# Microdrones VSM User Guide

UgCS 3.1.871

# Contents

# 1 Connecting Microdrones autopilot to UgCS

-

## 1.1 First time vehicle connection

See Disclaimer.

Please follow these steps to connect an Microdrones vehicle to the UgCS:

1. Microdrones vehicle must be properly configured, calibrated and tested using tools and instruction from the official `Microdrones web site` prior to using it with UgCS. UgCS does not support initial configuration, setup and calibration of Microdrones vehicles.
2. Turn on the vehicle. There are two communication channels in a drone - uplink and downlink. Uplink channel is connected via USB-serial cable, plug it in the drone connector (you should connect to FC, not NC - connector flat side to the battery, see manufacturer manual for detailed explanation) and to the computer where VSM is running. The uplink channel can be used for mission uploading and commands execution. For telemetry reception you need downlink channel which is connected to the downlink wire on the drone bottom. You can use manufacturer provided downlink equipment which uses video transmitter sound channel for telemetry data transferring and downlink decoder ground unit. Alternatively you can connect your own USB-serial cable or radio-modem to the downlink wire. Before that you should switch the telemetry output to digital mode using a jumper on the drone board, see manufacturer manual for the details.

   For initial vehicle set up in UgCS you can connect either uplink or downlink whichever is more convenient for you.

   For all connection options you will have USB-serial device on the PC side so proper OS driver for virtual serial port should be installed. Please refer to your communication equipment manufacturer documentation about driver installation instructions.
3. Open *Vehicles* window in UgCS Client and wait until new vehicle appears there automatically. Either Uplink or Downlink connection should be available. Select the necessary vehicle and click *Edit* to start editing the corresponding vehicle profile. Now you can change the default vehicle name to be convenient for you:

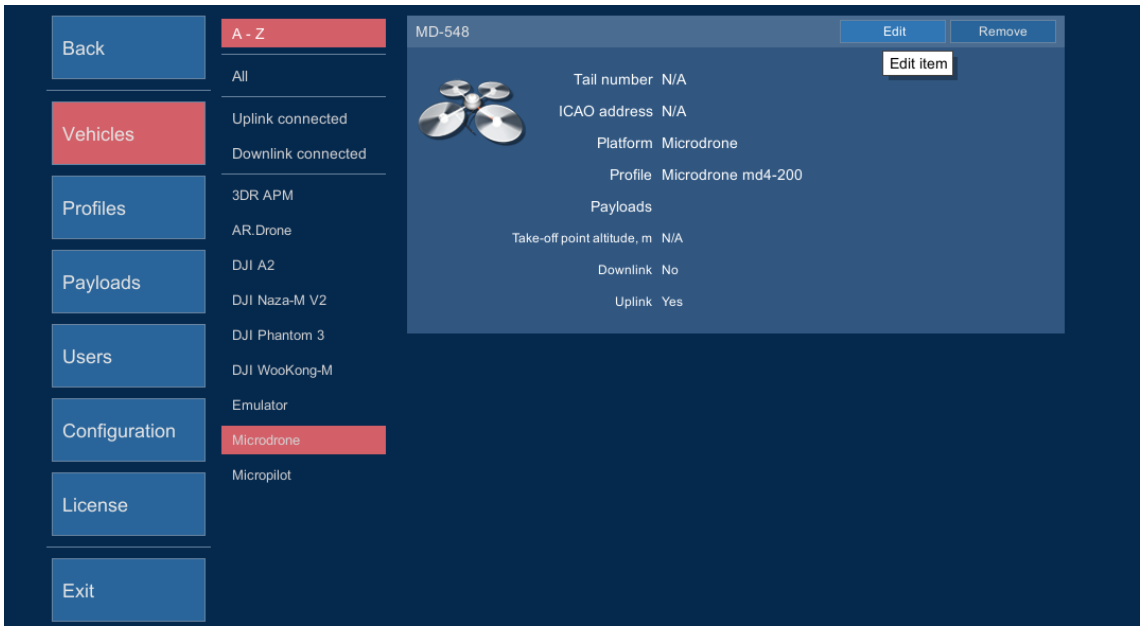


Figure 1: New Microdrones vehicle

4. Repeat steps above for each your Microdrones vehicle.

Note that if both uplink and downlink are connected only uplink will be active because the Microdrones stops telemetry sending after reception of commands via uplink. The downlink can be activated either by issuing "A-RM" command or by disconnecting uplink and re-inserting a battery.

## 1.2 Mission execution specifics

Home location cannot be modified by UgCS. It is controlled by the autopilot and is permanently bound to a take-off position.

## 1.3 Command execution specifics

| Command | Support | Notes |
|---|---|---|
| ARM | Partial | The command issuing does not turn on motors but allows doing it from RC without battery re-inserting. It also activates telemetry sending. |
| DISARM | Partial | Does not stop motors but activates uplink channel and deactivates telemetry sending. |
| AUTOMODE | No | |
| MANUALMODE | No | |
| CLICK & GO | No | |
| JOYSTICK | No | |
| HOLD | No | |
| CONTINUE | No | |
| RETURN HOME | No | |
| TAKEOFF | No | |
| LAND | No | |
| EMERGENCYLAND | No | |
| CAMERA_TRIGGER | No | |

## 1.4 Telemetry information specifics

Nothing specific.

## 1.5 Fail-safe actions

**GPS Lost:**

| Action | Result |
|---|---|
| Wait | Aircraft tries to maintain altitude |
| Land | Aircraft slowly decends |

**RC Lost:**

| Action | Result |
|---|---|
| Wait | Aircraft returns home and lands |
| Land | Aircraft returns home and lands |
| Return Home | Aircraft changes altitude to failsafe ALT(50m) and returns home |
| Continue | Aircaft continues mission |

**Battery Low:**

| Action | Result |
|---|---|
| Wait | If possible aircraft returns home and lands, if not possible slowly decends |
| Land | If possible aircraft returns home and lands, if not possible slowly decends |
| Return Home | If possible aircraft returns home and lands, if not possible slowly decends |
| Continue | Aircaft continues mission |

## 1.6 Configuration file

Default configuration file of the Microdrones VSM suits most needs and it is generally not necessary to modify it.

Configuration file location:

- **On Microsoft Windows:**

```
C:\Program Files (x86)\UgCS\bin\vsm-microdrones.conf
```

- **On GNU/Linux:**

```
/etc/opt/ugcs/vsm-microdrones.conf
```

- **On Apple OS X:**

```
/Users/[user name]/Library/Application Support/UGCS/configuration/vsm-microdrones.conf
```

### 1.6.1 Serial port configuration

Typically vehicle is connected to UgCS via radio datalink which appears as serial port when USB cable is plugged in. See Serial port configuration for details.

- **Example:**

```
connection.serial.1.name = COM21
connection.serial.1.baud = 38400
```

### 1.6.2 Common parameters

All VSMs share a common set of configuration file parameters described in Common configuration file parameters. Microdrones VSM configuration file prefix is:

```
vehicle.microdrones
```

### 1.6.3 Model name override

By default the VSM sets 'MD' model name for the Microdrones vehicles. It can be overriden to more specific name in the VSM configuration.

**Example:**

```
vehicle.microdrones.custom.my_drone.serial_number = 1102
vehicle.microdrones.custom.my_drone.model_name = MD4-200
```

In this example the model name for the drone with serial number '1102' is overriden to value 'MD4-200'.

## 1.7 Common configuration file parameters

VSM configuration file is a text file specified via command line argument - *-config* of the VSM application. Example:

```
--config /etc/opt/ugcs/vsm-ardupilot.conf
```

Each configuration parameter is defined as a line in the configuration file with the following structure:

```
name1.name2....nameX = value
```

where name1, name2 ... nameX are arbitrary names separated by dots to divide a variable into logical blocks and a value which can be a number value or a text string depending on the context. See below the description about common VSM configuration parameters.

### 1.7.1 UgCS server configuration

VSM can connect to UgCS in two different ways:

- Listen for connection form the UgCS server. See Listening address and Listening port.

  When VSM is configured in listening mode automatic VSM discovery can be set up, too. See Automatic service discovery

- Initiate connection to UgCS server. See UgCS server address and UgCS server port.

At least one of the above must be configured for VSM to work.

#### 1.7.1.1 Listening address

Optional.

- **Name:** ucs.local_listening_address = [IP address]
- **Description:** Local address to listen for incoming connections from UgCS server.
- **Default:** 0.0.0.0 (listen on all local addresses)
- **Example:** ucs.local_listening_address = 10.0.0.2

#### 1.7.1.2 Listening port

Optional.

- **Name:** ucs.local_listening_port = [port number]
- **Description:** Local TCP port to listen for incoming connections from UgCS server.
- **Example:** ucs.local_listening_port = 5556

#### 1.7.1.3 UgCS server address

Optional.

- **Name:** ucs.address = [IP address]
- **Description:** UgCS server address to connect to.
- **Example:** ucs.address = 1.2.3.4

#### 1.7.1.4 UgCS server port

Optional.

- **Name:** ucs.port = [port number]
- **Description:** UgCS server port.
- **Example:** ucs.port = 3335

#### 1.7.1.5 Retry timeout

Optional.

- **Name:** ucs.retry_timeout = [seconds]
- **Description:** Retry timeout for outgoing server connections in seconds.
- **Default:** 10
- **Example:** retry_timeout = 11

### 1.7.2 Automatic service discovery

VSM can respond to automatic service discovery requests form UgCS server.

When this parameter is not configured, service discovery is disabled.

Optional.

- **Name:** service_discovery.vsm_name = [Service name]
- **Description:** Human readable service name.
- **Example:** service_discovery.vsm_name = Ardupilot VSM

### 1.7.3 Logging configuration

#### 1.7.3.1 Level

Optional.

- **Name:** log.level = [error|warning|info|debug]
- **Description:** Logging level.
- **Default:** info
- **Example:** log.level = debug

#### 1.7.3.2 File path

Optional.

- **Name:** log.file_path = [path to a file]
- **Description:** Absolute or relative (to the current directory) path to a logging file. Logging is disabled if logging file is not defined. File should be writable. Backslash should be escaped with a backslash.
- **Example:** log.file = /var/opt/ugcs/log/vsm-ardupilot/vsm-ardupilot.log
- **Example:** log.file = C:\\Users\\John\\AppData\\Local\\UGCS\\logs\\vsm-ardupilot\\vsm-ardupilot.log

#### 1.7.3.3 Maximum single file size

Optional.

- **Name:** log.single_max_size = [size]
- **Description:** Maximum size of a single log file. When maximum size is exceeded, existing file is renamed by adding a time stamp and logging is continued into the empty file. [size] should be defined as a number postfixed by a case insensitive multiplier:
  - Gb, G, Gbyte, Gbytes: for Giga-bytes
  - Mb, M, Mbyte, Mbytes: for Mega-bytes
  - Kb, K, Kbyte, Kbytes: for Kilo-bytes
  - no postfix: for bytes
- **Default:** 100 Mb
- **Example:** log.single_max_size = 500 Mb

#### 1.7.3.4 Maximum number of old log files

Optional.

- **Name:** log.max_file_count = [number]
- **Description:** Log rotation feature. Maximum number of old log files to keep. After reaching single_max_size of current log file VSM will rename it with current time in extension and start new one. VSM will delete older logs so the number of old logs does not exceed the max_file_count.
- **Default:** 1
- **Example:** log.max_file_count = 5
  -

### 1.7.4 Mission dump path

Optional.

- **Name:** [prefix].mission_dump_path = [path to a file]
- **Description:** File to dump all generated missions to. Timestamp is appended to the name. Delete the entry to disable mission dumping. All directories in the path to a file should be already created.
- **Example:** vehicle.ardupilot.mission_dump_path = C:\\tmp\\ardupilot_dump

### 1.7.5 Command execution control

When vehicle is connected via unreliable link VSM will retry each command several times before failing. This section describes the parameters which control the command execution.

#### 1.7.5.1 Command try count

- **Name:** vechicle.command_try_count = <number of="" times>=""></number>
- **Description:** Number of times the command will be issued before declaring it as failed. Must be greater than zero.
- **Default:** 3
- **Example:** vechicle.command_try_count = 5

#### 1.7.5.2 Command timeout

- **Name:** vechicle.command_timeout = <timeout in="" seconds>=""></timeout>
- **Description:** Time to wait for response on issued command before retrying.
- **Unit:** second
- **Default:** 1

## 1.8 Communication with devices

VSM can communicate with Vehicle over different communication channels

Currently supported channels:

- Serial port, see Serial port configuration for details.
- Outgoing TCP, see Outgoing TCP connection configuration for details.
- Incoming TCP, see Incoming TCP connection configuration for details.
- Outgoing UDP, see Outgoing UDP connection configuration for details.
- Incoming UDP, see Incoming UDP connection configuration for details.
- Incoming UDP (any peer), see Incoming UDP connection configuration (any peer) for details.
- Named pipe , see Named pipes for details.
- vsm-proxy (XBee), see Proxy configuration for details.

### 1.8.1 Serial port configuration

VSM which communicates with vehicles via serial ports should define at least one serial port, otherwise VSM will not try to connect to the vehicles. Port name and baud rate should be both defined.

#### 1.8.1.1 Port name

Required.

- **Name:** connection.serial.[index].name = [regular expression]
- **Description:** Ports which should be used to connect to the vehicles by given VSM. Port names are defined by a [regular expression] which can be used to define just a single port or create a port filtering regular expression. Expression is case insensitive on Windows. [index] is a arbitrary port indexing name.
- **Example:** connection.serial.1.name = /dev/ttyUSB[0-9]+|com[0-9]+
- **Example:** connection.serial.2.name = com42

#### 1.8.1.2 Port baud rate

Required.

- **Name:** connection.serial.[index].baud.[baud index] = [baud]
- **Description:** Baud rate for port opening. [baud index] is an optional arbitrary name used when it is necessary to open the same serial port using multiple baud rates. [index] is an arbitrary port indexing name.
- **Example:** connection.serial.1.baud.1 = 9600
- **Example:** connection.serial.1.baud.2 = 57600
- **Example:** connection.serial.2.baud = 38400

#### 1.8.1.3 Excluded port name

Optional.

- **Name:** connection.serial.exclude.[exclude index] = [regular expression]
- **Description:** Ports which should not be used for vehicle access by this VSM. Port names are defined by a [regular expression] which can be used to define just a single port or create a port filtering regular expression. Filter is case insensitive on Windows. [exclude index] is a arbitrary indexing name used when more than one exclude names are defined.
- **Example:** connection.serial.exclude.1 = /dev/ttyS.∗
- **Example:** connection.serial.exclude = com1

#### 1.8.1.4 Serial port arbiter

Optional.

- **Name:** connection.serial.use_arbiter = [yes|no]
- **Description:** Enable (yes) or disable (no) serial port access arbitration between VSMs running on the same machine. It is recommended to have it enabled to avoid situation when multiple VSMs try to open the same port simultaneously.
- **Default:** yes
- **Example:** connection.serial.use_arbiter = no

### 1.8.2 Outgoing TCP connection configuration

VSM can be configured to connect to the vehicle via TCP. VSM will try to establish connection to the specified address:port.

Used to connect to vehicle simulator or when vehicle is equipped with WiFi adapter.

#### 1.8.2.1 Remote TCP port

Required.

- **Name:** connection.tcp_out.[index].port = [port number]
- **Description:** Remote port to connect to.
- **Example:** connection.tcp_out.1.port = 5762

#### 1.8.2.2 IP-address for outgoing TCP connection

Required.

- **Name:** connection.tcp_out.[index].address = [IP-address]
- **Description:** IP-address of vehicle to connect to.
- **Example:** connection.tcp_out.1.address = 10.0.0.111

### 1.8.3 Incoming TCP connection configuration

VSM can be configured to listen for incoming TCP connections from the vehicle. Multiple vehicles are supported on the same port.

Used to connect to vehicle equipped with WiFi adapter.

#### 1.8.3.1 Local listening TCP port

Required.

- **Name:** connection.tcp_in.[index].local_port = [port number]
- **Description:** Remote port to connect to.
- **Example:** connection.tcp_in.1.local_port = 5762

#### 1.8.3.2 Local IP address

Optional.

- **Name:** connection.tcp_in.[index].local_address = [IP-address]
- **Description:** Local ip address to bind to.
- **Default:** 0.0.0.0 (all interfaces)
- **Example:** connection.tcp_in.1.local_address = 127.0.0.1

### 1.8.4 Outgoing UDP connection configuration

VSM can be configured to connect to the vehicle via UDP. VSM will try to establish UDP connection to the specified address:port.

#### 1.8.4.1 Remote IP-address for UDP

Required.

- **Name:** connection.udp_out.[index].address = [IP-address]
- **Description:** Remote IP-address to send outgoing UDP packets to.
- **Example:** connection.udp_out.1.address = 192.168.1.1

#### 1.8.4.2 Remote UDP port

Required.

- **Name:** connection.udp_out.[index].port = [port number]
- **Description:** Remote UDP port to send outgoing packets to.
- **Example:** connection.udp_out.1.port = 14551

#### 1.8.4.3 Local IP-address for UDP

Optional.

- **Name:** connection.udp_out.[index].local_address = [IP-address]
- **Description:** Local ip address to bind to.
- **Default:** 0.0.0.0 (bind to all interfaces)
- **Example:** connection.udp_out.1.local_address = 0.0.0.0

#### 1.8.4.4 Local UDP port

Optional.

- **Name:** connection.udp_out.[index].local_port = [port number]
- **Description:** Local UDP port to listen for incoming packets on.
- **Default:** 0 (bind to random port)
- **Example:** connection.udp_out.1.local_port = 14550

### 1.8.5 Incoming UDP connection configuration

VSM can be configured to listen for UDP conncections from the vehicle. Vehicle must be actively sending heartbeat/telemetry on specified UDP port before it can be detected by VSM. VSM will automatically detect multiple vehicles on the same port. This is very useful for "drone swarm" setups as there is no need to specify connector for each vehicle and no need to know the IP address of each vehicle in advance.

#### 1.8.5.1 Local UDP port

Required.

- **Name:** connection.udp_in.[index].local_port = [port number]
- **Description:** Local UDP port to listen for incoming packets on.
- **Example:** connection.udp_in.1.local_port = 14550

#### 1.8.5.2 Local IP-address for UDP

Optional.

- **Name:** connection.udp_in.[index].local_address = [IP-address]
- **Description:** Local ip address to bind to.
- **Default:** 0.0.0.0 (bind to all interfaces)
- **Example:** connection.udp_in.1.local_address = 0.0.0.0

### 1.8.6 Incoming UDP connection configuration (any peer)

This connection type is similar to "udp_in" with the exception that all incoming traffic will be recevied as one stream. It is used for special purpose connections and cannot be used to connect vehicles.

#### 1.8.6.1 Local UDP port

Required.

- **Name:** connection.udp_any.[index].local_port = [port number]
- **Description:** Local UDP port to listen for incoming packets on.
- **Example:** connection.udp_any.1.local_port = 14550

#### 1.8.6.2 Local IP-address for UDP

Optional.

- **Name:** connection.udp_any.[index].local_address = [IP-address]
- **Description:** Local ip address to bind to.
- **Default:** 0.0.0.0 (bind to all interfaces)
- **Example:** connection.udp_any.1.local_address = 0.0.0.0

### 1.8.7 Named pipes

VSM is able to communicate with vehicle over named pipe. The pipe must already exist.

- **Name:** connection.pipe.[index].port = pipe_name
- **Description:** Pipe name
- **Example:** connection.pipe.1.name = \\pipe\my_named_pipe

### 1.8.8 Proxy configuration

VSM is able to communicate with vehicle via proxy service which redirects dataflow received from vehicle through TCP connection to VSM and vice versa using specific protocol. In other words proxy service appears as a router between vehicle and VSM. At the moment there is one implementation of proxy in UgCS called XBee Connector which retranslates data from ZigBee network to respective VSM.

#### 1.8.8.1 IP-address for proxy

Required.

- **Name:** connection.proxy.[index].address = [IP-address]
- **Description:** IP-address to connect proxy to. Specify local or remote address.
- **Example:** connection.proxy.1.address = 127.0.0.1

#### 1.8.8.2 TCP port for proxy

Required.

- **Name:** connection.proxy.[index].port = [port number]
- **Description:** TCP port to be connected with proxy through. Should be the same as in congiguration on proxy side.
- **Example:** connection.proxy.1.port = 5566

## 2 Disclaimer

DISCLAIMER OF WARRANTIES AND LIMITATIONS ON LIABILITY.

(a) SMART PROJECTS HOLDINGS MAKE NO REPRESENTATIONS OR WARRANTIES REGARDING THE ACCURACY OR COMPLETENESS OF ANY CONTENT OR FUNCTIONALITY OF THE PRODUCT AND ITS DOCUMENTATION.

(b) SMART PROJECTS HOLDINGS DISCLAIM ALL WARRANTIES IN CONNECTION WITH THE PRODUCT, AND WILL NOT BE LIABLE FOR ANY DAMAGE OR LOSS RESULTING FROM YOUR USE OF THE PRODUCT. INCLUDING BUT NOT LIMITED TO INJURY OR DEATH OF USER OR ANY THIRD PERSONS OR DAMAGE TO PROPERTY.

(c) THE SOFTWARE IS SUPPLIED AS IS WITH NO WARRANTIES AND CAN BE USED ONLY AT USERS OWN RISK.