge-устройств с применением TensorFlow Lite и аппаратного ускорителя Google Coral Edge TPU.

- Навигация и SLAM: Алгоритмы одновременной локализации и построения карты (SLAM) на основе данных лидара (например, Cartographer, GMapping или `slam_toolbox` в ROS 2). Для планирования пути используется алгоритм A*/D* Lite и динамическое окно (Dynamic Window Approach, DWA) для локального обхода препятствий.
- Фильтрация данных: Фильтр Калмана и комплементарные фильтры для обработки данных с инерциальных датчиков (IMU), барометра и дальномеров.
- Сетевое взаимодействие: Протокол MAVLink для телеметрии и управления полетом. RESTful API (HTTP/JSON) для интеграции с верхнеуровневыми системами. RTSP для потоковой передачи видео.

3.3 Структура программы

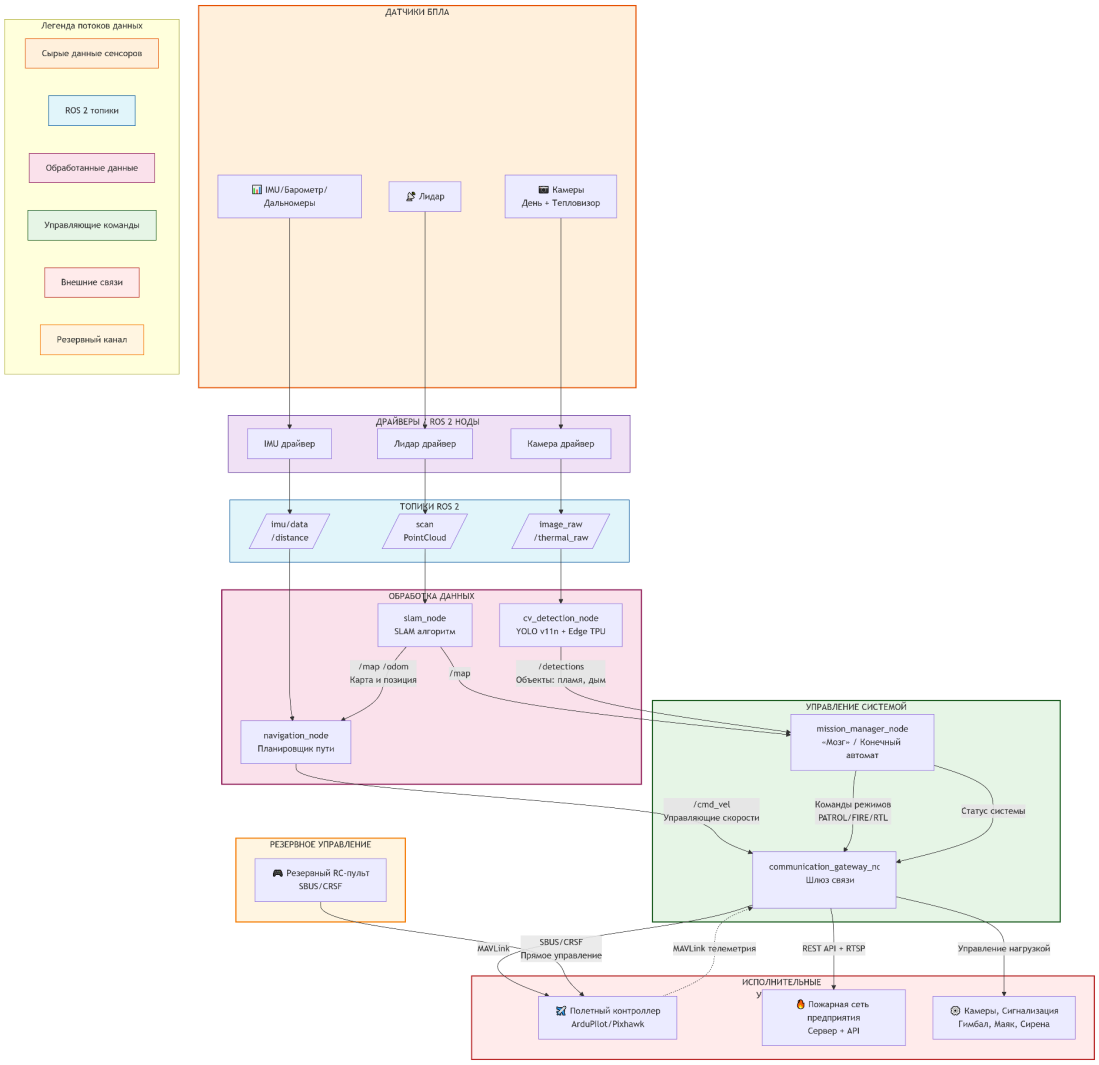
ПО построено по модульному принципу на базе фреймворка ROS 2. Каждый модуль реализован как независимая нода (узел), обменивающаяся данными через топики (асинхронно) и сервисы (синхронно).

Основные программные модули (ноды ROS 2):

1. `slam_node`: Принимает данные с лидара, строит и обновляет карту занятости, оценивает позицию БПЛА.
2. `navigation_node` (Nav2): На основе карты и цели строит глобальный и локальный маршруты, отправляет управляющие скорости полетному контроллеру.
3. `cv_detection_node`: Принимает видеопотоки с камер, выполняет инференс нейросетевой модели на Edge TPU, публикует координаты обнаруженных объектов (пламя, дым) и их уверенность.
4. `mission_manager_node` («Мозг»): Реализует конечный автомат системы. Координирует работу всех модулей, обрабатывает события (детекция, низкий заряд), инициирует смену режимов, исполняет сценарии `fail-safe`.
5. `communication_gateway_node`: Управляет всеми внешними каналами связи. Отвечает за:
    - `mavlink_bridge`: Связь с полетным контроллером через MAVROS.
    - `fire_network_client`: Отправку REST-запросов и RTSP-потока в пожарную сеть.
    - `payload_manager`: Управление камерами (переключение, наведение) и сигнализацией (маячок, сирена).
6. `battery_monitor_node`: Мониторит напряжение и заряд аккумулятора, публикует предупреждения для `mission_manager_node`.
7. `diagnostic_node`: Собирает диагностические данные со всех нод и аппаратных компонентов, ведет лог работы системы.

3.4 Взаимодействие компонентов программного комплекса

Настоящий раздел описывает связи и протоколы взаимодействия между компонентами программного комплекса АСМПБ «ЭФЭР-ГАРДИАН». Комплекс построен по распределенной модульной архитектуре, где взаимодействие происходит как внутри бортового вычислителя, полетного контроллера, так и с внешними системами.



3.4.1 Внутренние связи (внутри бортового вычислителя RPi 5)

Взаимодействие между основными модулями ПО (`mission_manager`, `cv_detection`, `slam_node` и др.) реализовано на базе ROS 2 (Robot Operating System 2).

- Механизм: Публикация/подписка на топики (topics) и вызов сервисов (services) через промежуточный слой DDS (Data Distribution Service).
- Назначение: Этот подход обеспечивает слабую связанность модулей, их независимое развертывание, замену и отладку. Каждый модуль (нода) общается с другими, только зная формат сообщения в топике, но не имея прямой ссылки на исполняемый код другого модуля.
- Ключевые топики:
    - `/sensor/scan` (тип: `sensor_msgs/LaserScan`) - данные лидара от драйвера к `slam_node`.
    - `/sensor/image_raw` и `/sensor/thermal_raw` (тип: `sensor_msgs/Image`) - кадры с камер к `cv_detection_node`.
    - `/cv/detections` (тип: `vision_msgs/DetectionArray`) - результаты детекции (пламя, дым) от `cv_detection_node` к `mission_manager_node`.
    - `/map` и `/localization/pose` - карта и позиция БПЛА от `slam_node` к `navigation_node` и `mission_manager_node`.
    - `/navigation/cmd_vel` (тип: `geometry_msgs/Twist`) - целевые скорости от `navigation_node` к `communication_gateway_node`.

3.4.2 Связи с бортовым аппаратным обеспечением и автопилотом

- С полетным контроллером (Pixhawk): Осуществляется через протокол MAVLink.
    - Компонент: Модуль `mavlink_bridge` (часть `communication_gateway_node`).

    - Физический интерфейс: Последовательный порт (UART/SPI).
    - Данные: программный комплекс отправляет автопилоту высокоуровневые команды (взлет, посадка, полет в точку, целевые скорости), а также команды управления полезной нагрузкой. Автопилот возвращает сырую телеметрию (сырые данные IMU, состояние моторов) и обработанную навигационную информацию (высота, позиция по внутренней фильтрации).
- С датчиками (Лидар, Камеры, Edge TPU): Осуществляется через драйверы ОС Linux и/или специализированные SDK.
    - Интерфейсы: USB 3.0 (для камер и Coral Edge TPU), UART (для многих лидаров).
    - Драйверы, входящие в состав программного комплекса (или устанавливаемые как зависимости), предоставляют данные сенсоров в виде ROS-топиков, делая их доступными для обработки другими модулями.

3.4.3 Внешние связи (с инфраструктурой заказчика и оператором)

- С Наземной станцией управления (GCS - Mission Planner / QGroundControl):
    - Протокол: MAVLink, поверх радиоканала (900 МГц / 2.4 ГГц).
    - Инициатор: Канал инициируется и поддерживается полетным контроллером (Pixhawk). Программный комплекс на RPi5 может транслировать свои кастомные телеметрические сообщения (логи ROS, статус детекции), используя тот же MAVLink-канал.
- С Пожарной сетью предприятия:
    - Протоколы: REST API over HTTP(S) (для команд и событий) и RTSP (для видеопотока).
    - Компонент: Модуль `fire_network_client` (часть `communication_gateway_node`).

    - Данные: Отправка JSON-сообщений о событиях (тревога), статусе. Прием команд ("Отбой тревоги"). Потоковая передача видео по RTSP. *Форматы JSON-сообщений приведены в Приложении Б ТЗ*.
- С Резервным каналом управления (RC-пульт):
    - Протокол: Цифровой протокол управления (SBUS/CRSF).
    - Связь: Прямая аппаратная связь между RC-приемником и полетным контроллером. Программный комплекс не участвует в этой связи. Это "последняя линия обороны" оператора, имеющая наивысший приоритет и полностью отменяющая любые автономные команды, сгенерированные ПО.

Таким образом, "связи программы" в этом контексте — это архитектура взаимодействия распределенных компонентов программного комплекса, использующая:

1. ROS 2 — для внутренней, слабосвязанной коммуникации модулей.
2. MAVLink — как "мост" к низкоуровневому автопилоту и канал к GCS.
3. Стандартные сетевые протоколы (REST, RTSP) — для интеграции с инфраструктурой заказчика.
4. Прямые аппаратные протоколы (SBUS) — для обеспечения аварийного управления, лежащего вне ПО.

Это разделяет ответственность и обеспечивает отказоустойчивость: отказ ПО на RPi5 (например, модуля связи) не лишает оператора возможности взять управление на себя через RC-пульт и посадить БПЛА.

4. ИСПОЛЬЗУЕМЫЕ ТЕХНИЧЕСКИЕ СРЕДСТВА

ПО предназначено для работы на следующей бортовой аппаратной платформе БПЛА:

Компонент	Модель / Требования
Бортовой вычислитель	Одноплатный компьютер Raspberry Pi 5, ОЗУ не менее 8 ГБ. Основной модуль, на котором работает ROS-2.
Аппаратный ускоритель ИИ	Google Coral USB Accelerator (Edge TPU).
Полетный контроллер	Pixhawk 6C / Cube Orange+ (или аналогичные) с прошивкой ArduPilot.
Основные датчики	2D/3D Лидар (напр., RPLIDAR A3 / Livox Mid-40). Комбинированная камера + тепловизор. IMU, барометр, сонары / дальномеры (вниз/вперед/вбок).
Коммуникационные модули	Радиомодем 900 МГц для MAVLink. Wi-Fi модуль 802.11ac. RC-приемник 2.4 ГГц (SBUS/CRSF). Аналоговый / цифровой VTX 5.8 ГГц.
Периферия	Сервопривод / гимбал для камер. Светозвуковая сигнализация (маячок, сирена).

5. ВЫЗОВ И ЗАГРУЗКА

1. Загрузка: ПО загружается автоматически после подачи питания на бортовой вычислитель (Raspberry Pi). Запускается операционная система, которая автоматически инициирует запуск ROS-2 - нод через системный демон (systemd) или launch-файлы.
2. Инициализация: Последовательно инициализируются все модули: проверяются датчики, загружается нейросетевая модель в память Edge TPU, устанавливаются соединения с полетным контроллером и сетями.
3. Переход в режим ожидания: После успешной инициализации система переходит в состояние `STANDBY`. На наземную станцию управления (GCS) и в пожарную сеть (при наличии связи) передается статус «Готов к работе».
4. Запуск миссии: Для начала патрулирования оператор с GCS отправляет команду «Взлет» (через MAVLink), что приводит к переходу системы в состояние `TAKEOFF`, а затем `PATROL`.

6. ВХОДНЫЕ ДАННЫЕ

- Данные от сенсоров (сырые):
  - Точечное облако (Point Cloud) от лидара.
  - Кадры видеопотока (RAW, RGB) от дневной камеры.
  - Термограммы (массив температур) от тепловизора.
  - Показания IMU (ускорение, угловая скорость).
  - Показания барометра (давление).
  - Данные с ультразвуковых/лазерных дальномеров.
- Команды от внешних систем:
  - От GCS (через MAVLink): Команды взлета, посадки, загрузки маршрута, изменения параметров, запрос телеметрии.
  - От Пожарной сети (через REST API): Команда «ОТБОЙ ТРЕВОГИ», запрос на смену режима камеры (видео/тепловизор).
  - От RC-пульта (через SBUS/CRSF): Низкоуровневые команды управления по каналам: газ, крен, тангаж, рыскание, переключатели режимов (наивысший приоритет).
- Конфигурационные файлы:
  - Параметры ROS-нод (`.yaml`).
  - Карта помещения, предварительно построенная или обновляемая онлайн.
  - Маршрут патрулирования (список локальных координат).
  - Нейросетевая модель в формате `.tflite` (для Edge TPU).
  - Параметры взаимодействия с пожарной сетью заказчика.

7. ВЫХОДНЫЕ ДАННЫЕ

- Данные для управления полетом:
    - Управляющие команды для полетного контроллера (целевые скорости по осям, углы, положение) через MAVLink.
- Оперативная информация для внешних систем:
    - Для GCS: Полная телеметрия MAVLink (положение, скорость, ориентация, состояние батареи, режим работы), диагностические сообщения, логи.
    - Для Пожарной сети (REST API / JSON):
        - *События:* `{"event": "FIRE_DETECTED", "drone_id": "01", "confidence": 0.89, "class": "flame", "local_coords": [x,y,z], "timestamp": "..."}`.
        - *Статус:* `{"status": "PATROLLING", "battery_level": 85, ...}`.
        - *Аварии:* `{"event": "SYSTEM_FAILURE", "component": "LIDAR", ...}`.
    - Для Пожарной сети (RTSP): Видеопоток H.264 с графическим наложением (bounding boxes) вокруг обнаруженных объектов (пламя, дым).
- Управление полезной нагрузкой:
    - Сигналы на сервопривод гимбала для наведения камер.
    - Команды на включение/выключение проблескового маячка и сирены.
- Логи и записи:
    - Полетные логи (черный ящик) в формате ArduPilot `.bin`.
    - Логи работы ROS-нод (`rosbag2` записи).
    - Лог детекций (время, координаты, уверенность, класс).
    - Логи взаимодействия с пожарной сетью заказчика (команды управления, передача данных).
    - Логи перехода дрона из одного рабочего состояния в другое, в т.ч. в fail-safe.

8. ТЕРМИНЫ И СОКРАЩЕНИЯ

8.1 Термины и определения

Термин	Определение
ROS 2	Robot Operating System 2 — набор ПО, библиотек и инструментов для разработки робототехнических систем.
Нода (Узел)	Исполняемый процесс в рамках графа ROS 2, выполняющий определенную вычислительную задачу.
Топик	Канал асинхронной передачи данных между нодами ROS 2 по принципу «издатель-подписчик» (publisher-subscriber).
SLAM	Simultaneous Localization and Mapping — одновременная локализация и построение карты.
MAVLink	Легкий протокол обмена сообщениями для связи с бортовым оборудованием (автопилотами) и наземными станциями.
REST API	Архитектурный стиль для построения веб-сервисов; в данном контексте — интерфейс обмена данными в формате JSON по HTTP(S).

Термин	Определение
TensorFlow Lite	Облегченная версия фреймворка машинного обучения TensorFlow, предназначенная для запуска моделей на мобильных и edge-устройствах.
Edge TPU	Специализированный тензорный процессор (TPU) от Google для ускорения нейронных сетей на периферийных устройствах.
Геозона	Виртуальный географический периметр, выход за пределы которого запрещен или вызывает определенную реакцию БПЛА.
Fail-Safe	Заранее запрограммированный безопасный сценарий действий системы при возникновении сбоя или отказа.

8.2 Перечень сокращений

Сокращение	Расшифровка
ПО	Программное обеспечение
БПЛА	Беспилотный летательный аппарат
GCS	Ground Control Station — наземная станция управления

IMU	Inertial Measurement Unit — инерциальный измерительный модуль (акселерометр + гироскоп)
LIDAR	Light Detection and Ranging — оптическая система дистанционного зондирования, основанная на использовании лазера
FPV	First Person View — вид «от первого лица», видеопоток с бортовой камеры
CV	Computer Vision — компьютерное зрение
AI / ИИ	Artificial Intelligence / Искусственный интеллект
RTL	Return To Launch — возврат к точке взлета (в данном контексте — возврат на зарядную станцию)
JSON	JavaScript Object Notation — текстовый формат обмена данными
RTSP	Real Time Streaming Protocol — протокол потоковой передачи в реальном времени
FPS	Frames Per Second — кадров в секунду