Parrot AR.Drone VSM User Guide

UgCS 2.11.227

Contents

1 Connecting AR.Drone to UgCS

1.1 First time vehicle connection

See Disclaimer.

Only AR.Drone version 2.0 is supported.

Please follow these steps to connect an AR.Drone vehicle to the UgCS:

1. AR.Drone should have Flight Recorder installed. See manufacturer manual for details as well as official web-site: http://ardrone2.parrot.com/

2. The connection to the drone is based on Wi-Fi network. The drone creates Wi-Fi access point (typically named "ardrone2_XXXXXX") to which your PC should connect (the one where VSM is running).

   Attention

   Notes on ArDrone WiFi connectivity:

   - Always ensure you have connected to the ArDrone Wi-Fi access point.
   - Reconnect to ArDrone manually.
     It is recommended to manually reconnect to ArDrone WiFi network after power-cycling the ArDrone. E.g. after battery change. This is because it can take significant time until OS reconnects to the WiFi automatically.
   - ArDrone does not support multiple controllers connected to single drone.
     Please make sure your iPad or iPhone device is disconnected from ArDrone WiFi and you do not have ArDrone applications running on your iPad or iPhone before connecting to UgCS.
     Please disconnect the computer running UgCS from ArDrone WiFi network before you try to connect your iPad or iPhone to ArDrone WiFi.

3. As soon as uplink and downlink connection is established, the vehicle should appear in the active vehicles list in main (map) view. Open *Vehicles* window from main menu and choose the corresponding vehicle for editing by clicking on the menu item and selecting *Edit* button. Now you can select the vehicle profile and change the default vehicle name to be convenient for you:

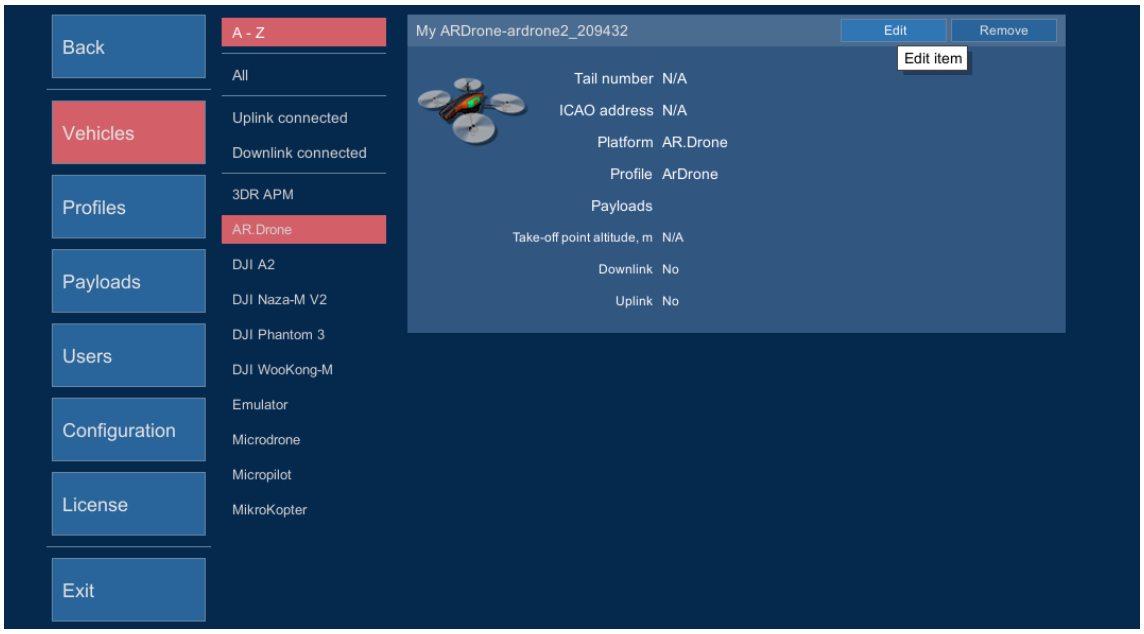

Figure 1: New AR.Drone vehicle

Vehicle profile needs to be assigned to allow mission planning with this vehicle. Vehicle avatar should be assigned in vehicle profile to properly see the vehicle location on map.

1.1.1 Multiple drones connection

All AR.Drone vehicles in default configuration use the same IP address, so it is not possible to connect multiple AR.Drone vehicles to the same computer without doing non-standard vehicle reconfiguration. Community written instructions about how to do it exist, but they are not officially supported by AR.Drone manufacturer Parrot. The easiest way how to connect multiple AR.Drone vehicles to UgCS is to connect each vehicle to its own computer with AR.Drone VSM running on it:

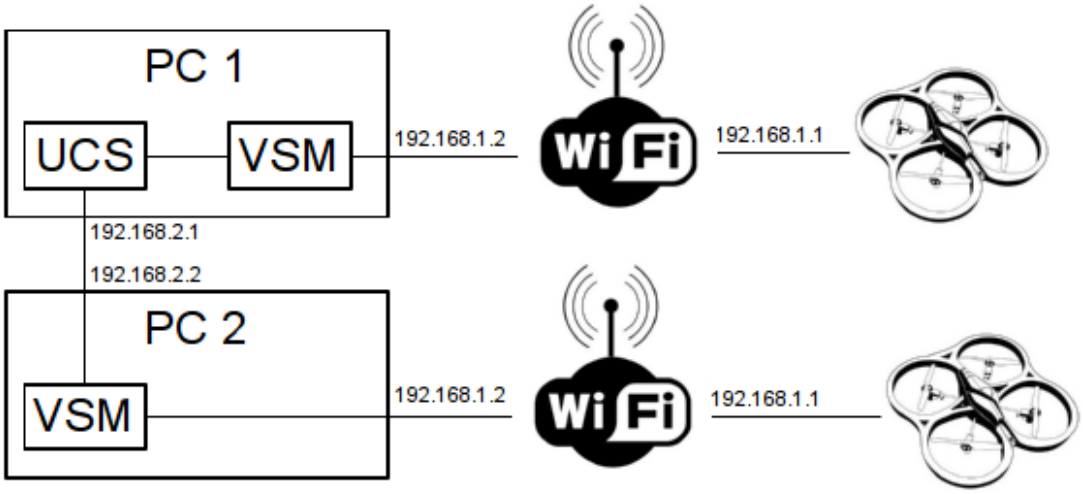

Figure 2: Multiple AR.Drone vehicles connection

In this case Automatic VSM Discovery protocol should discover all vehicles that are connected to PCs within current local area network. Some times it is necessary trough to add addresses of all computers to UgCS VSM

configuration:

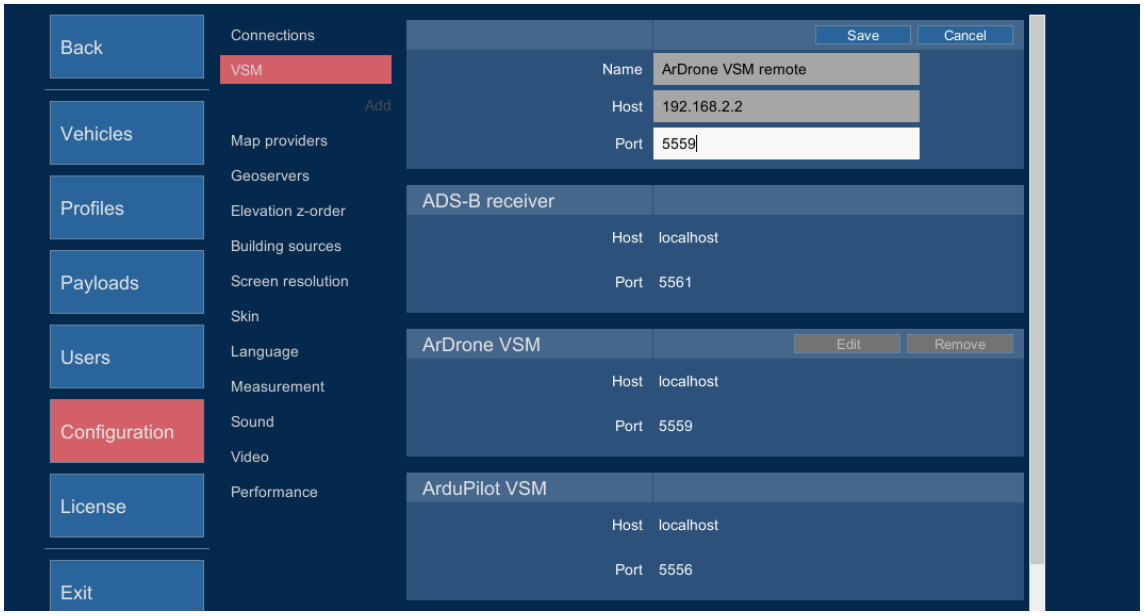

Figure 3: Adding AR.Drone VSM

AR.Drone vehicles in UgCS are identified by a MavLink ID which is also the same for all AR.Drone vehicles, thus does not allow to distinguish one vehicle from another. To overcome this problem, it is necessary to assign custom serial number for connected vehicle on each computer. That is, for example, AR.Drone on PC1 will have serial number 1, but AR.Drone on PC2 will have serial number 2:

Add these lines to the first VSM:

```
vehicle.ardrone.custom.my_drone.system_id = 1
vehicle.ardrone.custom.my_drone.model_name = AR.drone 2.0
vehicle.ardrone.custom.my_drone.serial_number = 1
```

And these lines to the second one:

```
vehicle.ardrone.custom.my_drone.system_id = 1
vehicle.ardrone.custom.my_drone.model_name = AR.drone 2.0
vehicle.ardrone.custom.my_drone.serial_number = 2
```

Of course, you need to make sure that network connection between UCS and computers with VSMs exist.

1.2 Mission execution specifics

- Fail-safe settings in mission properties are ignored.

ArDrone has the following harcoded fail-safe settings:

Condition	Behavior	Notes
On GPS signal loss	Unknown	Has not been tested. Should be Wait or Land.
On WiFi signal loss	Continue mission	
On low battery	Land	

Mission waypoint actions supported by ArDrone:

Flight plan element / action	Support	Notes
Camera control	No	
Camera trigger	No	
Yaw	Yes	
Land	Yes	
Panorama	No	
Takeoff	Yes	
Wait	Yes	
Camera by time	No	
Camera by distance	No	

Please carefully check your flight plan for maximum distance from ground station. It is worth not to fly far (maximum 30-50 meters) from a GS because the AR.Drone is equipped with a short-range Wi-Fi.

In each route for AR.Drone you should always have landing points. In any flight plan a landing point should be near you or starting point. If you do not specify a landing point, the AR.Drone will loiter in air until empty battery. Landing point ensures that AR.Drone will automatically return and land at this point even if connection is lost between the drone and a GS.

It might be necessary to wait several minutes after the AR.Drone is powered on in order to have strong GPS signal. It was noticed that the AR.Drone positioning quality is very poor if the flight is started instantly after powering on the drone.

ArDrone will not accept missions where any waypoint is lower than ground elevation at base location. It is not able to fly below starting point.

1.3 Command execution specifics

Command	Support	Notes
ARM	No	
DISARM	No	
AUTOMODE	Yes	Takeoff and start mission execution. If there is no mission on the drone it stays on the ground.
MANUALMODE	No	
CLICK & GO	No	
JOYSTICK	No	
HOLD	Yes	Pause mission execution. The drone will loiter at its current position. It is possible to disconnect GS PC and connect by a native application for manual control.
CONTINUE	Yes	Continue mission execution if previously paused by HOLD.
RETURNHOME	No	
TAKEOFF	No	
LAND	Yes	
EMERGENCYLAND	Yes	Beware! Drone will fall down immediately.
CAMERA_TRIGGER	No	Trigger camera shutter

1.4 Command visibility

UGCS Client shows command buttons in different shades. Highlighted buttons suggest recommended commands, depending on vehicle current state. You can always press all buttons regardless of shade.

Command visibility:

State	Button highlighted	Button shaded
Armed	LAND, EMERGENCYLAND, HOLD, CONTINUE	AUTO
Disarmed	AUTO, CONTINUE, HOLD	LAND, EMERGENCYLAND

Armed state inconsistency:

Note

Sometimes ArDrone will report itself as "Armed" even though it sits on the ground with motors off. It can happen after "LAND" command issued from the client. This is a flaw in ArDrone firmware and can be ignored. The state returns to "Disarmed" after a power cycle.

1.5 Telemetry information specifics

Nothing specific.

1.6 Video link

The AR.Drone has on-board HD camera and broadcasts the video stream via Wi-Fi. To watch the video in UgCS you should add the video stream on the video configuration page. Use this URL:

```
tcp://192.168.1.1:5555/
```

Currently it is supported only for UgCS client running on the computer which has direct Wi-Fi connection to AR.-Drone.

1.7 Configuration file

Default configuration file of the AR.Drone 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-ardrone.conf
  ```

- **On GNU/Linux:**

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

- **On Apple OS X:**

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

1.7.1 Common parameters

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

```
vehicle.ardrone
```

1.7.2 UDP connection configuration

Mandatory. At least one parameters set should be configured, otherwise VSM will not try to connect to the vehicle. See UDP connection configuration

Usually the AR.Drone has IP-address 192.168.1.1 assigned for its on-board network interface. The AR.Drone side UDP port is 14551, the VSM side should use 14550.

- **Example:**

```
vehicle.ardrone.detector.1.udp_local_address = 0.0.0.0
vehicle.ardrone.detector.1.udp_local_port = 14550
vehicle.ardrone.detector.1.udp_address = 192.168.1.1
vehicle.ardrone.detector.1.udp_port = 14551
```

1.7.3 Model name and serial number override

Optional.

- **Name:** vehicle.ardrone.custom.[name].system_id = [system id]
- **Name:** vehicle.ardrone.custom.[name].model_name = [model name]
- **Name:** vehicle.ardrone.custom.[name].serial_number = [serial number]
- **Description:** In UgCS each vehicle is identified by a unique combination of model name and serial number represented as text strings. By default, Ar.Drone vehicles are identified with a model name Ar.Drone and serial number equal with the Mavlink system id read from the vehicle. It can be overridden by these parameters, where [name] is an arbitrary vehicle name, [system id] is the original Mavlink system id which should be overridden, [model name] is a new model name to be visible to the UgCS, [serial number] is a new serial number to be visible to the UgCS.
- **Example:**

```
vehicle.ardrone.custom.my_drone.system_id = 2
vehicle.ardrone.custom.my_drone.model_name = AR.drone 2.0
vehicle.ardrone.custom.my_drone.serial_number = 123456
```

1.8 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.

1.8.1 UgCS server configuration

1.8.1.1 Listening address

Mandatory.

- **Name:** ucs.local_listening_address = [IP address]
- **Description:** Local TCP address to listen for incoming connections from UgCS server. Specify *0.0.0.0* to listen from all local addresses.
- **Example:** ucs.local_listening_address = 0.0.0.0

1.8.1.2 Listening port

Mandatory.

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

1.8.2 Logging configuration

1.8.2.1 Level

Optional.

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

1.8.2.2 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

1.8.2.3 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

1.8.2.4 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

1.8.3 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

1.8.4 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

1.9 Communication with vehicle

VSM can communicate with Vehicle over different communication channels

Currently supported channels:

- Serial port, see Serial port configuration for details.
- TCP link, see TCP connection configuration for details.
- UDP link, see UDP connection configuration for details.
- vsm-proxy (XBee), see Proxy configuration for details.

1.9.1 Serial port configuration

Optional. 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. [prefix] is unique for each VSM.

1.9.1.1 Port name

Optional.

- **Name:** [prefix].[port 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. [port index] is a arbitrary port indexing name.
- **Example:** vehicle.ardupilot.serial_port.1.name = /dev/ttyUSB[0-9]+|com[0-9]+
- **Example:** vehicle.ardupilot.serial_port.2.name = com42

1.9.1.2 Port baud rate

Optional.

- **Name:** [prefix].[port 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. [port index] is an arbitrary port indexing name.
- **Example:** vehicle.ardupilot.serial_port.1.baud.1 = 9600
- **Example:** vehicle.ardupilot.serial_port.1.baud.2 = 57600
- **Example:** vehicle.ardupilot.serial_port.2.baud = 38400

1.9.1.3 Excluded port name

Optional.

- **Name:** [prefix].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:** vehicle.ardupilot.serial_port.exclude.1 = /dev/ttyS.∗
- **Example:** vehicle.ardupilot.serial_port.exclude = com1

1.9.1.4 Serial port arbiter

Optional.

- **Name:** [prefix].use_serial_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:** vehicle.ardupilot.serial_port.use_serial_arbiter = no

1.9.2 TCP connection configuration

Optional. VSM which communicates with vehicles over TCP should define at least one network connection, otherwise VSM will not try to connect to vehicles. [prefix] is unique for each VSM.

1.9.2.1 IP-address for outgoing TCP connection

Optional.

- **Name:** [prefix].detector.[con index].address = [IP-address]
- **Description:** IP-address of vehicle to connect to. Typically used for vehicle simulators.
- **Example:** vehicle.ardupilot.detector.1.address = 10.0.0.111

1.9.2.2 remote TCP port

Optional.

- **Name:** [prefix].detector.[con index].tcp_port = [port number]
- **Description:** Remote port to connect to.
- **Example:** vehicle.ardupilot.detector.1.tcp_port = 5762

1.9.3 UDP connection configuration

Optional. VSM which communicates with vehicles via network should define at least one network connection, otherwise VSM will not try to connect to vehicles. [prefix] is unique for each VSM.

1.9.3.1 Local IP-address for UDP

Optional.

- **Name:** [prefix].detector.[con index].udp_local_address = [IP-address]
- **Description:** Local IP-address to listen for incoming UDP packets on. Specify 0.0.0.0 if you want to listen on all local addresses.
- **Example:** vehicle.ardrone.detector.1.udp_local_address = 0.0.0.0

1.9.3.2 Local UDP port

Optional.

- **Name:** [prefix].detector.[con index].udp_local_port = [port number]
- **Description:** Local UDP port to listen for incoming packets on.
- **Example:** vehicle.ardrone.detector.1.udp_local_port = 14550

1.9.3.3 Remote IP-address for UDP

Optional.

- **Name:** [prefix].detector.[con index].udp_address = [IP-address]
- **Description:** Remote IP-address to send outgoing UDP packets to.
- **Example:** vehicle.ardrone.detector.1.udp_address = 192.168.1.1

1.9.3.4 Remote UDP port

Optional.

- **Name:** [prefix].detector.[con index].udp_port = [port number]
- **Description:** Remote UDP port to send outgoing packets to.
- **Example:** vehicle.ardrone.detector.1.udp_port = 14551

1.9.4 Proxy configuration

Optional. 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.

1.9.4.1 IP-address for proxy

Optional.

- **Name:** [prefix].tcp.[con index].proxy = [IP-address]
- **Description:** IP-address to connect proxy to. Specify local or remote address.
- **Example:** vehicle.ardupilot.tcp.1.proxy = 127.0.0.1

1.9.4.2 TCP port for proxy

Optional.

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

2 Disclaimer

DISCLAIMER OF WARRANTIES AND LIMITATIONS ON LIABILITY.

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

(b) SPH ENGINEERING 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.