

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

Готовые программы роботов и устранение неполадок

- [Примеры роботов](#)
- [Устранение неполадок](#)

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

Готовые шаблоны программ для типовых устройств. Перед запуском создайте `params.json` с адресом сервера (см. [Конфигурация](#)).

?????? 1. ?????????????? ?????? (??? LED)

Страница LED в веб-интерфейсе позволяет выбрать цвет (`#RRGGBB`) и яркость (0–100 %). Робот хранит текущее состояние и возвращает его в телеметрии.

```
from artex_robot import Robot

robot = Robot({"type": "LED", "name": "lamp-01", "telem_can": True})
robot.log.set_level("skip_TODO")

state = {"color": "#FFFFFF", "brightness": 100}

def hex_to_rgb(value):
    value = value.lstrip("#")
    return tuple(int(value[i:i + 2], 16) for i in (0, 2, 4))

@robot.com_handler
def on_command(type_command, command, author=None):
    robot.log("INFO", f"LED: {type_command}={command} от {author}")
    if isinstance(command, str) and command.startswith("#") and len(command) == 7:
        state["color"] = command.upper()
    elif isinstance(command, (int, float)) or (isinstance(command, str) and
command.isdigit()):
        state["brightness"] = max(0, min(100, int(command)))
    # здесь – вывод на реальный светодиод (PWM, NeoPixel и т.п.)
    r, g, b = hex_to_rgb(state["color"])
    k = state["brightness"] / 100
    print("RGB:", int(r * k), int(g * k), int(b * k))

@robot.telemetry("led")
```

```
def report():
    return dict(state)

@robot.mainloop
def loop(ctx):
    report()

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

Формат команд цвета и яркости зависит от версии веб-интерфейса — проверьте его по журналу робота (сообщение `recieved command from ...`) и при необходимости поправьте разбор в `on_command`.

?????? 2. ?????????? ?????? ? ?????????? (Raspberry Pi)

Робот на двух моторах (драйвер L298N / TB6612), FPV-камера, контроль заряда АКБ, остановка при потере команд.

```
import time
import cv2

from gpiozero import Robot as Chassis    # pip install gpiozero
from artex_robot import Robot

robot = Robot({"type": "Robot", "name": "tank-01",
              "telem_can": True, "video_can": True,
              "max_main_freq": 0.05, "max_send_freq": 0.05})
robot.log.set_level("skip_TODO")

chassis = Chassis(left=(17, 18), right=(22, 23))    # номера GPIO — по схеме робота
cam = cv2.VideoCapture(0)
cam.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter_fourcc(*"MJPG"))
cam.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
cam.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)

robot["speed"] = 0.6
robot["last_cmd"] = 0.0
```

```
COMMAND_TIMEOUT = 0.5
```

```
MOVES = {  
    "forward": lambda s: chassis.forward(s),  
    "back":    lambda s: chassis.backward(s),  
    "left":    lambda s: chassis.left(s),  
    "right":   lambda s: chassis.right(s),  
    "stop":    lambda s: chassis.stop(),  
}
```

```
@robot.com_handler
```

```
def on_command(type_command, command, author=None):  
    if isinstance(command, str) and command.startswith("speed:"):   
        robot["speed"] = int(command.split(":")[1]) / 100  
        return  
  
    move = MOVES.get(command)  
    if move is None:  
        robot.log("WARN", f"Неизвестная команда: {command}")  
        return  
  
    move(robot["speed"])  
    robot["last_cmd"] = time.monotonic()
```

```
@robot.telemetry("battery")
```

```
def read_battery():  
    return 11.9          # замените чтением АЦП (например, ADS1115)
```

```
@robot.telemetry("motion")
```

```
def read_motion():  
    return {"left": chassis.left_motor.value, "right": chassis.right_motor.value}
```

```
@robot.video("rgb")
```

```
def send_frame(frame):  
    return frame
```

```
@robot.mainloop
```

```
def loop(ctx):  
    # сторожевой таймер: нет команд — стоим  
    if time.monotonic() - ctx["last_cmd"] > COMMAND_TIMEOUT:  
        chassis.stop()
```

```

read_battery()
read_motion()

ok, frame = cam.read()
if ok:
    send_frame(frame)

try:
    robot.init_connector()
    robot.start()
finally:
    chassis.stop()
    cam.release()

```

?????? 3. ?????????????? ?????? (??? ?????????????????)

Робот-симулятор: хранит координаты и курс, рисует «вид сверху» в кадр и передаёт телеметрию. Подходит для отладки интерфейса, моделей ИИ и сервера без реального железа.

```

import math, time
import numpy as np, cv2
from artex_robot import Robot

robot = Robot({"type": "Robot", "name": "sim-01",
              "telem_can": True, "video_can": True})
robot.log.set_level("skip_TODO")

sim = {"x": 0.0, "y": 0.0, "heading": 0.0, "v": 0.0, "w": 0.0, "battery": 12.6}
ACTIONS = {"forward": (0.5, 0), "back": (-0.5, 0), "left": (0, 1.2),
           "right": (0, -1.2), "stop": (0, 0)}

@robot.com_handler
def on_command(type_command, command, author=None):
    if command in ACTIONS:

```

```

sim["v"], sim["w"] = ACTIONS[command]

@robot.telemetry("pose")
def pose():
    return {k: round(sim[k], 3) for k in ("x", "y", "heading")}

@robot.telemetry("battery")
def battery():
    return round(sim["battery"], 2)

@robot.video("rgb")
def render(frame):
    return frame

def draw():
    img = np.full((360, 360, 3), 40, np.uint8)
    cx, cy = int(180 + sim["x"] * 40), int(180 - sim["y"] * 40)
    hx = int(cx + 15 * math.cos(sim["heading"]))
    hy = int(cy - 15 * math.sin(sim["heading"]))
    cv2.circle(img, (cx, cy), 12, (0, 200, 255), -1)
    cv2.line(img, (cx, cy), (hx, hy), (255, 255, 255), 2)
    return img

@robot.mainloop
def loop(ctx):
    dt = 0.1
    sim["heading"] += sim["w"] * dt
    sim["x"] += sim["v"] * math.cos(sim["heading"]) * dt
    sim["y"] += sim["v"] * math.sin(sim["heading"]) * dt
    sim["battery"] = max(10.0, sim["battery"] - 0.0005 - 0.002 * abs(sim["v"]))
    pose(); battery(); render(draw())

robot.init_connector()
robot.start()

```

?????? 4. ?????????? ??????? ??:
 ?????????? ? ??????????????

Бортовой компьютер может сам обрабатывать видео и отправлять на сервер не только кадры, но и выделенные признаки — например, результаты детекции. Модели ИИ на сервере получают их вместе с телеметрией.

```
from ultralytics import YOLO          # pip install ultralytics
model = YOLO("yolov8n.pt")           # лёгкая модель для бортового компьютера

@robot.telemetry("detections")
def detections(frame):
    result = model(frame, verbose=False)[0]
    return [
        {"cls": result.names[int(b.cls)], "conf": round(float(b.conf), 2),
         "xyxy": [round(v) for v in b.xyxy[0].tolist()]}
        for b in result.bboxes
    ]

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

На слабых бортовых компьютерах запускать детекцию не на каждом кадре (например, раз в 3–5 итераций) или переносите её в вычислительную среду платформы — см. книгу «Управление роботами с помощью ИИ».

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

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

Симптом	Причина	Решение
<code>ModuleNotFoundError: No module named 'cv2'</code> при импорте SDK	OpenCV не входит в зависимости пакета	<code>pip install opencv-python</code> (или <code>opencv-python-headless</code>) и <code>numpy</code>
<code>AttributeError: 'Conductor' object has no attribute 'command_handler'</code>	Не объявлен обработчик команд до <code>init_connector()</code>	Объявите функцию с декоратором <code>@robot.com_handler</code> (можно пустую)
<code>AttributeError: ... 'main_loop_func'</code>	Не объявлен основной цикл	Объявите функцию с декоратором <code>@robot.mainloop</code>
<code>TypeError: ... takes 0 positional arguments but 1 was given</code> в основном цикле	Функция основного цикла не принимает <code>context</code>	<code>def loop(context): ...</code>
<code>socketio.exceptions.ConnectionError</code> при запуске	Сервер недоступен или неверный адрес	Проверьте <code>server</code> и <code>port</code> в <code>params.json</code> , доступность сервера: <code>curl http://<сервер>:<порт>/socket.io/?EIO=4&transport=polling</code>
Робот не находит сервер <code>artex.arcon</code>	Не настроено имя по умолчанию	Укажите IP-адрес сервера в <code>params.json</code> или добавьте запись в <code>/etc/hosts</code>
Ошибка OpenCV <code>!_img.empty()</code> в <code>imencode</code>	В функцию <code>@robot.video</code> передан/возвращён пустой кадр (<code>None</code>)	Проверяйте результат <code>cam.read()</code> перед отправкой кадра
Параметры из <code>params.json</code> не применяются	Программа запущена из другого каталога	Запускайте из каталога с <code>params.json</code> или задайте <code>WorkingDirectory</code> в <code>systemd</code>
Консоль заполнена сообщениями <code>[TODO]</code>	Уровень журнала <code>all</code>	<code>"logger_level": "skip_TODO"</code> или <code>robot.log.set_level("skip_TODO")</code>
Робот в списке, но видео/телеметрии нет	Сервер не прислал <code>Ready</code> , или функции-источники не вызываются в основном цикле	Проверьте в журнале сообщение <code>in heartbeat recieved [...]</code> ; вызовите функции <code>@robot.telemetry</code> / <code>@robot.video</code> в <code>@robot.mainloop</code>
Команды не доходят до обработчика	Робот ещё не получил <code>Ready</code>	Дождитесь <code>heartbeat</code> ; проверьте журнал на <code>recieved command</code>
Видео отстаёт или «рвётся»	Слишком большие кадры для канала	Уменьшите разрешение (например, 320×240), увеличьте <code>max_send_freq</code> до 0.1

Симптом	Причина	Решение
Программа «висит» без связи после перезапуска сервера	SDK исчерпал 5 попыток переподключения	Используйте сторожевую проверку (ниже) и автоперезапуск через systemd
Обработчик команд подтормаживает	Долгие операции в <code>com_handler</code> блокируют приём событий	Выполняйте долгие действия в отдельном потоке, в обработчике только меняйте состояние

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

SDK делает 5 попыток переподключения. Если сервер был недоступен дольше, соединение не восстановится само. Завершите программу — systemd перезапустит её, и подключение будет выполнено заново:

```
import os, time

DISCONNECT_LIMIT = 30 # секунд

@robot.mainloop
def loop(ctx):
    if robot.sio.sio.connected:
        ctx["online_at"] = time.monotonic()
    elif time.monotonic() - ctx.get("online_at", time.monotonic()) > DISCONNECT_LIMIT:
        robot.log("ERROR", "Нет связи с сервером, перезапуск")
        stop_motors()
        os._exit(1)
    ...
```

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

SDK не перехватывает исключения в функции основного цикла — любая ошибка завершает программу. Для долгой автономной работы оборачивайте опасные участки:

```
@robot.mainloop
def loop(ctx):
    try:
        read_sensors()
    except Exception as ex:
```

```
robot.log("ERROR", f"Датчики: {ex}")
```

??????????

1. Включите полный журнал: `"logger_level": "all", "logger_type": "console"`.
2. Проверьте последовательность в журнале: `connected` → `in heartbeat recieved`
`['d_meta', 'Ready']` → `recieved command ...`.
3. Проверьте работа без платформы на [минимальном тестовом сервере](#).
4. Запросите журнал робота удалённо командой `{"file": "log"}` (при `"logger_type": "file"`).

?????????? ?????????????? ???????? 0.1.1

- `opencv-python` не указан в зависимостях пакета.
- Параметры `max_send_freq` и `max_main_freq` — это периоды в секундах.
- Поле `sid` в состоянии робота не заполняется SDK (всегда `null`).
- Команды `web_settings` принимаются, но не обрабатываются.
- Раздел `external` конфигурации зарезервирован и не используется.
- Нет встроенного сторожевого таймера команд и соединения — реализуйте их в своей программе (примеры выше и на странице [Обработка команд](#)).