

# ????????? API

Справочник публичного интерфейса SDK `artex-robot` версии 0.1.1.

## ????? Robot

```
from artex_robot import Robot
robot = Robot(conf: dict = {})
```

Создаёт робота, загружает конфигурацию из `params.json` и словаря `conf` (приоритет у `conf`), инициализирует журнал и Vault. Соединение с сервером **не** устанавливается — для этого вызовите `init_connector()`. Параметры — на странице [Конфигурация](#).

## ?????????

### @robot.com\_handler

Регистрирует обработчик команд. **Обязателен** — должен быть объявлен до вызова `init_connector()`.

```
@robot.com_handler
def on_command(type_command: str, command, author=None):
    ...
```

| Аргумент                  | Описание                                                                                                                    |
|---------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| <code>type_command</code> | Тип команды: <code>com</code> , <code>force_com</code> , <code>sys_com</code> и др. (см. <a href="#">Обработка команд</a> ) |
| <code>command</code>      | Сама команда — строка или структура, которую отправил оператор/модель                                                       |
| <code>author</code>       | Идентификатор автора команды (оператор или модель ИИ)                                                                       |

Обработчик вызывается в потоке Socket.IO-клиента, а не в основном цикле. Долгие действия выполняйте асинхронно (см. пример колёсного робота на странице [Примеры роботов](#)).

## @robot.telemetry(key)

Помечает функцию как источник телеметрии. При каждом вызове функции её результат записывается в Vault под ключом `key`.

```
@robot.telemetry("battery")
def read_battery():
    return adc.read_voltage()
```

- Результат должен сериализоваться в JSON-совместимый вид: число, строка, bool, список, словарь.
- На сервер отправляется **весь** словарь телеметрии, например `{"battery": 11.8, "speed": 0.4}`.
- Несколько функций с разными ключами дополняют общий словарь.
- Функция-обёртка ничего не возвращает — чтобы использовать значение ещё и локально, прочитайте датчик отдельно.

## @robot.video(cam\_id)

Помечает функцию как источник видео для камеры `cam_id`. Результат функции — кадр OpenCV (numpy-массив BGR) — сжимается в JPEG (качество 80) и записывается в Vault.

```
@robot.video("front")
def send_front(frame):
    return frame
```

- Функция **не должна** возвращать `None` — это вызовет ошибку OpenCV.
- Несколько камер — несколько функций с разными `cam_id`; на сервер уходит словарь `{"front": <jpeg>, "rear": <jpeg>}`.
- Если `cam_id` содержит `led` или `rgb_strip`, значение **не** кодируется в JPEG и передаётся как есть. Так LED-устройства передают своё текущее состояние (например, массив цветов светодиодной ленты).

## @robot.mainloop

Регистрирует функцию основного цикла. **Обязателен.**

```
@robot.mainloop
def main_loop(context: dict):
    ...
```

Функция вызывается бесконечно с паузой `max_main_freq` секунд. Аргумент `context` — словарь контекста робота (см. ниже). Исключения внутри функции не перехватываются SDK и завершают программу — обрабатывайте их сами.

## ??????

| Метод                                                                                                               | Описание                                                                                                                                                                                                                                                                                         |
|---------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>robot.init_connector()</code>                                                                                 | Подключается к серверу <code>server:port</code> , namespace <code>/robot</code> , передаёт паспорт устройства. Включено авто-переподключение: 5 попыток, задержка 1-5 с. Если сервер недоступен при первом подключении — выбрасывает исключение <code>socketio.exceptions.ConnectionError</code> |
| <code>robot.start()</code>                                                                                          | Запускает основной цикл. Блокирует выполнение. По Ctrl+C корректно останавливает потоки и отключается                                                                                                                                                                                            |
| <code>robot.stop()</code>                                                                                           | Останавливает отправку телеметрии и видео и отключается от сервера                                                                                                                                                                                                                               |
| <code>robot.set_context(key, value)</code>                                                                          | Кладёт объект в контекст робота                                                                                                                                                                                                                                                                  |
| <code>robot[key] = value</code> , <code>robot[key]</code> , <code>del robot[key]</code> , <code>key in robot</code> | То же через синтаксис словаря                                                                                                                                                                                                                                                                    |

## ????????

Контекст — словарь для объектов, которые нужны основному циклу и обработчикам: камеры, драйверы моторов, последние команды, состояние автомата.

```
robot["cam"] = cv2.VideoCapture(0)
robot["motors"] = MotorDriver()

@robot.mainloop
def loop(ctx):
    ok, frame = ctx["cam"].read()
```

## ?????? robot.log

```
robot.log("INFO", "Моторы инициализированы")
robot.log("WARN", "Низкий заряд")
robot.log("ERROR", "Нет ответа от контроллера")
```

```
robot.log.set_level("skip_TODO")    # скрыть служебные TODO-сообщения
robot.log.set_type("file")          # писать в all.log / error.log
```

| Уровень | Использование                                                                                            |
|---------|----------------------------------------------------------------------------------------------------------|
| INFO    | Штатные события                                                                                          |
| WARN    | Предупреждения                                                                                           |
| ERROR   | Ошибки — при <code>logger_type = file</code> дублируются в <code>error.log</code>                        |
| TODO    | Служебные сообщения SDK (выводятся на каждой итерации цикла, скрываются уровнем <code>skip_TODO</code> ) |

Файлы журнала создаются в текущем рабочем каталоге. Сервер может запросить их удалённо (см. [Протокол обмена с сервером](#)).

???????????? ???? (???)  
????????????????????????????)

| Атрибут     | Назначение                                                                                                                                                                              |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| robot.vault | Хранилище: <code>set telem(key, val)</code> , <code>set video(cam_id, frame)</code> , <code>get smeta()</code> , <code>get dmeta(key)</code>                                            |
| robot.sio   | Коннектор; <code>robot.sio.sio</code> — клиент <code>socketio.Client</code> , <code>robot.sio.emit(event, data)</code> — отправка произвольного события в namespace <code>/robot</code> |

Пример прямой записи телеметрии без декоратора: `robot.vault.set telem("temp", 36.6)`.