

??????????

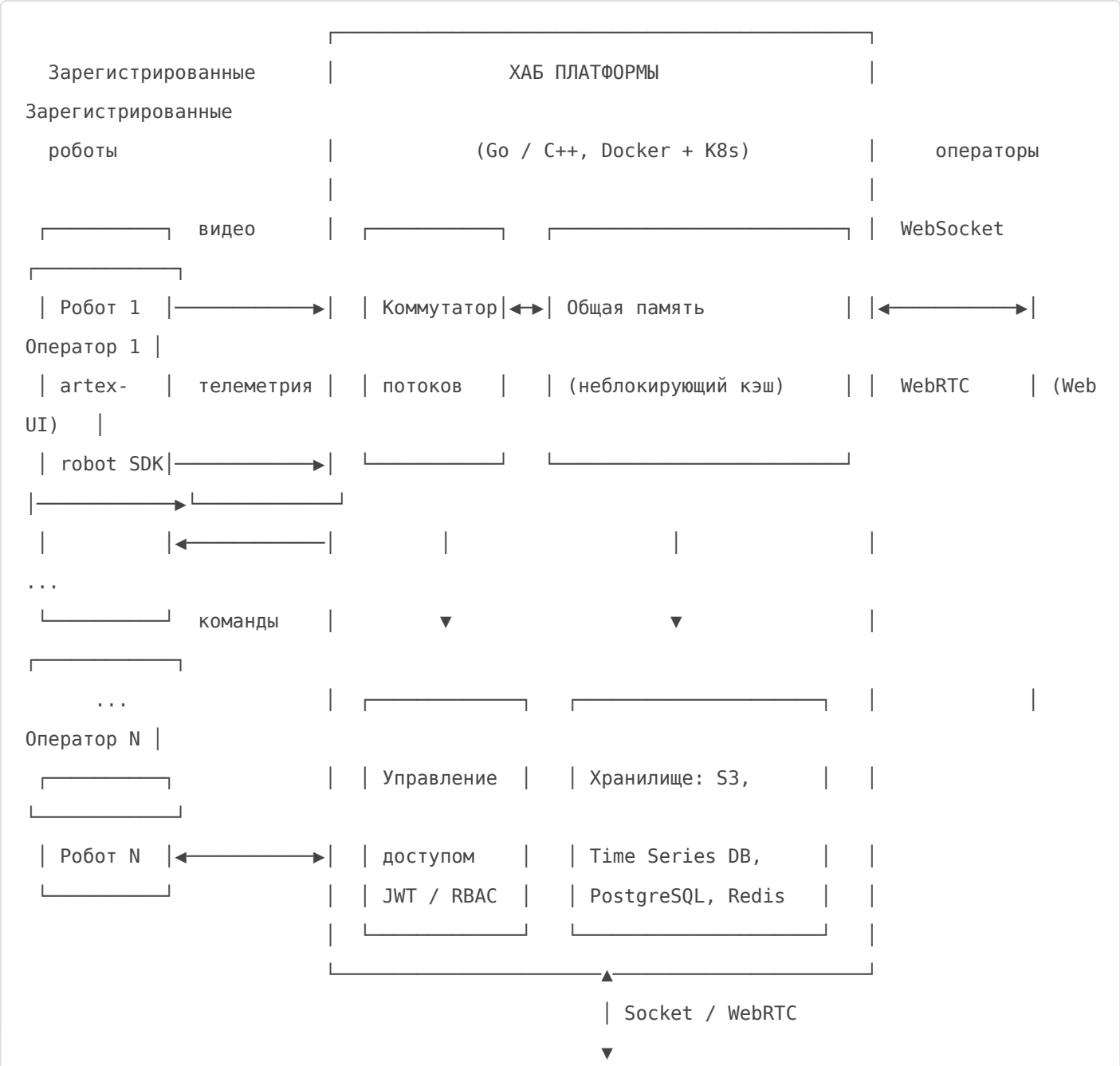
Компоненты, протоколы, серверные модули и хранение данных

- [Общая архитектура](#)
- [Протоколы и потоки данных](#)
- [Серверные модули](#)
- [Хранение данных](#)

????? ????????????

Платформа ARTEX построена по схеме «звезда»: в центре находится **ХАБ платформы**, к которому независимо подключаются роботы, операторы и модели ИИ. Никто из участников не общается напрямую — все потоки проходят через ХАБ, который коммутирует их, кэширует актуальное состояние и сохраняет данные.

????? ??????



Распределённая вычислительная среда
(Docker, K8s, GPU)
Модели ИИ оператора 1 ... оператора N
YOLO · Assist AI · RL Agent · Custom

??? ?????? ???????

Между участниками передаются три типа данных. Каждый поток идёт по своему каналу, чтобы тяжёлое видео не задерживало команды управления.

№	Поток	Направление	Содержание
1	Телеметрия, признаки	Робот → ХАБ → оператор и модели ИИ	Скорость, координаты, данные гироскопов и акселерометров, заряд АКБ, состояние дополнительных агрегатов, признаки (векторы), выделенные бортовыми моделями ИИ
2	Управление	Оператор / модель ИИ → ХАБ → робот	Управление движением, поворотом камеры, дополнительными аксессуарами. Основной источник — модель ИИ; оператор управляет при обучении или «подстраховке» модели
3	Видео	Робот → ХАБ → оператор и модели ИИ	Видео с камер, а также данные радаров, лидаров и т.п., преобразованные в видео. Оператору — для управления от первого лица и контроля работы ИИ; моделям — для анализа

????????????? ????????????

Компонент	Технологии	Ответственность
-----------	------------	-----------------

Робот	Python, SDK <code>artex-robot</code> , OpenCV	Съёмка видео и передача на сервер; сбор и передача телеметрии; работа бортовых моделей ИИ (обработка видео и данных датчиков); получение и применение команд управления
Платформа (ХАБ)	Go, C++	Идентификация роботов; приём видео и телеметрии, загрузка в кэш и хранилище; неблокирующий кэш с методами <code>push</code> и <code>get</code> ; коммутация роботов, операторов и моделей; подключение операторов и моделей к видео и данным
Оператор (Web-приложение)	React, WebSocket, WebRTC	Приём видео и телеметрии от робота; управление роботом и передача управления на платформу; мониторинг работы моделей ИИ, их входных и выходных данных и логов; адаптивный интерфейс для ноутбука и телефона
Вычислительная среда	Docker, Kubernetes, GPU	Обработка видео (YOLO); обработка временных рядов, RL; формирование команд управления
Хранилище	S3 / MinIO, Time Series DB, PostgreSQL, Redis	Сохранение видео, телеметрии, команд и логов; развёртывание, подключение и тестирование хранилища
Управление платформой (WEB-портал)	Web	Мониторинг подключённых роботов и операторов; ограничение «видимости» роботов для операторов; настройка сохранения данных

???? ?????????????????????????????????

Класс устройств	Типичные контроллеры	Способ подключения
RC-танки, машинки	ESP32 / STM32 + бортовой компьютер (Raspberry Pi 4)	SDK <code>artex-robot</code>
Наземные роботы	ROS2 / собственное ПО	SDK <code>artex-robot</code> (узел-мост)
Дроны (БПЛА)	PX4 / ArduPilot + компаньон-компьютер	SDK <code>artex-robot</code>
IoT-датчики	MQTT / GPIO	MQTT-брокер (планируется)

Класс устройств	Типичные контроллеры	Способ подключения
Светодиодные устройства (LED, RGB-лента)	Любой Python-контроллер	SDK <code>artex-robot</code> (тип <code>LED</code>)
Виртуальные роботы	Unity / Gazebo	SDK <code>artex-robot</code> — тот же протокол, что и у реальных

????????? ??????????????

- 1. **Всё через ХАБ.** Роботы, операторы и модели не знают адресов друг друга — это упрощает подключение, контроль доступа и запись данных.
- 2. **Неблокирующая общая память.** ХАБ хранит только *актуальный* кадр и *актуальный* вектор телеметрии (с отметкой времени). Медленный потребитель не тормозит быстрого: каждый читает последнее значение методом `get`.
- 3. **Модель и человек равноправны.** Команды от оператора и от модели ИИ имеют одинаковый формат; различается только автор и приоритет.
- 4. **Модульность.** Серверная логика собирается из модулей в визуальном редакторе с горячей перезагрузкой (hot reload). Каждый сервис — отдельный Docker-контейнер.
- 5. **Всё записывается.** Видео, телеметрия и команды (включая автора команды) сохраняются для анализа и обучения.

Подробнее: [Протоколы и потоки данных](#), [Серверные модули](#), [Хранение данных](#).

Вопросы и ответы

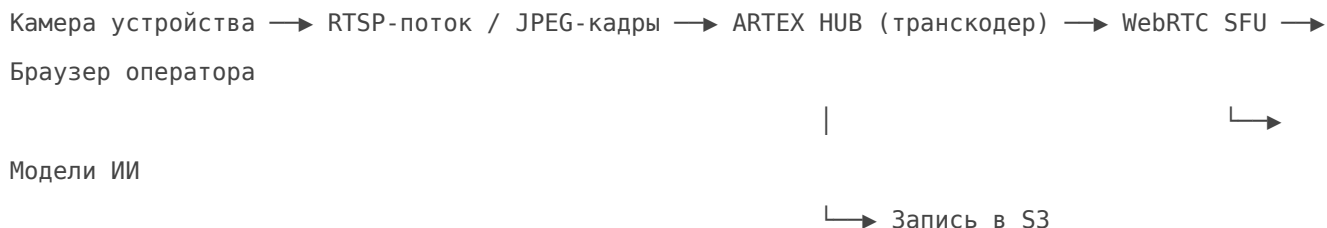
На странице описано, по каким протоколам участники обмениваются данными с ХАБом платформы и как движется каждый поток.

Таблица протоколов

№	Связь	Поток	Протокол
1	Робот ↔ ХАБ	Телеметрия, признаки	Socket (в текущей версии SDK — Socket.IO, namespace <code>/robot</code>)
2	Робот ↔ ХАБ	Управление	Socket (Socket.IO, событие <code>command</code>)
3	Робот → ХАБ	Видео	RTSP / WebRTC или «кадры» через Socket (в текущей версии — JPEG-кадры через Socket.IO)
4	Внутри ХАБа	Все потоки	Общая память — неблокирующий кэш (словарь)
5	ХАБ ↔ оператор	Контроль и обучение робота	WebSocket (данные и команды), WebRTC (видео)
6	ХАБ ↔ модели ИИ (вычислительная среда)	Телеметрия, управление, видео	Socket (телеметрия и управление), WebRTC / Socket (видео)
7	Модули сервера между собой	Служебные вызовы	REST API + gRPC
8	IoT-датчики → ХАБ	Показания датчиков	MQTT (publish/subscribe, QoS 1) — планируется
9	Администратор → сервер	Обслуживание	SSH

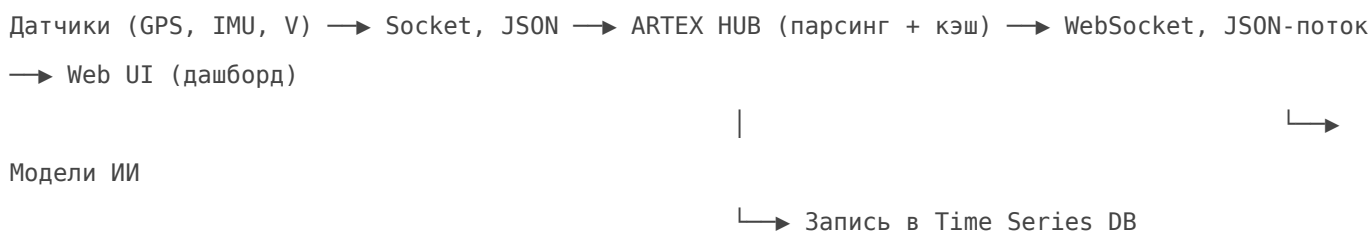
Протокол робота подробно описан в книге «ПО для робота» на странице [Протокол обмена с сервером](#).

Вопросы 1. Ответы



- Робот передаёт видео либо потоком RTSP, либо отдельными кадрами (JPEG, качество 80) через Socket.
- ХАБ хранит в кэше **актуальный кадр** (или набор из X последних кадров) и направляет его в вычислительную среду и оператору.
- Для браузера видео преобразуется в WebRTC; SFU ретранслирует поток нескольким получателям без повторного кодирования.
- Параллельно видео записывается в объектное хранилище S3.

????? 2. ???????????



- Телеметрия передаётся в формате JSON, например: `{"speed": 1.2, "gps": [55.65, 37.66], "battery": 11.8, "imu": {...}, "timestamp": 1790155251.12}`.
- Состав полей не фиксирован: робот может добавить новый датчик без изменения сервера и интерфейса.
- ХАБ хранит **актуальный вектор** телеметрии с учётом времени создания и раздаёт его потребителям.

????? 3. ???????????



- Команда содержит автора (`author`) и полезную нагрузку вида `{тип_команды: команда}`.
- ХАБ проверяет права автора на работа и применяет правило приоритета: команда оператора имеет приоритет над командой модели (режим «подстраховки»).

- Команды `Стоп` и `DISARM` обрабатываются вне очереди.

???????? ?????????????? ??????????

Параметр	Целевое значение
Камера → оператор	< 200 мс
Команда → привод	< 50 мс
Частота телеметрии	до 30 сообщений/с (JSON)
Частота отправки данных SDK по умолчанию	20 сообщений/с (<code>max_send_freq = 0.05</code> с)

????????????? ?????? ?????????????????? ?????????

1. Робот открывает соединение с сервером и передаёт паспорт устройства (`name`, `type`, `telem_can`, `video_can`).
2. Сервер присваивает подключению идентификатор (SID) и регистрирует робота в реестре устройств.
3. Сервер периодически отправляет **heartbeat**: запрашивает динамическое состояние (`d_meta`) и сигнал готовности (`Ready`).
4. Получив `Ready`, робот запускает циклы отправки телеметрии и видео.
5. Сервер пересылает роботу команды от операторов и моделей.
6. При обрыве связи SDK автоматически переподключается (до 5 попыток с задержкой 1-5 с); сервер помечает робота как отключённого и уведомляет интерфейс (порт `robot.disconnected`).

???????? ?????????? ?????????? ??????

ХАБ предоставляет операторам и моделям два базовых метода работы с общей памятью:

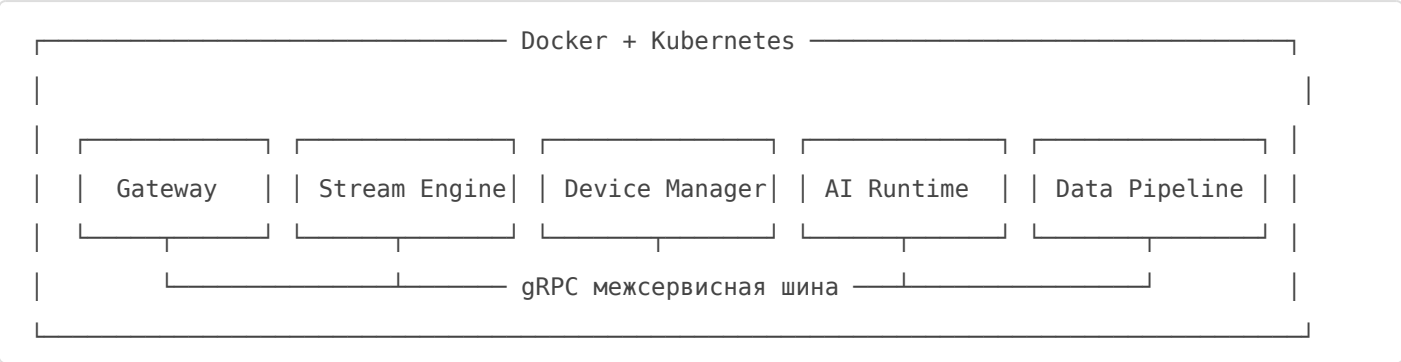
Метод	Назначение
<code>push(key, value)</code>	Записать новое значение (кадр, вектор телеметрии, команду) с отметкой времени. Не блокирует читателей.
<code>get(key)</code>	Получить последнее актуальное значение. Не ждёт появления нового значения.

Рекомендуемая схема именования ключей — `<робот>.<поток>[.<канал>]`, например `robot_1.video.rgb`, `robot_1.telemetry`, `robot_1.command`.

????????? ???????

Серверная часть ARTEX состоит из независимых сервисов на Go и C++. Каждый сервис упакован в Docker-контейнер, оркестрация и масштабирование выполняются Kubernetes. Сервисы взаимодействуют через общую шину gRPC, внешние клиенты — через REST API и WebSocket.

????????? ???????????

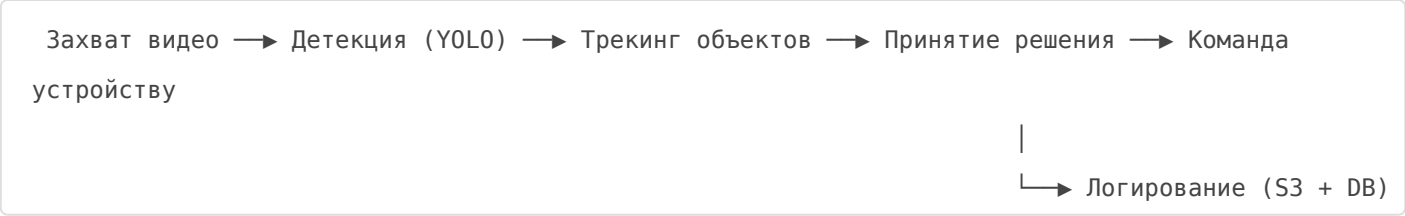


Сервис	Назначение	Основные функции
Gateway (API Gateway)	Единая точка входа для внешних клиентов	REST API и WebSocket; JWT-авторизация; ограничение частоты запросов (rate limiting)
Stream Engine	Коммутатор видео	Приём RTSP и кадров, преобразование RTSP → WebRTC, мультиплексирование видео и данных, транскодирование
Device Manager	Управление устройствами	Реестр устройств, подключение устройств, передача команд, контроль статуса и heartbeat
AI Runtime	Среда исполнения моделей	Загрузка моделей (YOLO / RL / Custom), распределение GPU, ведение входов, выходов и логов моделей
Data Pipeline	Запись данных	Сохранение в S3, Time Series DB, PostgreSQL; горячий кэш Redis

??????????? ??????????? (pipeline)

Прикладная логика обработки данных собирается из **модулей** в визуальном редакторе. Каждый модуль отвечает за одну задачу и имеет именованные порты — входы и выходы. Пользователь перетаскивает модули на рабочее поле и соединяет порты, формируя конвейер. Изменения применяются без перезапуска сервера (hot reload).

Типовой конвейер автономного управления:



???????????????? ???? ???? ?

Модуль	Назначение	Порты
ClientIOModule	Хостит веб-интерфейс управления устройствами	robot.connected, robot.disconnected, robot.telemetry, robot.video
ConsoleLogModule	Вывод всех проходящих данных в консоль сервера для отладки	—
RobotWSModule	Мост между физическим устройством и платформой (WebSocket)	robot.command (выход к устройству), robot.telemetry, robot.video, robot.connected, robot.disconnected
RobotsModule	Источник данных от группы роботов (используется аналогично RobotWSModule)	robot.telemetry, robot.video и др.
YoloModule	Детекция объектов на видеопотоке в реальном времени	robot.video (вход), yolo.detect (выход)
VideoRecordModule	Запись видеопотока в файл по командам video_start_record / video_stop_record	robot.video, команды

Подробное описание модулей и их использования в интерфейсе — на странице «Веб-интерфейс управления роботами» книги «Управление роботами с помощью ИИ», примеры конвейеров — на странице [Конвейеры модулей](#).

???????? ???? ???? ?

- Выход можно соединить с несколькими входами — данные будут доставлены всем получателям.
- Порты соединяются по типу данных: robot.video ↔ robot.video, robot.telemetry ↔ robot.telemetry.
- ConsoleLogModule можно подключить к любому порту для отладки.

- Для записи данных используйте модули записи (VideoRecordModule) или сервис Data Pipeline.

???????? ?????????? ??????????

Интерфейс	Клиенты	Назначение
Socket.IO (namespace <code>/robot</code>)	Роботы (SDK)	Регистрация, телеметрия, видео, команды
WebSocket	Web UI оператора, модели ИИ	Данные в реальном времени и команды
WebRTC	Браузер оператора, модели ИИ	Видео с малой задержкой
REST API	Web UI, портал управления, внешние системы	Пользователи, устройства, сессии, настройки
gRPC	Внутренние сервисы	Межсервисное взаимодействие
SSH	Администраторы	Обслуживание серверов

???????? ???? ?

Платформа сохраняет все данные, проходящие через ХАБ: они нужны для разбора соревнований, отладки моделей и обучения новых моделей ИИ. Для каждого типа данных используется подходящее хранилище.

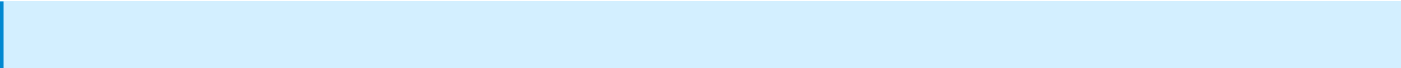
???????? ?

Хранилище	Что хранится	Особенности
S3 / MinIO	Видео, датасеты, снапшоты (версии) моделей ИИ, лог-файлы	Объектное хранилище; разделение по бакетам
Time Series DB	Телеметрия: GPS, скорость, IMU, заряд батареи, показания датчиков	Хранение и выборка по времени, агрегация
PostgreSQL	Пользователи, реестр устройств, сессии, права доступа, настройки	Реляционные данные
Redis	Горячий кэш, состояние в реальном времени	In-memory; актуальные значения для ХАБа
Файловая система	Локальные логи и временные файлы	Используется на этапе демо-прототипа

??? ?????????? ? ??? ??????

На каждом сеансе управления роботом ведётся лог данных:

Данные	Хранилище	Содержание записи
Видео	S3	Кадры/фрагменты видео с отметкой времени и идентификатором робота и камеры
Телеметрия	Time Series DB	Вектор телеметрии с временем создания на роботе и временем приёма на сервере
Команды оператора	Time Series DB / S3	Команда, автор, время, робот
Команды моделей ИИ	Time Series DB / S3	Команда, модель, версия модели, время, робот
Выходы моделей	S3	Результаты детекции, логи модели



Благодаря синхронной записи «кадр — телеметрия — команда оператора» каждая сессия ручного управления превращается в готовый датасет для Imitation Learning. Подробнее — на странице [Сбор данных и обучение моделей](#).

???????????? ???? ?

???????????? ???? ?

```
s3://artex-video/<robot_sid>/<session_id>/<camera>/<timestamp>.jpg  
s3://artex-datasets/<dataset_name>/<version>/...  
s3://artex-models/<model_name>/<version>/weights.*  
s3://artex-logs/<service>/<date>/...
```

???????????? ???? ?

Параметры сохранения задаются администратором в портале управления платформой:

- какие потоки записывать (видео, телеметрия, команды) — для всех роботов или выборочно;
- частота прореживания видео при записи (например, каждый N-й кадр);
- срок хранения данных;
- включение записи видео по команде оператора (`video_start_record` / `video_stop_record`).

???? ????-????????

На этапе демо-прототипа логи сохраняются в файлы; развёртывание, подключение и тестирование S3 и Time Series DB выполняются на этапе прототипа «Альфа».