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

Обзор SDK artex-robot, установка и первый запуск

- Обзор SDK artex-robot
- Быстрый старт
- Установка базового ПО на робота для управления через платформу
- Конфигурация

????? SDK artex-robot

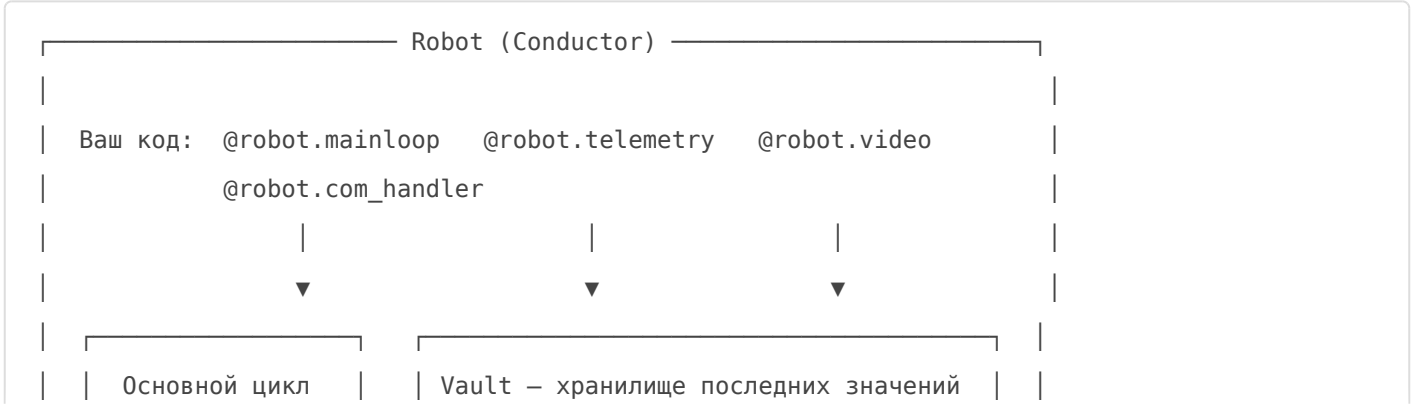
artex-robot — Python-библиотека (SDK), которая подключает устройство к платформе ARTEX. SDK берёт на себя соединение с сервером, регистрацию устройства, отправку видео и телеметрии и приём команд. Разработчику остаётся описать три вещи: **как читать датчики**, **как получать кадры с камеры** и **как выполнять команды**.

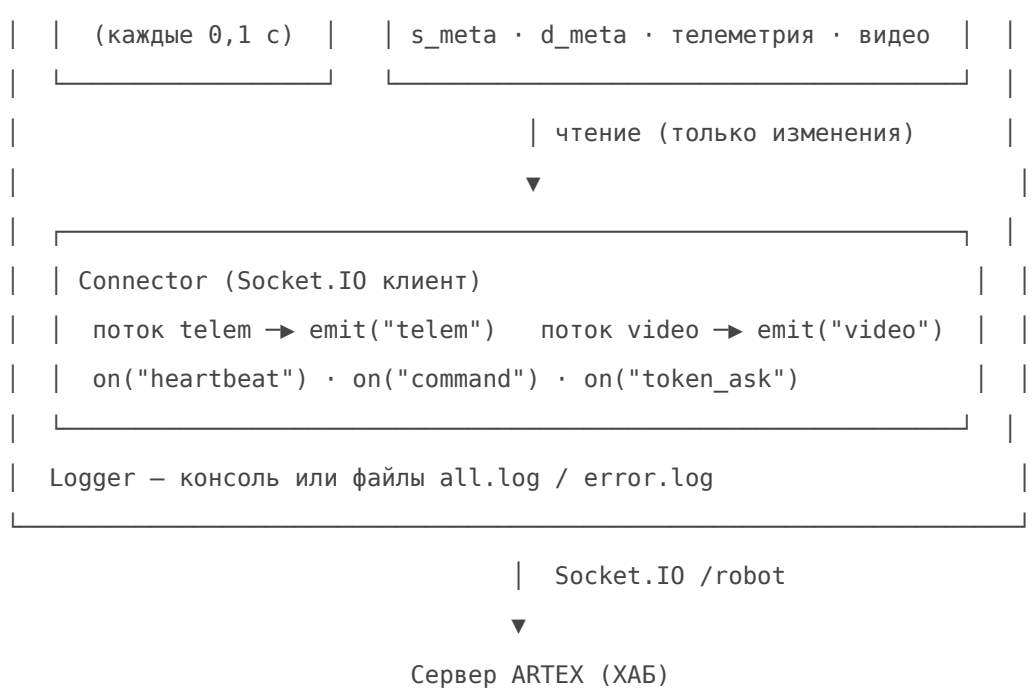
Параметр	Значение
Пакет	`artex-robot` (PyPI)
Импорт	`from artex_robot import Robot`
Текущая версия	0.1.1
Python	3.8+
Зависимости	`python-socketio` ≥ 5.10 (устанавливается автоматически); `opencv-python` и `numpy` — для видео (устанавливаются вручную)
Протокол	Socket.IO (WebSocket), namespace `/robot`
Разработчики	ИЦИИ МИФИ

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

SDK работает на любом устройстве, где запускается Python 3: Raspberry Pi, Jetson, мини-ПК, ноутбук. Микроконтроллеры (ESP32, STM32, Arduino) подключаются через бортовой компьютер — SDK работает на компьютере и передаёт команды контроллеру по UART / I2C / USB. Виртуальные роботы в симуляторе подключаются тем же SDK.

??? ??????? SDK





Компонент	Назначение
Robot (Conductor)	Главный объект. Читает конфигурацию, хранит контекст, запускает основной цикл, регистрирует обработчики через декораторы
Vault	Хранит паспорт устройства (`s_meta`), состояние (`d_meta`), последние значения телеметрии и последние кадры. Кадры сразу кодируются в JPEG
Connector	Клиент Socket.IO: подключение, авто-переподключение, обработка heartbeat и команд, фоновые потоки отправки телеметрии и видео
Logger	Журнал событий: в консоль или в файлы `all.log` / `error.log`

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

1. Вы создаёте объект `Robot` с конфигурацией (имя, тип, адрес сервера).
2. Декоратором `@robot.telemetry("ключ")` помечаете функции, которые читают датчики; декоратором `@robot.video("камера")` — функции, которые возвращают кадр.
3. В функции `@robot.mainloop` вызываете эти функции — их результаты автоматически попадают в Vault.
4. `robot.init_connector()` подключается к серверу; `robot.start()` запускает бесконечный основной цикл.
5. Получив от сервера сигнал готовности, SDK в фоне отправляет на сервер всё новое из Vault.
6. Команды от оператора или модели ИИ вызывают вашу функцию `@robot.com_handler`.

>

> Отправка данных и основной цикл независимы: основной цикл только обновляет значения в Vault, а отдельные потоки отправляют на сервер последние значения со своей частотой. Если значение не менялось, повторно оно не отправляется.

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

- Быстрый старт — установка и первый запуск.
- Конфигурация — параметры `params.json`.
- Справочник API — все методы и декораторы.
- Обработка команд — типы команд и их обработка.
- Протокол обмена с сервером — события Socket.IO и форматы сообщений.
- Примеры роботов — LED, колёсный робот с камерой, виртуальный робот.
- Устранение неполадок.

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

За 10 минут подключим устройство к платформе ARTEX: оно появится в списке подключений, будет передавать телеметрию и видео и принимать команды из веб-интерфейса.

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

- Устройство с Python 3.8+ (Raspberry Pi, ноутбук, мини-ПК).
- Сетевой доступ от устройства к серверу ARTEX.
- USB- или CSI-камера (необязательно).

??? 1. ?????????

```
python3 -m venv ~/artex-env
source ~/artex-env/bin/activate

pip install artex-robot
pip install opencv-python numpy        # нужны для работы с видео
```

> В текущей версии SDK (0.1.1) модуль видео импортирует `cv2` при любом запуске, поэтому `opencv-python` нужен даже устройствам без камеры. На Raspberry Pi без графической оболочки используйте `opencv-python-headless`.

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

Создайте рядом со скриптом файл `params.json`:

```
{
  "name": "tank-01",
  "type": "Robot",
  "server": "http://artex.arcon",
  "port": 80,
  "logger_type": "console",
  "logger_level": "skip_TODO"
```

```
}
```

`server` и `port` — адрес сервера платформы (узнайте у администратора). Все параметры описаны на странице Конфигурация.

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

Создайте файл `robot.py`:

```
import cv2
from artex_robot import Robot

robot = Robot()                      # параметры берутся из params.json

# 1. Обработка команд от оператора и моделей ИИ
@robot.com_handler
def on_command(type_command, command, author=None):
    print(f"Команда {command} (тип {type_command}) от {author}")
    if command == "stop":
        pass                         # остановить двигатели

# 2. Телеметрия: значение, которое вернёт функция, уйдёт на сервер под ключом "battery"
@robot.telemetry("battery")
def read_battery():
    return 12.1                      # здесь чтение реального датчика

# 3. Видео: кадр, который вернёт функция, уйдёт на сервер как камера "rgb"
@robot.video("rgb")
def send_frame(frame):
    return frame                     # numpy-массив BGR, SDK сам сожмёт его в JPEG

# Камеру кладём в контекст робота, чтобы получить её в основном цикле
cam = cv2.VideoCapture(0)
robot["cam"] = cam

# 4. Основной цикл: вызывается каждые 0,1 с
@robot.mainloop
def main_loop(context):
    read_battery()
```

```
    ok, frame = context["cam"].read()
    if ok:                              # пустой кадр отправлять нельзя
        send_frame(frame)

robot.init_connector()                  # подключение к серверу
robot.start()                           # запуск основного цикла (Ctrl+C — остановка)
```

> Обработчик `@robot.com_handler` и основной цикл `@robot.mainloop` обязательны: без них SDK версии 0.1.1 завершится с ошибкой `AttributeError`. Если команды не нужны, объявите пустой обработчик. Функция основного цикла всегда принимает аргумент `context`.

> Функция с декоратором `@robot.video` не должна возвращать `None`: SDK пытается сжать результат в JPEG, и пустой кадр вызывает ошибку OpenCV, которая останавливает программу. Поэтому кадр читается и проверяется в основном цикле, а в декорированную функцию передаётся только успешно прочитанный кадр.

??? 4. ?????? ? ????????

```
python robot.py
```

В консоли появятся сообщения `[INFO] -- connected` и `[INFO] -- in heartbeat recieved [...]`. Откройте веб-интерфейс платформы: устройство `tank-01` появится на странице «Подключения». Откройте его — вы увидите видео и телеметрию `{"battery": 12.1}`. Отправьте команду `stop` из поля ввода команд — в консоли робота появится `Команда stop (тип com) от ...`.

??? 5. ?????????? (systemd)

```
# /etc/systemd/system/artex-robot.service
[Unit]
Description=ARTEX robot
After=network-online.target
Wants=network-online.target

[Service]
WorkingDirectory=/home/pi/robot
ExecStart=/home/pi/artex-env/bin/python robot.py
```

```
Restart=always
RestartSec=5

[Install]
WantedBy=multi-user.target
```

```
sudo systemctl enable --now artex-robot
journalctl -u artex-robot -f
```

`WorkingDirectory` обязателен: SDK ищет `params.json` в текущем рабочем каталоге. `Restart=always` перезапускает программу при сбое; чтобы перезапуск срабатывал и при долгой потере связи с сервером, добавьте в основной цикл сторожевую проверку — см. Устранение неполадок.

??? ??????

- Обработка команд — как превратить команды в движение.
- Примеры роботов — готовые шаблоны для LED, колёсного робота, виртуального робота.

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

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

1. Установка библиотек

```
pip install artex-robot
```

> если при запуске потребуются доп зависимости, их необходимо доустановить вручную

2. Развертывание ПО

```
from artex_robot import Robot

robot = Robot({"type":"LED"})
robot.log.set_level("skip_TODO")

@robot.com_handler
def custom_handler(type_command, command, author = None):
    print(f"Custom com_handler got {command} with type:{type_command} from author - {author}")

video = True
if video:
    import cv2
    cl_cam = cv2.VideoCapture(0)
    cl_cam.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter_fourcc(*'MJPG'))

    #robot.set_context("cl_cam", cl_cam)
    robot["cl_cam"] = cl_cam

@robot.telemetry("temp_sensor")
def read_temperature():
    return 1
```

```
@robot.video("rgb")
def read_frame(cam):
    _, frame = cam.read()
    if _ :
        return frame

@robot.mainloop
def main_loop(context):

    read_temperature()
    read_frame(context["cl_cam"])

robot.init_connector()
robot.start()
```

??????????

Параметры робота задаются двумя способами, которые можно комбинировать:

1. Файл `params.json` в **текущем рабочем каталоге** программы.
2. Словарь, переданный в конструктор: `Robot({...})`.

Значения из словаря имеют приоритет над `params.json`. Удобная схема: общие параметры (адрес сервера, логирование) — в `params.json`, а тип устройства — в коде.

```
robot = Robot({"type": "LED"})    # остальное — из params.json
```

?????????

Параметр	Тип	По умолчанию	Описание
`name`	str	`"DefaultName"`	Имя устройства, отображается в интерфейсе оператора
`type`	str	`"DefaultType"`	Тип устройства. Определяет страницу управления в интерфейсе: `Robot`, `LED`, `Drone`
`server`	str	`"http://artex.arcon"`	Адрес сервера платформы (схема + хост, без порта)
`port`	int	`80`	Порт сервера
`telem_can`	bool	`false`	Признак «устройство передаёт телеметрию» — передаётся серверу в паспорте
`video_can`	bool	`false`	Признак «устройство передаёт видео» — передаётся серверу в паспорте
`max_send_freq`	float	`0.05`	**Период** (в секундах) между отправками телеметрии и видео. `0.05` — до 20 отправок в секунду

Параметр	Тип	По умолчанию	Описание
`max_main_freq`	float	`0.1`	**Период** (в секундах) паузы основного цикла. `0.1` — около 10 итераций в секунду
`logger_type`	str	`"console"`	Куда писать журнал: `console` — в стандартный вывод, `file` — в файлы `all.log` и `error.log`
`logger_level`	str	`"all"`	Уровень журнала: `all` — все сообщения, `skip_TODO` — все, кроме служебных сообщений `TODO`
`external`	dict	`{}`	Зарезервировано для внешних модулей (в версии 0.1.1 не используется)

> Несмотря на слово `freq` в названии, параметры `max_send_freq` и `max_main_freq` задают период в секундах, а не частоту. Чтобы отправлять данные 10 раз в секунду, укажите `0.1`.

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

При подключении SDK передаёт серверу **паспорт** (`s_meta`) — параметры `name`, `type`, `telem_can`, `video_can`. Сервер использует его для отображения устройства и выбора интерфейса управления.

Состояние (`d_meta`) меняется во время работы и возвращается серверу по запросу heartbeat:

Поле	Значения	Смысл
`sid`	идентификатор или `null`	Идентификатор подключения
`ready`	`init` → `ok`	`ok` после сигнала готовности от сервера; до этого команды игнорируются
`telem_run`	bool	Запущен ли поток отправки телеметрии
`video_run`	bool	Запущен ли поток отправки видео

????? ??????

Сценарий	`max_main_freq`	`max_send_freq`
LED, простые датчики	0.1	0.1
Колёсный робот с камерой (FPV)	0.03–0.05	0.033–0.05
Слабый канал связи / мобильная сеть	0.1	0.1–0.2

Реальная частота отправки не выше частоты основного цикла: данные отправляются, только если основной цикл успел обновить их в Vault. Размер одного кадра 640×480 в JPEG (качество 80) — около 30–60 КБ; при 20 кадрах/с это 5–10 Мбит/с. Для слабых каналов уменьшайте разрешение кадра перед отправкой.

??????? `params.json`

Колёсный робот:

```
{
  "name": "tank-01",
  "type": "Robot",
  "server": "http://192.168.1.10",
  "port": 80,
  "telem_can": true,
  "video_can": true,
  "max_send_freq": 0.05,
  "max_main_freq": 0.05,
  "logger_type": "file",
  "logger_level": "skip_TODO"
}
```

Светодиодная лампа:

```
{
  "name": "lamp-hall",
  "type": "LED",
  "server": "http://192.168.1.10",
  "telem_can": true,
  "logger_level": "skip_TODO"
}
```