Руководство по эксплуатации Программы для ЭВМ «Цифровая сенсорная платформа»

город Москва, 2026

Содержание

Перечень принятых сокращений

В настоящем документе используются следующие сокращения и соответствующие им определения.

Термин/ сокращение	Определение/расшифровка
CV	Computer Vision (компьютерное зрение)
Docker	Платформа контейнеризации для развёртывания приложений в изолированных средах
docker-compose	Инструмент для описания и запуска многоконтейнерных Docker-приложений
FPS	Frames Per Second — количество кадров в секунду
OPC UA	Open Platform Communications Unified Architecture — протокол промышленного взаимодействия
RTSP	Real Time Streaming Protocol — протокол потоковой передачи видеоданных в реальном времени
SDK	Software Development Kit — набор средств разработки для взаимодействия с оборудованием
YAML	Текстовый формат описания конфигурационных данных

Термин/ сокращение	Определение/расшифровка
ZMQ	ZeroMQ — библиотека обмена сообщениями на базе сокетов
Modbus TCP	Промышленный протокол связи для обмена данными с контроллерами по сети Ethernet
АСУТП	Автоматическая система управления технологическим процессом
БД	База данных
КИП	Контрольно-измерительные приборы
ОС	Операционная система
ПЛК	Программируемый логический контроллер
ПО	Программное обеспечение
Программа	Цифровая сенсорная платформа
СУБД	Система управления базами данных
Тег	Именованная переменная в системе АСУТП или в структуре OPC UA, хранящая значение измеряемого или расчётного показателя

1. Введение

Настоящий документ является руководством по эксплуатации программы для ЭВМ «Цифровая сенсорная платформа» (далее — «ПО», «Программа»).

Программа представляет собой набор микросервисов и библиотеку, обеспечивающих цепочку сбора, передачи, обработки и хранения данных из различных промышленных источников. Программа не имеет графического пользовательского интерфейса. Интерфейсом взаимодействия с Программой являются конфигурационные файлы сервисов, переменные окружения и программный интерфейс библиотеки.

Программа разворачивается на базе операционной системы Linux. Каждый сервис поставляется в виде Docker-контейнера. Развёртывание и оркестрация сервисов осуществляется при помощи docker-compose. В качестве СУБД используется PostgreSQL, в качестве шины данных — Apache Kafka.

2. Описание интерфейса

2.1 Общие сведения

Все сервисы Программы поставляются в виде Docker-образов и развёртываются при помощи файлов docker-compose.yaml. Конфигурирование каждого сервиса осуществляется посредством:

– **Конфигурационных файлов** в формате YAML, монтируемых в контейнер;

– **Переменных окружения**, задаваемых в файлах .env или непосредственно в docker-compose.yaml.

Типовой порядок развёртывания сервиса:

1. Подготовить конфигурационный файл сервиса на основе шаблона;

2. Задать параметры подключения к БД, Kafka, OPC UA и другим зависимостям;

3. Описать сервис в файле docker-compose.yaml;

4. Запустить сервис командой docker compose up -d <имя_сервиса>.

Подключение новых устройств и источников данных не требует остановки работающих сервисов — достаточно развернуть дополнительные экземпляры соответствующих сервисов с индивидуальной конфигурацией.

2.2. Подсистема сбора и обработки видеоданных

2.2.1. Видеошлюз (video-gateway)

Видеошлюз инкапсулирует взаимодействие с камерой и обеспечивает два выходных канала: передачу кадров через ZMQ-сокет и отправку видеопотока на внешний видеосервер.

Конфигурация осуществляется через переменные окружения:

Переменная	Описание
DRIVER_TYPE	Тип драйвера камеры: rtsp, daheng, mock и др.
CAMERA_URL	Адрес камеры (для RTSP — URL потока)
FPS	Частота захвата кадров (кадров/с)
ZMQ_PUB_ADDR	Адрес ZMQ-сокета для публикации кадров (например, tcp://0.0.0.0:5555)
STREAM_URL_OUT	Адрес видеосервера для отправки RTSP-потока
ENABLE_STREAM	Включение/отключение отправки видеопотока на видеосервер

Модульная архитектура драйверов. Видеошлюз поддерживает подключение модулей драйверов для различных типов камер. Каждый драйвер реализует единый интерфейс для захвата кадров. Доступные драйверы:

- **rtsp** — для IP-камер, предоставляющих видеопоток по протоколу RTSP. Выполняет декодирование потока и нарезку на отдельные кадры;
- **daheng** — для специализированных камер компьютерного зрения Daheng с интеграцией через SDK производителя;
- **mock** — тестовый драйвер для разработки и отладки без физической камеры, например, из записанных файлов.

Добавление поддержки нового типа камеры осуществляется путём реализации модуля драйвера, наследующего базовый интерфейс.

Выходные каналы:

- **ZMQ PUB** — публикация сжатых кадров с метаданными (временная метка захвата, последовательный номер) по паттерну

«издатель — подписчик». Сервисы компьютерного зрения подключаются к этому сокету для получения кадров;

➢ **RTSP-поток** — энкодинг и упаковка кадров в видеопоток с отправкой на заданный эндпойнт внешнего видеосервера для организации трансляций.

2.2.2. Опрос регистров ПЛК (modbus-poll)

Сервис предназначен для циклического опроса регистров программируемого логического контроллера по протоколу Modbus TCP и записи полученных значений в базу данных.

Конфигурация осуществляется через файл config.yaml:

```yaml
modbus:
  host: 192.168.1.100      # IP-адрес ПЛК
  port: 502                # Порт Modbus TCP
  timeout: 10              # Таймаут соединения (с)

polling:
  interval_sec: 1.0        # Интервал опроса (с)

tags:
  temperature_1:
    description: "Температура датчика 1 (°C)"
    address: 100           # Адрес регистра
    data_type: FLOAT32     # Тип данных
    word_order: big        # Порядок байтов
    agg_op: mean           # Функция агрегации
```

Для каждого тега задаются: адрес регистра, тип данных, порядок байтов и функция агрегации. Результаты записываются в общую базу данных подсистемы видеоданных.

2.2.3. Библиотека platform-sdk: получение кадров (FrameReceiver)

Класс FrameReceiver предоставляет высокоуровневый интерфейс для получения кадров от видеошлюза через ZMQ-сокет. Предназначен для использования в прикладных сервисах компьютерного зрения.

Подключение:

```
from platform_sdk.connectors.sources.frame import FrameReceiver, ReceiveMode

receiver = FrameReceiver(
    address="tcp://video-gateway:5555",
    mode=ReceiveMode.LATEST
)
receiver.connect()
frames = receiver.get_data()
```

Стратегии получения кадров (ReceiveMode):

Режим	Описание
LATEST	Возвращает наиболее свежий доступный кадр без буферизации. Промежуточные кадры, не успевшие быть обработанными, отбрасываются
BATCH	Набирает пакет из заданного количества последовательных свежих кадров и возвращает их единовременно
SEQUENTIAL	Последовательная обработка кадров с буферизацией и механизмом принудительного сброса буфера при необходимости

Каждый полученный кадр содержит метаданные: исходную временную метку захвата и последовательный номер кадра.

2.2.4. Библиотека platform-sdk: запись результатов (MeasurementWriter)

Класс MeasurementWriter предоставляет унифицированный интерфейс для записи результатов расчётов сервисов компьютерного зрения в базу данных. Инкапсулирует логику работы с БД, включая жизненный цикл данных, автоматическую очистку устаревших записей и архивацию для воспроизводимости.

Подключение:

```
from platform_sdk.connectors.sinks.measurement import MeasurementWriter

writer = MeasurementWriter(
    host="localhost",
    port=5432,
```

```
    database="platform"
)
writer.connect()
```

Конфигурация через переменные окружения или аргументы конструктора:

Параметр	Переменная окружения	По умолчанию	Описание
host	POSTGRES_HOST	localhost	Адрес PostgreSQL
port	POSTGRES_PORT	5432	Порт PostgreSQL
database	POSTGRES_DB	platform	Имя базы данных
retention_interval	DATA_RETENTION_HOURS	72	Время хранения данных тегов (ч)
calls_retention_interval	CALLS_RETENTION_HOURS	72	Время хранения архивов вызовов (ч)
check_interval	RESULTS_CHECK_INTERVAL	0	Интервал проверки результатов (с)

Основные возможности:

- Запись результатов расчётов в виде тегов с контролем времени хранения и автоматической очисткой;
- Автоматическая архивация входных данных (включая кадры) и результатов с привязкой к временной метке вычисления и идентификатору запуска сервиса;
- Архивация вызывается регулярно по настраиваемому интервалу или принудительно из пользовательского кода;
- Управление схемами таблиц в БД

2.2.5. Синхронизация архивов (validation-sync)

Сервис выполняет регулярную отправку архивированных данных о работе сервисов компьютерного зрения (входные данные и результаты расчётов) из локальной базы данных в удалённую для обеспечения централизованного хранения и валидации.

Конфигурация через переменные окружения:

Переменная	**Описание**
SOURCE_POSTGRES_HOST, SOURCE_POSTGRES_PORT, SOURCE_POSTGRES_DB	Параметры подключения к локальной БД (источник)
TARGET_POSTGRES_HOST, TARGET_POSTGRES_PORT, TARGET_POSTGRES_DB	Параметры подключения к удалённой БД (приёмник)
SYNC_INTERVAL_MINUTES	Интервал синхронизации (мин), по умолчанию — 1

2.2.6. Синхронизация архивов (validation-sync)

Сервис считывает значения тегов из базы данных PostgreSQL, выполняет их агрегацию и записывает в узлы OPC UA сервера с настраиваемой периодичностью.

Конфигурация через файл config.yaml:

```yaml
opc:
  url: "opc.tcp://opc-server:4840/opcua/server/"
  namespace: "forkit-opcua"

postgres:
  host: "localhost"
  port: 5432
  db: "platform"

poll_interval: 5  # Интервал обновления OPC (с)
```

Сервис производит агрегацию показателей и единовременно обновляет значения в OPC UA с заданным интервалом, что позволяет избежать привязки частоты обновления к FPS сервисов компьютерного зрения. Конфигурации тегов для записи формируются автоматически на основе данных, зарегистрированных сервисами CV через MeasurementWriter.

2.2.7. OPC UA сервер (opc-server)

Сервис предоставляет локальный OPC UA сервер для целей разработки и тестирования. Включает веб-сервер (HTTP API) для динамического добавления узлов в OPC UA структуру без перезапуска сервера.

Допустимо использование внешнего OPC UA сервера, предоставленного специалистами АСУТП. В этом случае данный сервис не разворачивается.

Конфигурация через переменные окружения:

Переменная	**Описание**
OPC_URL	Адрес OPC UA эндпойнта (по умолчанию: opc.tcp://0.0.0.0:4840/opcua/server/)
OPC_NAMESPACE	Пространство имён OPC UA
OPC_HTTP_HOST, OPC_HTTP_PORT	Адрес и порт HTTP API для динамического управления узлами
DYNAMIC_MODE	Включение динамического режима добавления узлов (по умолчанию: true)
PRELOAD_STRUCTURE	Предзагрузка структуры узлов из конфигурационного файла structure.yaml

2.2.8. OPC UA сервер (opc-server)

Для организации видеотрансляций используется внешний видеосервер, поддерживающий приём RTSP-видеопотоков. При локальных развёртываниях рекомендуется использование видеосервера go2rtc.

Конфигурация go2rtc осуществляется через файл go2rtc.yaml:

```
streams:
  fm9_1: []          # Имя потока; источник назначается видеошлюзом
webrtc:
  filters:
    udp_ports: [50000, 50100]  # Диапазон UDP-портов для WebRTC
```

Видеосервер принимает RTSP-поток от видеошлюза и предоставляет его в различных форматах (MSE, WebRTC) для просмотра через браузер или встраивания в дашборды.

2.3. Подсистема сбора и обработки данных АСУТП

2.3.1. Шлюз OPC UA — Kafka (opc-to-kafka)

Сервис выполняет циклический опрос OPC UA сервера по заданным тегам и формирует сообщения в шину данных Kafka.

Конфигурация через файл config.yaml:

```
app:
  cycle_msec: 5000          # Интервал опроса OPC (мс)
  check_tag_update: 30      # Интервал проверки обновления списка тегов (с)

kafka:
  ip: kafka-ml
  port: 9092
  topic_data: opc_in        # Топик для данных
  topic_tags: opc_in_tags   # Топик для метаданных тегов

opc_client:
  server_url: opc.tcp://<ip>:<port>
  tags:
    - path: Objects/WinCC/PLC1/tag1
    - path: Objects/WinCC/PLC1/tag2
```

Для каждого тега указывается полный путь в структуре OPC UA. Сервис поддерживает аутентификацию Kafka (SASL).

2.3.2. Ретрансляция Kafka — Kafka (kafka-to-kafka)

Сервис предназначен для передачи сообщений между экземплярами Kafka в распределённой среде. Применяется в схемах, где Edge-устройство передаёт данные на центральный сервер.

Конфигурация через файл config.yaml:

```yaml
consumer:
  bootstrap_servers:
    - <edge_ip>:<port>
  group_id: group_edge

producer:
  bootstrap_servers:
    - <server_ip>:<port>

topics:
  - opc_in
  - opc_in_tags
prefix: out_          # Префикс для целевых топиков
```

Сервис ретранслирует указанные топики из Kafka-источника в Kafka-приёмник, добавляя к именам целевых топиков заданный префикс. Механизм retention Kafka на Edge-устройстве обеспечивает буферизацию данных и устойчивость к разрывам сетевого соединения.

2.3.3. Запись в БД (kafka-to-db)

Сервис потребляет сообщения из Kafka, формирует из них временные ряды, выполняет агрегацию (resample) к круглым минутам и записывает результаты в базу данных PostgreSQL.

Конфигурация через файл config.yaml:

```yaml
consumer:
  bootstrap_servers:
    - <ip>:<port>
  topic_data: opc_in_m1
  topic_tags: opc_in_tags

database:
  dbname: '<dbname>'
  host: '<ip>'
  port: '<port>'

asset:
  id: 1
  name: 'Участок'
```

Сервис автоматически создаёт необходимые схемы и таблицы в базе данных при их отсутствии. Параметр asset задаёт

идентификатор и наименование производственного объекта для привязки данных.

2.3.4. Расчёт статистик (aggregate-hub)

Сервис выполняет расчёт статистических агрегатов на основе временных рядов, хранящихся в базе данных, с записью результатов в целевую таблицу.

Конфигурация через файл config.yaml:

```yaml
database:
  dbname: <dbname>
  host: <host>
  port: <port>

aggregate:
  source_table: 'tag_data_m1'     # Исходная таблица
  target_table: 'tag_stats'       # Целевая таблица
  tags: [2, 17]                   # Идентификаторы тегов
  timeframes:
    3600: '1h'                    # Окна агрегации (с): метка
  refresh_period: 3               # Период обновления (мин)
  history_horizon_days: 1         # Глубина пересчёта (дней)
```

Поддерживается расчёт агрегатов по нескольким временным окнам одновременно. Сервис выполняет обновление с настраиваемой периодичностью и поддерживает пересчёт по историческим данным.