УТВЕРЖДАЮ

Директор

АО «Петрозаводскмаш»

_____ /_____

« » _____ 2025 г.

УТВЕРЖДАЮ

Директор

АО ЦПР «ЭФЭР»

_____ /_____

« » _____ 2025 г.

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

РУКОВОДСТВО ПРОГРАММИСТА

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»)

РУКОВОДСТВО ПРОГРАММИСТА

Версия 1.0.0

Листов [27]

2025

АННОТАЦИЯ

Настоящее руководство программиста распространяется на программное обеспечение (далее – ПО) «Автономная система мониторинга пожарной безопасности на базе БПЛА (АСМПБ «ЭФЭР-ГАРДИАН»)» версии 1.0.0.

Руководство содержит технические характеристики программного комплекса, описание его архитектуры, процедуры сборки, установки, настройки, обновления и отладки. Документ предназначен для инженеров-программистов, осуществляющих развертывание, сопровождение и модификацию системы.

СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ

1.1 Область применения

Руководство программиста распространяется на программный комплекс автономной системы мониторинга пожарной безопасности на базе БПЛА (АСМПБ «ЭФЭР-ГАРДИАН»). Документ предназначен для использования специалистами (программистами, инженерами-разработчиками, системными интеграторами) предприятия-разработчика (АО ЦПР «ЭФЭР») и сервисными инженерами заказчика (АО «Петрозаводскмаш») при установке, настройке, обновлении, отладке и техническом сопровождении системы.

1.2 Краткое описание возможностей

Программный комплекс обеспечивает выполнение следующих ключевых функций:

- Полностью автономное патрулирование БПЛА по заданному маршруту внутри помещений.
- Одновременное построение карты (SLAM) и локализация.
- Детекция очагов возгорания (классы «пламя» и «дым») в видеопотоке с использованием нейросетевой модели YOLOv11n, оптимизированной для работы на периферийном тензорном процессоре (Edge TPU).
- Автоматическая реакция на обнаружение угрозы: сближение, стабилизация, передача тревожного сигнала с координатами в пожарную сеть предприятия.
- Управление жизненным циклом: автоматический возврат на зарядную станцию при низком заряде, точная посадка и зарядка.
- Реализация отказоустойчивого поведения (fail-safe) при потере связи или отказе компонентов.
- Интеграция с внешними системами управления (GCS) и мониторинга (пожарная сеть) по стандартным протоколам (MAVLink, REST API, RTSP).

1.3 Уровень подготовки программиста

Для эффективной работы с данным руководством и программным комплексом персонал должен обладать следующими знаниями и навыками:

- Обязательные:
    - Опыт администрирования операционных систем семейства Linux (Ubuntu/Debian) на уровне командной строки (bash).

    - Понимание основ сетевых технологий (IP-адресация, порты, базовые утилиты диагностики: `ping`, `ssh`, `ifconfig`).
    - Базовые знания архитектуры и принципов работы фреймворка Robot Operating System 2 (ROS 2) (понятия ноды, топика, сервиса, launch-файла).
    - Навыки работы с системами контроля версий, в частности Git.
- Рекомендуемые:
    - Опыт работы с ArduPilot/PX4 и протоколом MAVLink.
    - Понимание основ Python и C++ для чтения кода и простейшей модификации.
    - Базовое знакомство с концепциями компьютерного зрения и машинного обучения (форматы моделей, инференс).
    - Знание основных протоколов прикладного уровня: HTTP/REST, RTSP.

2. НАЗНАЧЕНИЕ И УСЛОВИЯ ПРИМЕНЕНИЯ

2.1. Назначение программного комплекса

ПО АСМПБ «ЭФЭР-ГАРДИАН» является ключевым интеллектуальным компонентом системы, преобразующим аппаратную платформу БПЛА в автономного робота для мониторинга безопасности. Оно предназначено для:

- Автоматизации процесса обнаружения пожаров в труднодоступных зонах крупных объектов.
- Снижения рисков для персонала за счет исключения его нахождения в опасных зонах при обходах.
- Обеспечения непрерывного и круглосуточного контроля за счет роя автономных БПЛА.
- Предоставления службам быстрого реагирования визуального подтверждения и точных координат инцидента.

2.2. Условия применения

2.2.1. Аппаратные требования (бортовой комплекс):

- Вычислитель: Одноплатный компьютер Raspberry Pi 5, модель с 8 ГБ ОЗУ.
- Аппаратный ускоритель ИИ: Google Coral USB Accelerator (Edge TPU).
- Полетный контроллер: Pixhawk 6C / Cube Orange+ с интерфейсами UART, I2C, SPI.
- Сенсоры:
    - Лидар: 2D-лидар типа RPLIDAR A3 (подключение via UART-USB).
    - Камера: Комбинированный модуль, включающий неохлаждаемый микроболометрический тепловизор (например, FLIR Lepton) и дневную камеру HD/4K.
    - IMU, барометр, дальномеры: Встроены в полетный контроллер или установлены как отдельные модули.
- Коммуникации:
    - Wi-Fi модуль (встроенный в RPi 5 или внешний USB) стандарта 802.11ac/ax.
    - Радиомодем для связи с GCS (например, Holybro Sik Telemetry Radio V3, 915 МГц).
    - RC-приемник цифрового протокола (например, TBS Crossfire (CRSF) или FrSky (SBUS)).
- Периферия: Сервопривод/гимбал для камер, проблесковый маячок, звуковой излучатель.

2.2.2. Программные требования (бортовой вычислитель):

- Операционная система: Ubuntu Linux 22.04 LTS (архитектура aarch64/arm64). Рекомендуется использование специализированного образа для Raspberry Pi.
- Промежуточное ПО для робототехники: ROS 2 Humble Hawksbill (полная установка Desktop).
- Прошивка автопилота: ArduPilot Copter стабильной ветки версии 4.4.x.
- Дополнительные библиотеки и зависимости:
    - MAVROS (пакет `ros-humble-mavros`) для связи с автопилотом.
    - TensorFlow Lite Runtime версии 2.14.0 и выше.
    - OpenCV (Python и C++ bindings) версии 4.8.0 и выше.
    - Драйверы устройств: `librealsense` (для совместимых камер), `rplidar_ros` (для лидара), `libedgetpu` (для Coral USB Accelerator).
    - Среда выполнения Python: Python 3.10.
- Среда разработки/сборки (рекомендуемая для ПК разработчика):
    - ПК с ОС: Ubuntu 22.04 LTS или Windows 10/11 с WSL2 (Ubuntu 22.04).
    - Инструменты: ROS 2 Humble, colcon, git, VS Code с расширениями для ROS 2 и Python.

3. ХАРАКТЕРИСТИКА ПРОГРАММЫ

3.1. Архитектура и основные компоненты

ПО построено по модульному принципу на основе фреймворка ROS 2. Каждый модуль реализован как независимая нода (узел), с обменом данными через топики (асинхронно) и сервисы (синхронно) с использованием промежуточного слоя DDS.

Ключевые программные ноды:

1. `mission_manager_node` (Ядро системы / FMS): Реализует конечный автомат (State Machine) системы. Координирует все модули, обрабатывает события (детекция, низкий заряд), управляет переходами между режимами (`STANDBY`, `PATROL`, `FIRE_DETECTED`, `RTL` и др.).
2. `cv_detection_node` (Модуль компьютерного зрения): Принимает видеопотоки с камер, выполняет инференс нейросетевой модели YOLOv11n в формате `.tflite` на Edge TPU. Публикует координаты и уверенность обнаруженных объектов (`flame`, `smoke`) в топик `/cv/detections`.
3. `slam_node` (Модуль навигации и картографии): Обрабатывает данные с лидара (топик `/sensor/scan`). Реализует алгоритм SLAM (на базе `slam_toolbox` или `cartographer`) для построения карты занятости (`/map`) и определения позиции БПЛА (`/localization/pose`).
4. `navigation_node` (Планировщик маршрута): На основе карты от `slam_node` и цели от `mission_manager` строит глобальный и локальный путь. Использует стек Nav2 для планирования (A*/D* Lite) и локального обхода препятствий (DWA). Публикует управляющие скорости в топик `/navigation/cmd_vel`.
5. `communication_gateway_node` (Шлюз связи): Агрегирует функции связи:
    - `mavlink_bridge`: Обеспечивает связь между ROS 2 и автопилотом через MAVROS (трансляция телеметрии, отправка команд).
    - `fire_network_client`: Отвечает за интеграцию с пожарной сетью: отправка JSON-событий по REST API, потоковая передача видео по RTSP.
    - `payload_manager`: Управляет полезной нагрузкой: переключение режимов камеры (видео/тепловизор), наведение гимбала, управление светозвуковой сигнализацией.
6. `battery_monitor_node`: Регулярно опрашивает данные о напряжении аккумулятора с автопилота (через MAVROS) и публикует предупреждения при достижении пороговых значений.

7. `diagnostic_node`: Собирает диагностические данные со всех нод (загрузка CPU, память, статус) и публикует их в стандартизированном топике `/diagnostics`.

3.2. Временные и эксплуатационные характеристики

- Частота работы ключевых нод:
  - `mission_manager_node`: 10 Гц (цикл принятия решений).
  - `cv_detection_node`: 15-20 Гц (при использовании Edge TPU). При отказе Edge TPU и работе на CPU RPi 5 — 3-5 Гц.
  - `slam_node`: 10 Гц (синхронно с частотой обновления лидара).
  - Планирование пути (`navigation_node`): 10 Гц.
  - Цикл управления полетом (MAVROS -> автопилот): 50 Гц.
- Режим работы: Непрерывный, циклический. Работает до получения команды на остановку или до заданного в конфигурации ПО разряда АКБ с последующей автономной зарядкой.
- Требования к памяти:
  - Оперативная память (ОЗУ): Минимум 4 ГБ для базовой работы. Для стабильной работы всех компонентов, включая SLAM и нейросеть, рекомендуется 8 ГБ.
  - Постоянная память (ПЗУ): ~10-15 ГБ (включая ОС, ROS 2, все библиотеки, исходный код, модель ИИ, место для логов).
- Средства контроля и восстановления:
  - Каждая нода публикует стандартизированные диагностические сообщения (ROS 2 DiagnosticArray).
  - Критические ноды (особенно `mission_manager`) защищены watchdog-механизмами, способными инициировать их перезапуск при зависании.
  - Система управления службами systemd используется для автозапуска всего ПО при старте системы и его мониторинга.
  - Подробное логирование в несколько каналов: консольные логи, файловые логи нод, структурированные записи `rosbag2`, полетные логи автопилота (`.bin`).
  - Реализована полная логика fail-safe согласно ТЗ (см. раздел «Сообщения»).

4. ОБРАЩЕНИЕ К ПРОГРАММЕ

4.1. Сборка программного комплекса из исходного кода (на ПК разработчика или целевом RPi)

1. **Подготовка среды**: Установите ROS 2 Humble и необходимые зависимости.

```
bash

sudo apt update && sudo apt upgrade

sudo apt install python3-colcon-common-extensions ros-humble-desktop ros-humble-navigation2 ros-humble-slam-toolbox ros-humble-vision-msgs ros-humble-mavros ros-humble-mavros-extras python3-pip

pip3 install tflite-runtime opencv-python
```

2. **Получение исходного кода**: Создайте рабочее пространство и склонируйте репозиторий.

```
bash

mkdir -p ~/fr_guardian_ws/src

cd ~/fr_guardian_ws/src

git clone <URL_РЕПОЗИТОРИЯ_АО_ЦПР_ЭФЭР> .
```

3. **Установка специфичных зависимостей**:

```
bash

rosdep install --from-paths . --ignore-src -r -y
```

4. **Сборка проекта**: Используйте систему сборки `colcon`.

```
bash

cd ~/fr_guardian_ws

colcon build --symlink-install --cmake-args
-DCMAKE_BUILD_TYPE=Release
```

Флаг `--symlink-install` *ускоряет последующие сборки.*

4.2. Установка и первоначальный запуск на целевом устройстве (БПЛА)

1. Развертывание образа: Скопируйте предварительно собранный образ SD-карты на целевой RPi 5 или выполните шаги 4.1 непосредственно на устройстве.
2. Настройка автопилота:
   - Подключите полетный контроллер к ПК с Mission Planner / QGroundControl.
   - Загрузите файл с параметрами (`config/ardupilot_params.param`), поставляемый в составе ПО.
   - Проверьте и при необходимости настройте калибровку датчиков, порты, геозону.
3. Настройка конфигурации ПО:
   - Основные параметры системы находятся в YAML-файлах: `~/fr_guardian_ws/install/fr_guardian/share/fr_guardian/config/`.
   - Обязательно отредактируйте:
     - `network_config.yaml`: IP-адреса сервера пожарной сети, настройки RTSP.
     - `hardware_ports.yaml`: Пути к устройствам (`/dev/ttyACM0` для автопилота, `/dev/videoX` для камер).
     - `detection_params.yaml`: Путь к модели `.tflite`, пороги уверенности.
4. Первый запуск:
   - Активируйте рабочее пространство.

```
bash

source ~/fr_guardian_ws/install/setup.bash
```

- Запустите основной launch-файл, который инициирует все ноды.

```
bash

ros2 launch fr_guardian main_launch.py
```

5. Настройка автозапуска (как системной службы):
   - В составе ПО имеется скрипт `scripts/install_as_service.sh`.
   - Выполните его с правами суперпользователя. Он создаст и активирует службу `systemd` с именем `fr-guardian`.

```
bash

sudo bash ~/fr_guardian_ws/src/scripts/install_as_service.sh

sudo systemctl enable fr-guardian

sudo systemctl start fr-guardian
```

4.3. Обновление ПО

1. Остановите текущую работу системы (`Ctrl+C` в терминале или `sudo systemctl stop fr-guardian`).
2. Обновите исходный код из репозитория в ветке `main`.

```
bash

cd ~/fr_guardian_ws/src

git pull origin main
```

3. Пересоберите проект.

```
bash

cd ~/fr_guardian_ws

colcon build --symlink-install
```

4. Перезапустите систему (`ros2 launch...` или `sudo systemctl start fr-guardian`).

4.4. Обновление нейросетевой модели

1. Подготовка модели: Новую модель YOLO, обученную на актуальных данных, необходимо сконвертировать в формат TensorFlow Lite, оптимизированный для Edge TPU. Используйте скрипты в директории `tools/model_conversion/`.
2. Развертывание модели: Полученный файл (например, `yolo11n_guardian_edgetpu.tflite`) скопируйте в директорию хранения моделей на RPi (по умолчанию `~/models/`).
3. Активация модели: Укажите путь к новой модели в конфигурационном файле `detection_params.yaml` (параметр `model_path`) и перезапустите ноду `cv_detection_node` или всю систему.

5. ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ

5.1. Входные данные

- Данные от сенсоров (сырые):
  - Лидар: Последовательность измерений дальности (тип ROS: `sensor_msgs/LaserScan`). Топик: `/sensor/scan`.
  - Камеры: Кадры изображения (тип: `sensor_msgs/Image`). Топики: `/sensor/image_raw` (дневная), `/sensor/thermal_raw` (тепловизор).
  - IMU/Барометр: Данные ориентации, ускорения, высоты (тип: `sensor_msgs/Imu`, `sensor_msgs/FluidPressure`). Доставляются через MAVROS.
- Команды управления:
  - От наземной станции (GCS): MAVLink-команды (взлет, посадка, загрузка миссии). Обрабатываются MAVROS и транслируются в топики ROS (например, `mavros/setpoint_position/local`).
  - От пожарной сети: HTTP POST-запросы в формате JSON на эндпоинт `/api/command`. *Примеры команд от пожарной сети смотри в Приложении Б ТЗ.*
  - Параметры конфигурации: YAML-файлы, содержащие настройки нод, IP-адреса, порты, пути к файлам.

5.2. Выходные данные

- Данные для управления полетом:
  - Целевые скорости и позиции (тип: `geometry_msgs/Twist`, `geometry_msgs/PoseStamped`). Публикуются в топик `/navigation/cmd_vel` и передаются в MAVROS для отправки на автопилот.
- Оперативная информация для внешних систем:
  - Для GCS: Расширенная телеметрия MAVLink, включающая кастомные сообщения с состоянием нод, результатами детекции. Отправляется через радиомодем.
  - Для пожарной сети:
    - События: JSON-сообщения, отправляемые через HTTP POST на сервер заказчика. *Примеры сообщений в пожарную сеть смотри в Приложении Б ТЗ.*
    - Видеопоток: RTSP-стрим с наложенными bounding boxes (bbox) обнаруженных объектов. Адрес потока: `rtsp://<BROADCAST_IP>:8554/guardian_stream`.
- Логи и диагностика:

- Файловые логи: Текстовые логи нод в директории `~/.ros/log/`.
- Бинарные логи ROS: Записи в формате rosbag2 (`.db3`) по топикам. Включаются/выключаются параметром `enable_rosbag`.
- Полетные логи: Логи автопилота в формате `.bin`, сохраняемые на SD-карте полетного контроллера.

6. СООБЩЕНИЯ

Система генерирует сообщения различного уровня для информирования программиста/оператора. Основные каналы: консоль ROS, системный журнал (`journalctl`), диагностические топики, REST API.

6.1. Сообщения в консоль ROS (уровень INFO / ERROR / WARN)

- При запуске:
  - `[mission_manager]: INFO: System initialized. Current state: STANDBY.`
  - `[cv_detection]: INFO: Neural network model loaded successfully from /home/pi/models/model_edgetpu.tflite.`
  - `[mavlink_bridge]: INFO: Connected to FCU at /dev/ttyACM0, system ID: 1.`
- При нормальной работе:
  - `[mission_manager]: INFO: Transition from PATROL to FIRE_DETECTED.`
  - `[cv_detection]: INFO: Detection: flame at [x=320, y=240, z=9], conf=0.92.`
  - `[fire_network_client]: INFO: Alarm sent successfully to fire server. Response code: 200.`
- Предупреждения (WARN):
  - `[battery_monitor]: WARN: Battery level critical (15%). Initiating RTL.`
  - `[communication_gateway]: WARN: Wi-Fi signal low (-75 dBm).`
  - `[cv_detection]: WARN: Edge TPU not found. Falling back to CPU (performance degraded).`
- Ошибки (ERROR):
  - `[slam_node]: ERROR: No data from LIDAR on topic /sensor/scan for 5.0s.`
  - `[mavlink_bridge]: ERROR: FCU connection lost.`
  - `[fire_network_client]: ERROR: Failed to send alarm. Network unreachable.`

6.2. Диагностические сообщения (ROS 2 DiagnosticArray)

Публикуются в топик `/diagnostics`. Содержат структурированную информацию о состоянии каждого узла (уровень загрузки CPU, использование памяти, статус сенсоров).

6.3. Действия программиста по сообщениям

- При ошибке `No data from LIDAR`: Проверить физическое подключение лидара, питание, наличие драйвера `rplidar_ros`. Запустить команду `ros2 topic echo /sensor/scan`.
- При предупреждении `Edge TPU not found`: Проверить подключение Coral USB Accelerator (`lsusb`). Убедиться, что загружены драйверы `libedgetpu`.
- При ошибке `FCU connection lost`: Проверить USB-кабель к полетному контроллеру, питание контроллера, правильность настроек порта в MAVROS.
- При ошибке сети от `fire_network_client`: Проверить доступность Wi-Fi сети, IP-адреса и порты сервера в конфигурации.

7. ТЕРМИНЫ И СОКРАЩЕНИЯ

7.1 Термины и определения

Термин	Определение
Нода (Узел ROS)	Исполняемый процесс в графе ROS 2, выполняющий вычисления и обмен данными с другими нодами.
Топик (Topic)	Именованный канал, по которому ноды обмениваются сообщениями по принципу «издатель-подписчик».
FMS (Flight Management System)	В контексте ПО – высокоуровневый модуль управления (`mission_manager_node`), реализующий логику конечного автомата системы.
MAVLink	Легкий протокол обмена сообщениями для связи с бортовым оборудованием (автопилотами).

MAVROS	Пакет ROS, обеспечивающий связь между ROS и устройством, использующим протокол MAVLink.
SLAM	Simultaneous Localization and Mapping – одновременная локализация и построение карты.
Тензорный процессор (TPU)	Специализированный процессор для ускорения операций линейной алгебры, используемых в нейронных сетях.
Launch-файл	XML- или Python-файл в ROS, позволяющий запускать несколько нод и задавать их параметры одновременно.

7.2 Перечень сокращений

Сокращение	Расшифровка	Подробное описание
ПО	Программное обеспечение	Совокупность программ, процедур, правил и соответствующей документации, относящихся к функционированию системы обработки данных. В контексте данного документа – комплекс программных модулей АСМПБ «ЭФЭР-ГАРДИАН».
БПЛА	Беспилотный летательный аппарат	Летательный аппарат без экипажа на борту, управляемый автоматически или оператором дистанционно. В системе используется мультироторный БПЛА (квадрокоптер).
GCS	Ground Control Station (Наземная станция управления)	Рабочее место оператора, оснащенное специализированным программным обеспечением (Mission Planner, QGroundControl) и аппаратурой для планирования

		миссий, управления и мониторинга состояния БПЛА.
ROS	Robot Operating System	Набор программных библиотек и инструментов для разработки робототехнических систем. В проекте используется версия ROS 2 (Humble Hawksbill) как основа для межмодульного взаимодействия.
CV	Computer Vision (Компьютерное зрение)	Область искусственного интеллекта, занимающаяся автоматическим извлечением, анализом и пониманием полезной информации из изображений или видеопоследовательностей. В системе реализована для детекции пламени и дыма.
ИИ/ AI	Искусственный интеллект / Artificial Intelligence	Способность компьютерных систем выполнять задачи, обычно требующие человеческого интеллекта. В проекте используется для анализа видеопотока с помощью нейронных сетей.

FPS	Frames Per Second (Кадров в секунду)	Единица измерения частоты кадров, количество кадров видеопотока, обрабатываемых или передаваемых за одну секунду. Ключевой параметр производительности модуля детекции.
REST API	Representational State Transfer Application Programming Interface	Архитектурный стиль построения распределенных веб-сервисов. В системе используется для программного взаимодействия с сервером пожарной сети предприятия (обмен JSON-сообщениями).
RTSP	Real Time Streaming Protocol	Сетевой протокол для управления потоковой передачей мультимедиа в реальном времени. Используется для трансляции видеопотока с борта БПЛА на сервер пожарной сети.

IMU	Inertial Measurement Unit (Инерциальный измерительный модуль)	Электронное устройство, состоящее из акселерометров, гироскопов и иногда магнитометров, предназначенное для измерения линейного ускорения, угловой скорости и ориентации объекта в пространстве.
LIDAR	Light Detection and Ranging	Оптическая технология дистанционного зондирования, измеряющая расстояние до объекта с помощью лазерных импульсов. В системе используется для построения карты помещения и навигации.
RPi	Raspberry Pi	Серия одноплатных компьютеров малого размера. В проекте используется модель Raspberry Pi 5 в качестве основного бортового вычислителя.

USB	Universal Serial Bus	Стандарт последовательной шины для подключения периферийных устройств к компьютеру. Используется для подключения камер, Edge TPU и других датчиков к RPi.
YAML	YAML Ain't Markup Language	Человеко-читаемый язык сериализации данных, часто используемый для конфигурационных файлов. В системе применяется для хранения параметров ROS-нод.
JSON	JavaScript Object Notation	Текстовый формат обмена данными, основанный на синтаксисе JavaScript. Основной формат для сообщений, передаваемых между системой и пожарной сетью предприятия.
FMS	Flight Management System (Система управления полетом)	В контексте проекта – высокоуровневый программный модуль (`mission_manager_node`), реализующий конечный автомат

		системы, ее основную логику и координацию всех подсистем.
SLAM	Simultaneous Localization and Mapping	Одновременная локализация и построение карты. Алгоритмическая задача построения карты неизвестной среды с одновременным отслеживанием текущего местоположения агента (БПЛА) внутри нее.
MAVLink	Micro Air Vehicle Link	Протокол обмена сообщениями для связи с бортовым оборудованием беспилотных аппаратов (автопилотами). Используется для связи между полетным контроллером, наземной станцией и бортовым вычислителем.
TPU	Tensor Processing Unit (Тензорный процессор)	Специализированная интегральная схема (ASIC), разработанная Google для ускорения операций линейной алгебры, лежащих в основе нейронных сетей. В проекте

		используется Google Coral Edge TPU.
FCU	Flight Control Unit (Блок управления полетом)	Аппаратно-программный комплекс, отвечающий за стабилизацию, навигацию и управление движением БПЛА. В проекте – синоним полетного контроллера (Pixhawk/Cube).
UART	Universal Asynchronous Receiver-Transmitter	Универсальный асинхронный приёмопередатчик – узел, обеспечивающий связь в асинхронном последовательном порту. Основной интерфейс связи между RPi и полетным контроллером.
SSH	Secure Shell	Сетевой протокол для безопасного удаленного управления операционной системой. Используется для доступа к бортовому вычислителю RPi.