

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

Команды приходят роботу от оператора (веб-интерфейс) или от модели ИИ через сервер. SDK разбирает сообщение и вызывает ваш обработчик `@robot.com_handler`.

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

Оператор / модель ИИ → ХАБ → событие "command" → SDK → `on_command(type_command, command, author)`

Сообщение от сервера имеет вид:

```
{
  "author": "operator-7",
  "payload": { "com": "forward" }
}
```

Ключ внутри `payload` — **тип команды**, значение — **сама команда**. SDK передаёт их в обработчик как `type_command="com"`, `command="forward"`, `author="operator-7"`.

Команды принимаются только после того, как сервер прислал сигнал готовности (`Ready` в `heartbeat`). Команды, пришедшие раньше, игнорируются.

???? ?????

Тип	Кто обрабатывает	Назначение
<code>com</code>	Ваш обработчик	Обычная команда управления
<code>force_com</code>	Ваш обработчик	Приоритетная команда — выполнить немедленно, прервав текущее действие (например, экстренная остановка)
<code>sys_com</code>	Ваш обработчик	Системная команда (перезагрузка, калибровка, смена режима)
<code>file</code>	SDK автоматически	Запрос журналов: <code>log</code> , <code>error</code> , <code>clear</code> (см. Протокол)


```

elif command == "disarm": motors.power_off()
elif isinstance(command, str) and command.startswith("speed:"):
    motors.set_speed(int(command.split(":", 1)[1]) / 100)
else:
    robot.log("WARN", f"Неизвестная команда {command}")
    return
ctx["last_command_time"] = time.monotonic()

```

```

@robot.com_handler
def on_command(type_command, command, author=None):
    robot.log("INFO", f"{type_command}:{command} от {author}")

    # Остановка – всегда и немедленно, от любого источника
    if command in STOP_WORDS or type_command == "force_com":
        apply(command, robot.context)
        return

    if type_command == "com":
        apply(command, robot.context)
    elif type_command == "sys_com":
        handle_system(command)

```

?????????????: ??????????? ???
 ??????? ????????

Если связь прервалась в момент движения, робот продолжит ехать с последней командой. Добавьте в основной цикл «сторожевой таймер»: если команд не было дольше заданного времени — остановка.

```

COMMAND_TIMEOUT = 0.5    # секунд

@robot.mainloop
def loop(ctx):
    last = ctx.get("last_command_time", 0)
    if time.monotonic() - last > COMMAND_TIMEOUT:
        ctx["motors"].drive(0, 0)

```

Для режима, когда оператор удерживает кнопку «вперёд», интерфейс или модель должны повторять команду чаще, чем раз в `COMMAND_TIMEOUT`. Для дискретных команд (движение до следующей команды) увеличьте таймаут или привяжите остановку к событию отключения.

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

Правило приоритета применяется на сервере. Если роботу нужно учитывать его и на борту (например, при прямом подключении модели), различайте источники по `author`:

```
OPERATOR_HOLD = 2.0 # секунд после команды оператора модель не управляет

@robot.com_handler
def on_command(type_command, command, author=None):
    now = time.monotonic()
    is_model = str(author).startswith("model")
    if is_model and now - robot.context.get("operator_time", 0) < OPERATOR_HOLD:
        return # оператор недавно вмешался
    if not is_model:
        robot.context["operator_time"] = now
    apply(command, robot.context)
```

Префикс `model` в идентификаторе автора — соглашение для примера; используйте идентификаторы, которые выдаёт ваша платформа.

Revision #3

Created 2026-09-23 09:32:16 UTC by Маслов Алексей

Updated 2026-09-23 09:35:04 UTC by Маслов Алексей