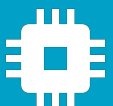# Aerobits

Subsystems for the
UAS intergration into
the airspace

Receiver AERO

Data sheet & User manual

Introduction

AERO™ belongs to the class of the smallest ADS-B receivers on market and has been developed for civil and commercial Unmanned Aircraft Systems. It is especially dedicated to UAS controllers supporting the MAVLink protocol. **AERO**™ operates on 1090MHz and allows to track air traffic (equipped with ADS-B technology) in the vicinity of 100km from UAS.

The goal is to ensure a safe separation between manned and unmanned aircraft. **AERO**™ opens the way to the implementation of the Detect and Avoid algorithms, supporting the integration of UAS into the airspace.

For more information please contact: support@aerobits.pl.

Main features

- Real-time aircraft tracking on 1090MHz
- Patented FPGA-In-The-Loop^{TM} technology with the capability of receiving thousands of frames per second
- High sensitive front-end with range up to 100km (300km with external 1dBi antenna)
- Programming via AT commands
- Simple plug&play integration

Date: 2023-06-07 Firmware version: 2.38.0 Document ref: 07a-v1.1.8

This document is subject to change without notice. For technical questions, contact: support@aerobits.pl

Contents

Date: 2023-06-07   Firmware version: 2.38.0 Document ref: 07a-v1.1.8

This document is subject to change without notice. For technical questions, contact: support@aerobits.pl

1 Technical parameters

1.1 Basic technical information

Parameter	Description	Typ.	Unit
Frequency	ADS-B	1090	MHz
Sensitivity	ADS-B	-85	dBm
Max. input		+10	dBm
ESD protection	RF part only		-
MAVLink (baud)		115200	bps
AERO (baud)	AT commands	115200	bps
Main connector	SM06B-GHS-TB(LF)(SN)		-
Antenna connector	RF-IPX125-1G-AU		-
Temperature range	Operating temperature	-30 to +85	°C
Storage temperature	Optimal storage temperature	-5 to +40	°C
Dimension		27.0 x 14.0 x 5.5	mm
Weight (with antenna)		2.8	grams

Table 1: General technical parameters.

1.2 Electrical specification

1.2.1 Basic electrical parameters

Parameter	Value
Input voltage	5 V
Current consumption	70 mA

Table 2: General electrical parameters.
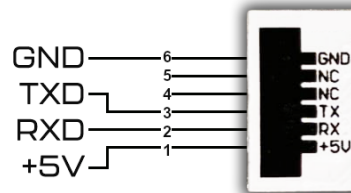
Figure 1: Appendant drawing of Receiver AERO .

1.2.2 PIN definition

PIN	Color	Name	Function
1	-	+5 V	Power supply
2	-	RX	MAVLink, AERO RXD
3	-	TX	MAVLink, AERO TXD
4	-	NC	Not connected
5	-	NC	Not connected
6	-	GND	Ground

Table 3: Pin definition.

1.2.3 LED indicators

LED	Color	Function
POWER	Green	Power supply indicator
ADS-B	White	Frame detection / receive indicator

Table 4: LED indicators.

1.3 Mechanical specification

1.3.1 Mechanical parameters

Parameter	Value
Dimensions	27.0 x 14.0 x 5.5mm
Weight	3 g

Table 5: Mechanical parameters of Receiver AERO

1.3.2 Dimensions
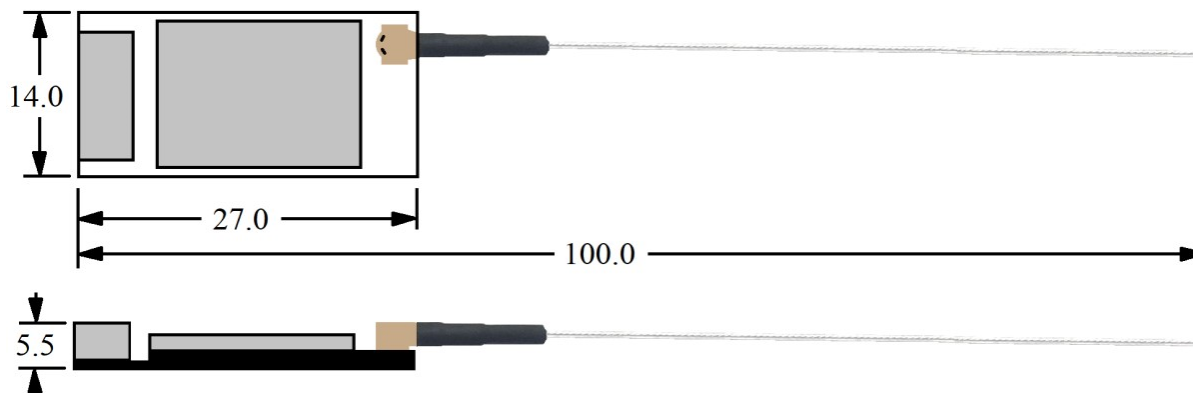
Figure 2: Dimensions of Receiver AERO

1.3.3 Connectors

Connector	Type	Example
Main	Installed on board	SM06B-GHS-TB(LF)(SN)
	Mating connector	GHR-06V-S
	Pins	SSHL-002T-P0.2
Antenna	Installed on board	RF-IPX125-1G-AU
	Mating connector	GSM-IPX or GSM-IPX/SMA-1G-150

Table 6: Connectors

2 Principle of operation

During work module goes through multiple states. In each state operation of the module is different. Each state and each transition is described in paragraphs below.
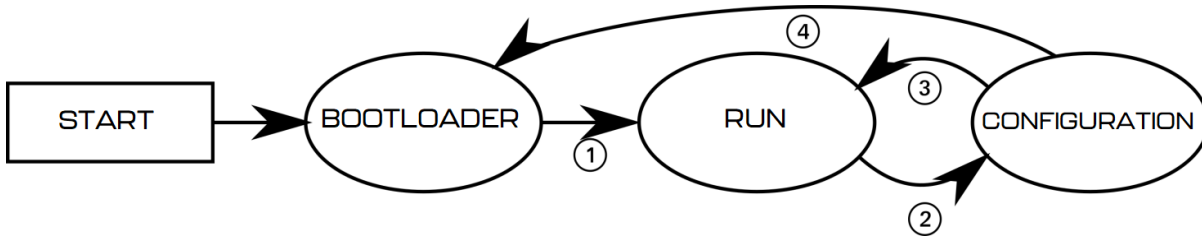

Figure 3: State machine of Receiver AERO

2.1 States of operation

2.1.1 BOOTLOADER state

This is an initial state of Receiver AERO after restart. Firmware update is possible here. Typically module transits automatically to RUN state. It is possible to lock module in this state (prevent transition to RUN state) using one of BOOTLOADER triggers. UART baud is constant and is set to 115200bps. After powering up module, it stays in this state for up to 3 seconds. If no BOOTLOADER trigger is present, module will transit to RUN state. Firmware up-grade is possible using Micro ADS-B App software. For automated firmware upgrading scenarios, aerobits_updater software is available. To acquire this program please contact: support@aerobits.pl.

2.1.2 RUN state

In this state module is working and receiving the data from aircrafts. It uses selected protocol to transmit received and decoded data to the host system. In this state of operation module settings are loaded from non-volatile internal memory, including main UART interface's baud.

2.1.3 CONFIGURATION state

In this mode change of stored settings is possible. Operation of the module is stopped and baud is set to fixed 115200bps. Change of settings is done by using AT-commands. Changes to settings are stored in non-volatile memory on exiting this state. Additional set of commands is also available in this state, allowing to e.g. reboot module into BOOTLOADER state, check serial number and firmware version. It is possible to lock module in this state (similarly to BOOTLOADER) using suitable command.

2.2 Transitions between states

For each of state transitions, different conditions must be met, which are described below. Generally, the only stable state is RUN. Module always tends to transit into this state. Moving to other states requires host to take some action.

2.2.1 BOOTLOADER to RUN transition

BOOTLOADER state is semi-stable: the module requires additional action to stay in BOOTLOADER state. The transition to RUN state will occur automatically after short period of time if no action will be taken. To prevent transition from BOOTLOADER state, one of following actions must be processed:

- Send `AT+LOCK=1` command while device is in BOOTLOADER state (always after power on for up to 3s)

- Send `AT+REBOOT_BOOTLOADER` command in CONFIGURATION state. This will move to BOOTLOADER state and will lock module in this state.

If none of above conditions are met, the module will try to transit into RUN state. Firstly it will check firmware integrity. When firmware integrity is confirmed, module will transit into RUN state, if not, it will stay in BOOTLOADER state.

To transit into RUN state:

- If module is locked, send `AT+LOCK=0` command

When module enters RUN mode it will send `AT+RUN_START` command.

2.2.2 RUN to CONFIGURATION transition

To transit from RUN into CONFIGURATION state, host should do one of the following:

- Send `AT+CONFIG=1` (using current baud).

When module leaves RUN state it sends `AT+RUN_END` message, then `AT+CONFIG_START` message on entering CONFIGURATION state. The former is sent using baud from settings, the latter always uses 115200bps baud.

2.2.3 CONFIGURATION to RUN transition

To transit from CONFIGURATION into RUN state, host should do one of the following:

- Send `AT+CONFIG=0` command.

When module leaves CONFIGURATION state it sends `AT+CONFIG_END` message, then `AT+RUN_START` message on entering RUN state. The former is always sent using 115200bps baud, the latter uses baud from settings.

2.2.4 CONFIGURATION to BOOTLOADER transition

To transit from CONFIGURATION into BOOTLOADER state, host should do one of the following:

- Send `AT+REBOOT_BOOTLOADER` command.
- Send `AT+REBOOT` and when module enters BOOTLOADER state, prevent transition to RUN state.

When entering the bootloader state, the module sends `AT+BOOTLOADER_START` .

3 UART configuration

Communication between module and host device is done using UART interface.

In CONFIGURATION and BOOTLOADER state transmission baud is fixed at 115200bps.

The UART interface uses settings as described in table 7.

UART Settings				
Parameter	Min.	Typ.	Max	Unit
Baud	-	115200	-	bps
Stop Bits Number	-	1	-	-
Flow Control	-	None	-	-
Parity Bit	-	None	-	-

Table 7: UART settings.

4 Settings

In RUN state, operation of the module is determined based on stored settings. Settings can be changed in CON-FIGURATION state using AT-commands. Settings can be written and read.

NOTE: New values of settings are saved in non-volatile memory when transitioning from CONFIGURATION to RUN state.

Settings are restored from non-volatile memory during transition from BOOT do RUN state. If settings become corrupted due to memory fault, power loss during save, or any other kind of failure, the settings restoration will fail, loading default values and displaying the `AT+ERROR (Settings missing, loaded default)` message as a result. This behavior will occur for each device boot until new settings are written by the user.

4.1 Write settings

After writing a new valid value to a setting, an `AT+OK` response is always sent.

```
AT+SETTING=VALUE
```
For example `AT+PROTOCOL=1`
Response: `AT+OK`

4.2 Read settings

```
AT+SETTING?
```
For example: `AT+PROTOCOL?`
Response: `AT+PROTOCOL=1`

4.3 Settings description

```
AT+SETTING=?
```
For example: `AT+PROTOCOL=?`
Response:

```
Setting: PROTOCOL
      Description: Selected protocol (0: NONE, 2: CSV, 3: MAVLINK)
      Type: Integer decimal
      Range (min.): 0
      Range (max.): 5
      Is preserved: 1
      Is restart needed: 0
```

4.4 Errors

Errors are reported using following structure:
`AT+ERROR (DESCRIPTION)`
DESCRIPTION is optional and contains information about error.

4.5 Command endings

Every command must be ended with one of the following character sequences: "\n", "\r" or "\r\n". Commands without suitable ending will be ignored.

4.6 Uppercase and lowercase

All characters (except preceding `AT+`) used in command can be both uppercase and lowercase, so following commands are equal:

```
AT+PROTOCOL?
AT+pRoToCoL?
```

NOTE: This statement is true in configuration state, not in bootloader state. in bootloader state all letters must be uppercase.

4.7 Available settings

Setting	Min	Max	Def	Comment
PROTOCOL	0	6	2	Selected protocol. Not all values are valid for all devices. 0 - None 2 - CSV (AERO) 3 - MAVLink
SUBPROTOCOL	0	0	0	Reserved for future use

Table 8: Settings

4.8 Example

As an example, to switch Receiver AERO module to CSV protocol, one should send following commands. "<<" indicates command sent to module, ">>" is a response.

```
<< AT+CONFIG=1\r\n
>> AT+OK\r\n
<< AT+PROTOCOL=2\r\n
>> AT+OK\r\n
>> AT+OK\r\n
<< AT+CONFIG=0\r\n
```

5 Commands

Apart from settings, module supports set of additional commands. Format of this commands are similar to those used for settings, but they do not affect operation of module in RUN state.

5.1 Commands in BOOTLOADER and CONFIGURATION state

5.1.1 AT+LOCK

`AT+LOCK=1` - Set lock to enforce staying in BOOTLOADER or CONFIGURATION state
`AT+LOCK=0` - Remove lock
`AT+LOCK?` - Check if lock is set

5.1.2 AT+BOOT

`AT+BOOT?` - Check if module is in BOOTLOADER state

Response:

`AT+BOOT=0` - module in CONFIGURATION state
`AT+BOOT=1` - module in BOOTLOADER state

5.2 Commands in CONFIGURATION state

5.2.1 AT+CONFIG

`AT+CONFIG=0` - Transition to RUN state.
`AT+CONFIG?` - Check if module is in CONFIGURATION state.

Response:

`AT+CONFIG=0` - module in RUN state
`AT+CONFIG=1` - module in CONFIGURATION state

5.2.2 AT+SETTINGS?

`AT+SETTINGS?` - List all settings. Example output:

```
AT+PROTOCOL=2
AT+SUBPROTOCOL=0
AT+BAUDRATE=0
```

5.2.3 AT+HELP

`AT+HELP` - Show all settings and commands with descriptions. Example output:

```
SETTINGS:
AT+PROTOCOL=2          [Selected protocol (0: NONE, 2: CSV, 3: MAVLINK)]
AT+SUBPROTOCOL=0       [Subprotocol of selected protocol]
COMMANDS:
AT+HELP                [Show this help]
AT+TEST                [Responds "AT+OK"]
AT+SETTINGS_DEFAULT    [Load default settings]
AT+REBOOT              [Reboot system]
```

5.2.4 AT+SETTINGS_DEFAULT

`AT+SETTINGS_DEFAULT` - Set all settings to their default value.

5.2.5 AT+SERIAL_NUMBER

`AT+SERIAL_NUMBER?` - Read serial number of module.

Response:

`AT+SERIAL_NUMBER=07-0001337`

5.2.6 AT+FIRMWARE_VERSION

`AT+FIRMWARE_VERSION?` - Read firmware version of module.

Response:

`AT+FIRMWARE_VERSION=10101017(May 11 2018)`

5.2.7 AT+REBOOT

`AT+REBOOT` - Restart module.

5.2.8 AT+REBOOT_BOOTLOADER

`AT+REBOOT_BOOTLOADER` - Restart module to BOOTLOADER state.

NOTE: This command also sets lock.

5.3 Commands in RUN state

`AT+CONFIG=1` - transition to CONFIGURATION state.

NOTE: This command also sets lock.

Date: 2023-06-07   Firmware version: 2.38.0 Document ref: 07a-v1.1.8

This document is subject to change without notice. For technical questions, contact: support@aerobits.pl

6 Protocols

6.1 CSV protocol (AERO)

CSV protocol is simple text protocol, that allows fast integration and analysis of tracked aircrafts. CSV messages start with '#' character and ends with "\r\n" characters. There are following types of messages:

1. ADS-B Aircraft message,

2. Statistics message.

NOTE: In future versions, additional comma-separated fields may be introduced to any CSV protocol message, just before CRC field, which is guaranteed to be at the end of message. All prior fields are guaranteed to remain in same order.

6.1.1 CRC

Each CSV message includes CRC value for consistency check. CRC value is calculated using standard CRC16 algorithm and its value is based on every character in frame starting from '#' to last comma ',' (excluding last comma). After calculation, value is appended to frame using hexadecimal coding. Example function for calculating CRC is shown below.

```
uint16_t crc16(const uint8_t* data_p, uint32_t length){
    uint8_t x;
    uint16_t crc = 0xFFFF;
    while (length--){
        x = crc>>8 ^ *data_p++;
        x ^= x>>4;
         crc = (crc<<8) ^ ((uint16_t)(x<<12)) ^ ((uint16_t)(x<<5)) ^ ((uint16_t)x);
    }
    return swap16(crc);
}
```

6.1.2 ADS-B Aircraft message

This message describes state vector of aircraft determined from ADS-B messages and is sent once per second. The message format is as follows:

```
#A:ICAO,FLAGS,CALL,SQ,LAT,LON,ALT_BARO,TRACK,
VELH,VELV,SIGS,SIGQ,FPS,NICNAC,ALT_GEO,ECAT,CRC\r\n
```

#A	Aircraft message start indicator	Example value
ICAO	ICAO number of aircraft (3 bytes)	3C65AC
FLAGS	Flags bitfield, see table 10	1
CALL	Callsign of aircraft	N61ZP
SQ	SQUAWK of aircraft	7232
LAT	Latitude, in degrees	57.57634
LON	Longitude, in degrees	17.59554
ALT_BARO	Barometric altitude, in feets	5000
TRACK	Track of aircraft, in degrees [0,360)	35
VELH	Horizontal velocity of aircraft, in knots	464
VELV	Vertical velocity of aircraft, in ft/min	-1344
SIGS	Signal strength, in dBm	-92
SIGQ	Signal quality, in dB	2
FPS	Number of raw MODE-S frames received from aircraft during last second	5
NICNAC	NIC/NAC bitfield, see table 11 (v2.6.0+)	31B
ALT_GEO	Geometric altitude, in feets (v2.6.0+)	5000
ECAT	Emitter category, see table 12 (v2.7.0+)	14
CRC	CRC16 (described in CRC section)	2D3E

Table 9: Descriptions of ADS-B message fields.

Value	Flag name	Description
0x0001	PLANE_ON_THE_GROUND	The aircraft is on the ground
0x0002	PLANE_IS_MILITARY	The aircraft is military object
0x0100	PLANE_UPDATE_ALTITUDE_BARO	During last second, barometric altitude of this aircraft was updated
0x0200	PLANE_UPDATE_POSITION	During last second, position (LAT & LON) of this aircraft was updated
0x0400	PLANE_UPDATE_TRACK	During last second, track of this aircraft was updated
0x0800	PLANE_UPDATE_VELO_H	During last second, horizontal velocity of this aircraft was updated
0x1000	PLANE_UPDATE_VELO_V	During last second, vertical velocity of this aircraft was updated
0x2000	PLANE_UPDATE_ALTITUDE_GEO	During last second, geometric altitude of this aircraft was updated

Table 10: ADS-B message Flags description.

The NIC/NAC bitfield is transmitted in big endian hexadecimal format without leading zeros. Table 11 describes its bitfield layout. The meaning of NIC/NAC indicators is exactly the same as described in ED-102A.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				NAC_p				NAC_v			NIC_{baro}	NIC			

Table 11: Structure of NIC/NAC bitfield in CSV protocol.

Below is a list of emitter category values returned in ECAT field.

Date: 2023-06-07   Firmware version: 2.38.0 Document ref: 07a-v1.1.8

This document is subject to change without notice. For technical questions, contact: support@aerobits.pl

ECAT value	Description
0	Unknown.
1	Light (below 15500 lbs.).
2	Small (15500 - 75000 lbs.).
3	Large (75000 - 300000 lbs.).
4	High-Vortex Large (aircraft such as B-757).
5	Heavy (above 300000 lbs.).
6	High performance (above 5g acceleration and above 400 knots).
7	Rotorcraft.
8	Reserved.
9	Glider, Sailplane.
10	Lighter-Than-Air.
11	Parachutist, Skydiver.
12	Ultralight, hang-glider, paraglider.
13	Reserved.
14	Unmanned Aerial Vehicle.
15	Space, Trans-atmospheric Vehicle.
16	Reserved.
17	Surface Vehicle - Emergency Vehicle.
18	Surface Vehicle - Service Vehicle.
19	Point Obstacle (includes Tethered Ballons).
20	Cluster obstacle.
21	Line obstacle.

Table 12: ADS-B emitter category values in CSV protocol.

If data of any field of frame is not available, then it is transmitted as empty. For example:

```
#A:4CA948,300,,2122,52.99750,13.76526,37000,169,442,0,-93,4,3,,6F1C\r\n
#A:424313,,,2362,52.43431,14.84535,37000,65,456,0,-91,6,0,,6843\r\n
```

NOTE: SIGS and SIGQ fields are updated based on raw MODE-S frames. They are calculated from frames received in last second. If there were no receiver frames (FPS=0), those fields will not be updated.

NOTE: SIGS is measured based on analog RF signal. This signal has DC offset of about 700mV.

6.1.3 Statistics message

This message contains some useful statistics about operation of module. Format of that frame is shown below:

```
#S:CPU,RES,RES,FPSS,RES,RES,CRC
```

#S	Statistics message start indicator	Example
CPU	CPU load in %	12.1
RES	Reserved for future use	-
RES	Reserved for future use	-
FPSS	Number of MODE-S frames received in last second	3
RES	Reserved for future use	-
RES	Reserved for future use	-
CRC	CRC16 (described in CRC section)	2D3E

Table 13: Statistics message fields.

Date: 2023-06-07   Firmware version: 2.38.0 Document ref: 07a-v1.1.8

This document is subject to change without notice. For technical questions, contact: support@aerobits.pl

6.2 MAVLink protocol

Receiver AERO can be switched to use MAVLink protocol. This can be achieved by altering PROTOCOL set-ting. When MAVLink protocol is used, module is sending list of aircrafts every second. MAVLink messages have standarized format, which is well described on official protocol webpage (mavlink.io/en/messages).

6.2.1 ADS-B Aircraft message

Aircrafts are encoded using ADSB_VEHICLE message (mavlink.io/en/messages/common.html#ADSB_VEHICLE). MAVLink message contains several data fields which are described below.

Field Name	Type	Description
ICAO_address	uint32_t	ICAO address
lat	int32_t	Latitude, expressed as degrees * 1E7
lon	int32_t	Longitude, expressed as degrees * 1E7
altitude	int32_t	Barometric/Geometric Altitude (ASL), in millimeters
heading	uint16_t	Course over ground in centidegrees
hor_velocity	uint16_t	The horizontal velocity in centimeters/second
ver_velocity	uint16_t	The vertical velocity in centimeters/second, positive is up
flags	uint16_t	Flags to indicate various statuses including valid data fields
squawk	uint16_t	Squawk code
altitude_type	uint8_t	Type from ADSB_ALTITUDE_TYPE enum
callsign	char[9]	The callsign, 8 chars + NULL
emitter_type	uint8_t	Type from ADSB_EMITTER_TYPE enum
tslc	uint8_t	Time since last communication in seconds

Table 14: MAVLink ADSB_VEHICLE message description

The ADS-B vehicle may transmit barometric, as well as geometric altitude. The SUBPROTOCOL setting allows for toggling altitude transmit priority:

- When set to 0, altitude field will be filled with geometric altitude first. If not available, barometric altitude will be used.

- When set to 1, barometric altitude wil be preferred.

7 Quick start

NOTE: Not all Mission Planner versions display ADS-B signals correctly. Make sure that your version of Mission Planer and Pixhawk is up to date.

7.1 Pixhawk update

When installing via Mission Planner, disconnect by clicking the button, then select the appropriate firmware for your device in Install Firmware Legacy tab and follow the instructions. During installation by another method - compatible release is the newest version of **ArduPilot Flight Stack**.
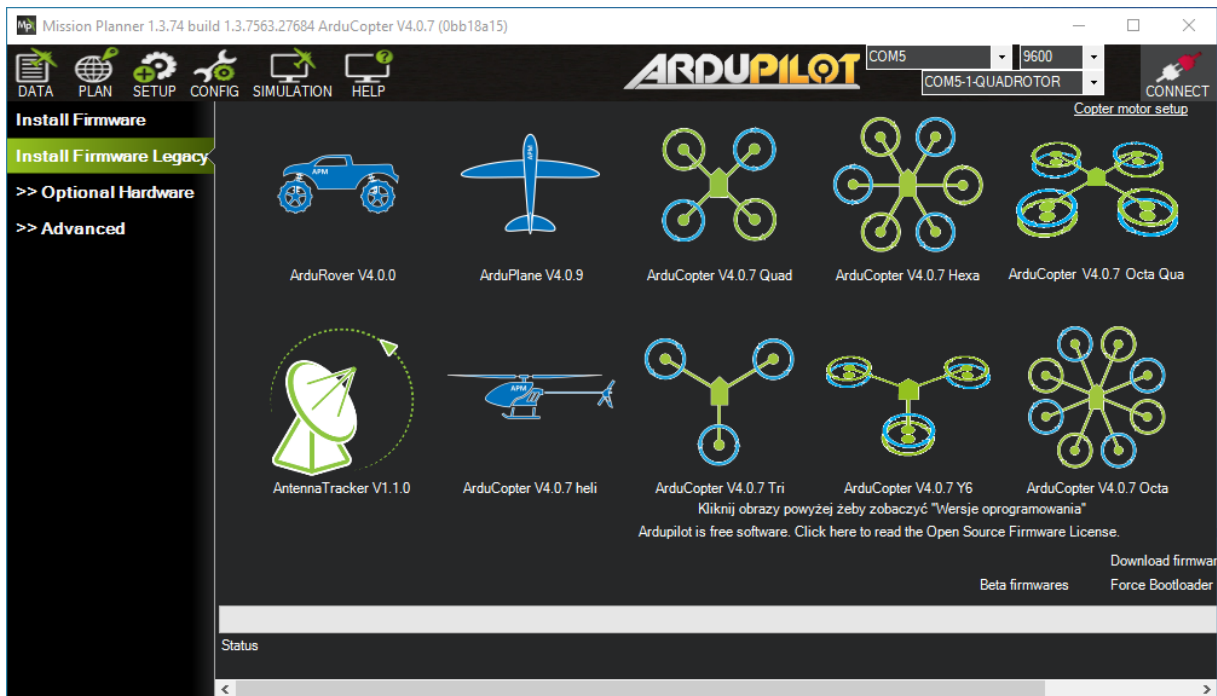
7.2 Mission Planner

Five steps to integrate **Aero** with Pixhawk4:

1. With the power turned off, connect Aero to Pixhawk4 using a standard telemetry cable. The following settings apply to the installation on the TELEM2 port.

2. Connect the USB cable between Pixhawk4 and your PC and run Mission Planner.

3. Connect to Pixhawk4 by clicking "Connect", then go to the "CONFIG" tab.
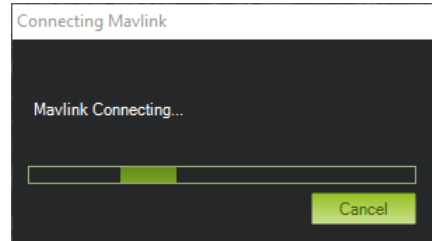
4. In the menu, go to "Full Parameter Tree" and set the following parameters:

a.	ADSB	->	ADSB_ENABLE	1
b.	SERIAL2	->	SERIAL2_BAUD	115
		->	SERIAL2_PROTOCOL	2
c.	SR0	->	SR0_ADSB	range 1 to 50 Hz

NOTE: Remember to send the changed settings to the controller by clicking "Write params".

5. **Restart your Pixhawk** and go to the main view. If there is air traffic in your area, you should see it on the map.

For more information visit: ardupilot ADS-B documentation.

7.3 QGroundControl

Mission Planer is a program designed for the Windows platform. QGroundControl is an alternative to Mission Planer with similar functionality. First steps are the same for both environments.

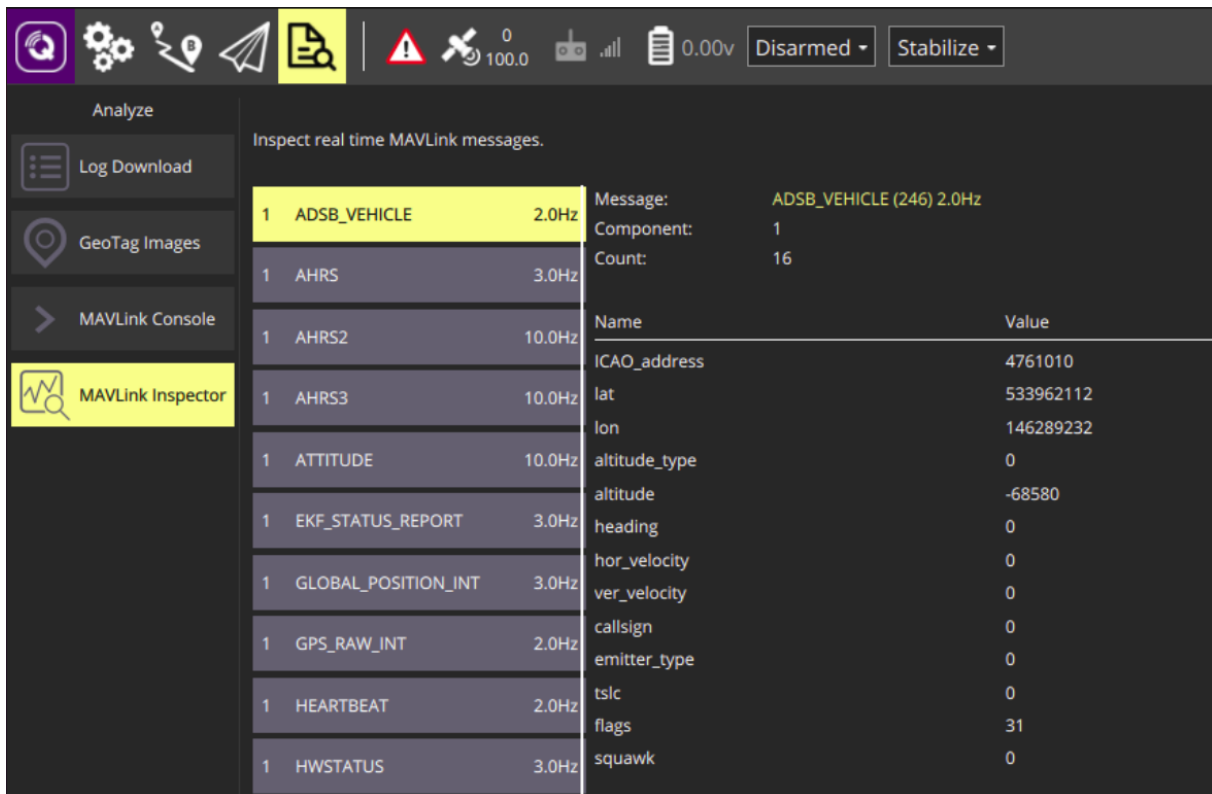After connection device to Pixhawk4, the program should detect it.

Set the parameters the same as in Mission Planer:

      a.   ADSB   ->   ADSB_ENABLED
      b.   SERIAL2   ->   SERIAL2_BAUD   115200
                               ->   SERIAL2_PROTOCOL   MAVLink2
      c.   SR0   ->   SR0_ADSB   range 1 to 50 Hz
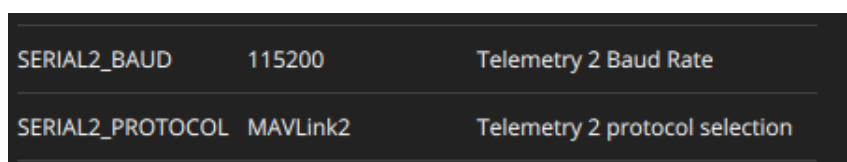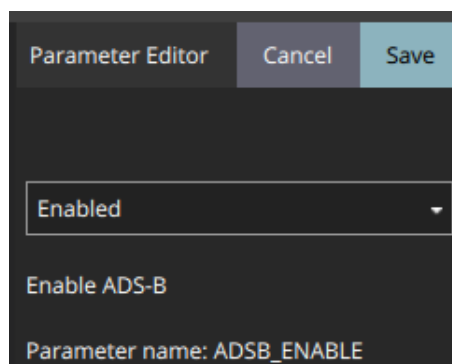
To make sure that the device receives the ADS-B signal correctly, you can check the MAVLink Inspector tab. Pa-

rameters like ADSB_VEHICLE and HEARTBEAT should be greater than 0 and count of received frames shoud have increasing tendency.

For more information visit: QGroundControl documentation.

8 General information

8.1 Module installation

There is a high concentration of various electronic systems on a small area at UAS. Try to keep as much separation between **AERO** and other devices, especially radio ones.

Despite the high robustness of **AERO** to jamming, try to install the antenna away from other on-board systems.

8.2 Antenna

Aero is supplied with an installed antenna. It allows you to save weight, but requires installation of the module in the sky view. If you want to integrate the module into the UAS interior, you can use an adapter such as U.FL ->   SMA and install an external antenna. With a correct selection of the antenna you can get a much larger range.

8.3 MAVLink vs. AERO protocol

AERO is based on OEM TT-SC1b module. The default is in MAVLink protocol mode, which is a binary protocol. If you want to use the module to work with another system, it is possible to switch the protocol to AERO, which has the ASCII representation. Details of the module programming can be found on the website.

Please read carefully

Information contained in this document is provided solely in connection with Aerobits products. Aerobits reserves the right to make changes, corrections, modifications or improvements to this document, and to products and services described herein at any time, without notice. All Aerobits products are sold pursuant to our own terms and conditions of sale. Buyers are solely responsible for the choice, selection and use of the Aerobits products and services described herein, and Aerobits assumes no liability whatsoever, related to the choice, selection or use of Aerobits products and services described herein. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services, it shall not be deemed a license granted by Aerobits for use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering use, in any manner whatsoever, of such third party products or services or any intellectual property contained therein.

UNLESS OTHERWISE SET FORTH IN AEROBITS TERMS AND CONDITIONS OF SALE, AEROBITS DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO USE AND/OR SALE OF AEROBITS PRODUCTS INCLUDING, WITHOUT LIMITATION, IMPLIED WARRANTIES OF MERCHAN-TABILITY, FITNESS FOR A PAR-TICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGE-MENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. UNLESS EXPRESSLY APPROVED IN WRITING BY AN AUTHORIZED AEROBITS REPRESENTATIVE, AEROBITS PRODUCTS ARE NOT RECOMMENDED, AUTHORIZED OR WARRANTED FOR USE IN LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS, NOR IN PRODUCTS OR SYSTEMS WHERE FAILURE OR MALFUNCTION MAY RESULT IN PERSONAL INJURY, DEATH, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE.