Subsystems for the
UAS integration into
the airspace

GS2L

Data sheet - User manual

Contents

1 Introduction

GS with Linux station is an ADS-B and FLARM Omni-directional receiver station with BLE/Wi-Fi RemoteID receiver and also multi-constellation GNSS sensor on board to provide best accuracy. LTE connectivity allows usage in all LTE/4G rich environments without the need for any additional cabling to send data. It has been designed to allow quick and easy assembly, enclosed in IP67 case for high weather condition resistance. Device comes with all necessary cables and antennas for straightforward installation.

Single-board computer with Linux operating system gives incredible expansion possibilities, much higher computing power and memory, way easier updates, quicker debugging and development times.

Supply voltage is provided using PoE Power over Ethernet, using regular ethernet cable between PoE power supply and GS Linux. Depending on cable quality range can be up to 100m, giving it a tremendous advantage over USB. At the same time it can still transfer data just like a regular ethernet cable, providing an alternative data connection.

It is a perfect solution for permanent installation in open areas for constant airspace monitoring and conducting VLOS/BVLOS operation where safety is critical.

Note:

The device to operate on FLARM frequency requires FLARM UAS license. The license can be obtained with the device from Aerobits upon purchase. FLARM library expire after year and must be updated with latest firmware.

Important:

Each firmware version becomes its own documentation. This document is relevant for firmware version 1.14.4. If your firmware version is different please find relevant documentation on our website aerobits.pl.

1.1 Applications

- **Airports and critical infrastructure**
- **Nationwide traffic management systems (manned and unmanned)**
- **Perfect solution for local airfields**
- **U-Space and UTM systems**
- **Ground Network air traffic surveillance systems**
- **Network based Remote Identification (central monitoring)**

For more information please contact support@aerobits.pl.

2 Technical parameters

2.1 Basic technical information

Table 1: General technical parameters

Parameter	Description	Typ.	Unit
First Band	ADS-B	1090	MHz
Second Band	FLARM	868	MHz
Third Band	BLE	2400	MHz
Fourth Band	Wi-Fi	2400	MHz
Fifth Band	UAT	978	MHz
Sixth Band	GNSS	1575	MHz
Sensitivity (ADS-B)		-95	dBm
Sensitivity (FLARM)		-109	dBm
Sensitivity (BLE)		-103	dBm
Sensitivity (Wi-Fi)		-103	dBm
Sensitivity (UAT)		-110	dBm
Sensitivity (GNSS)		-167	dBm
Ethernet (RJ45)	Standard Ethernet 10/100		
LTE Cat. 1	Data transport layer (global bands)		

2.2 Electrical specification

2.2.1 Power supply

Table 2: Power supply of GS2L

Parameter	Value
Power connector	Standard ethernet connector (power supply and optionally network) and Bulgin Buccaneer 400 Series connector (power supply PX0412/03P)
Power consumption	3.5 W
Power supply	100 - 240 VAC with PoE supply unit or 24 VDC with AC/DC converter

2.3 Mechanical specification

2.3.1 Mechanical parameters

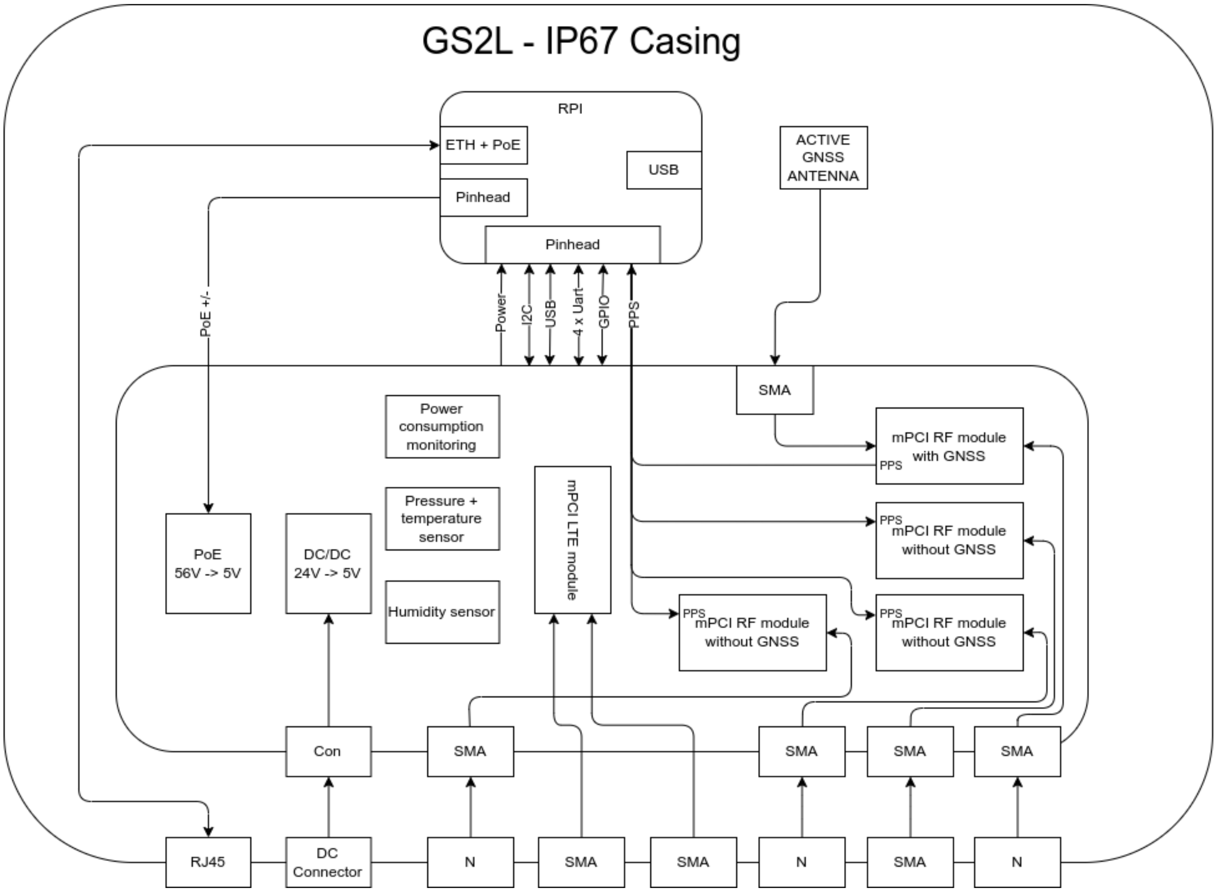
Table 3: Mechanical parameters of GS2L

Parameter	Value
Dimensions	272 x 276 x 96 mm
Weight	1.44 kg (Module without cables and antennas)

3 GS2L customization

3.1 Diagram

Diagram contains all possible configurations of GS2L, customer can configure ground station on their own.



3.2 Variants

GS2L has multiple configurations, the customer can set up a ground station with several RF modules, up to 4 maximum. Only one module is required to have a GNSS module on board, the rest of the RF modules will receive position information based on the first one. See recommended configuration below.

1. GNSS + RID
2. GNSS + ADS-B + FLARM
3. GNSS + ADS-B + UAT
4. GNSS + RID + ADS-B + FLARM
5. GNSS + RID + ADS-B + UAT

3.3 Additional kits

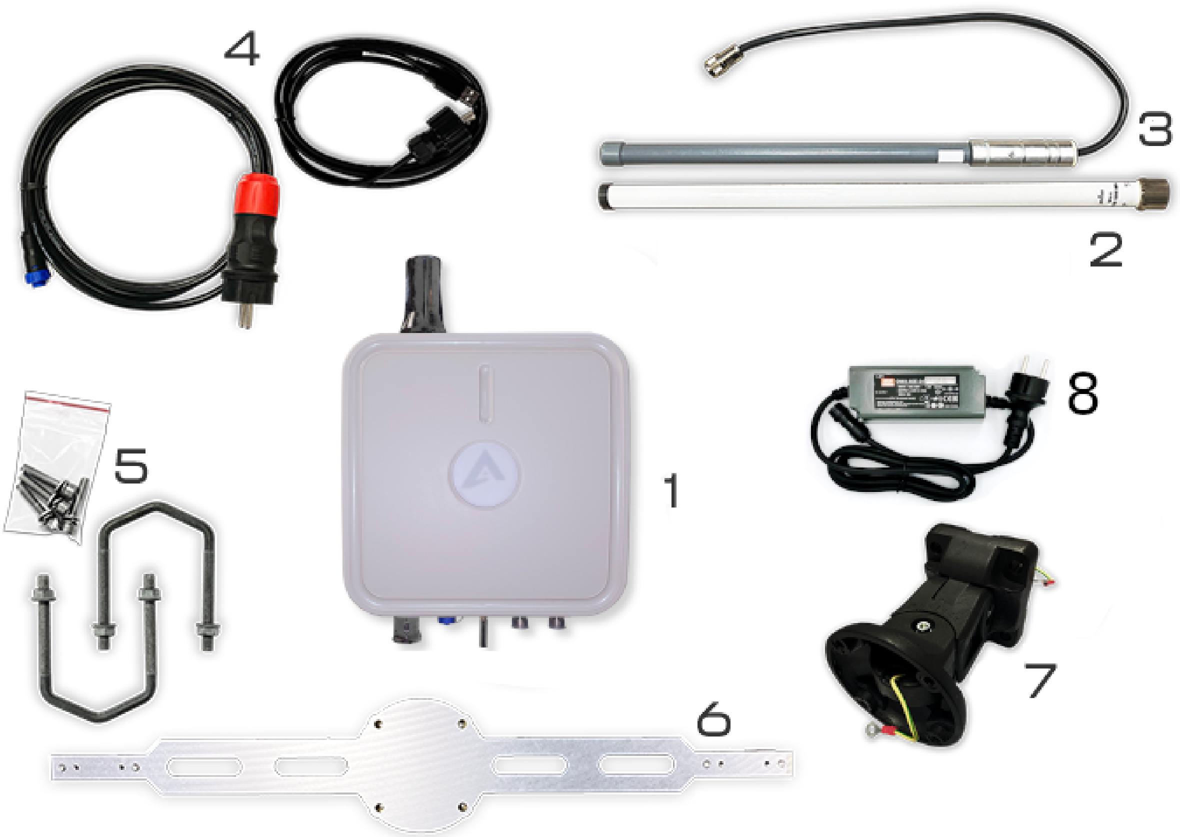
GS2L has several additional kits such as power delivery, antennas, cables etc.. For example customer can order GS2L with GNSS + RID and DC power supply, so in this case additional is kit need with antennas (sector or omnidirectional), power cables, DC converter. Remember GS2L is sold separately without kits included. See list of available kits below.

1. DC Power delivery with cables and AC/DC converter
2. PoE Power delivery with cables and PoE AC/DC converter
3. Sector Antenna with RF cable
4. Omnidirectional Antenna with RF cable
5. Mobile tripod
6. Mobile box
7. Powerbank (with cables)
8. Bracket for 2 omnidirectional antennas

4 Quick start

4.1 Scope of delivery

1. Ground Station with Linux
2. ADS-B/FLARM antenna sector or omnidirectional (optional)
3. BLE/Wi-Fi antenna sector or omnidirectional (optional)
4. Power Supply Cables (optional)
5. Small assembly parts
6. Antenna's installation arm (only with dual omnidirectional antenna setup)
7. Distance bracket
8. Power Supply PoE or AC/DC converter (optional)



4.2 Installation process

4.2.1 Mounting with sector and omnidirectional antennas

1. Take the GS out of the box and place facing down - as shown on the picture.

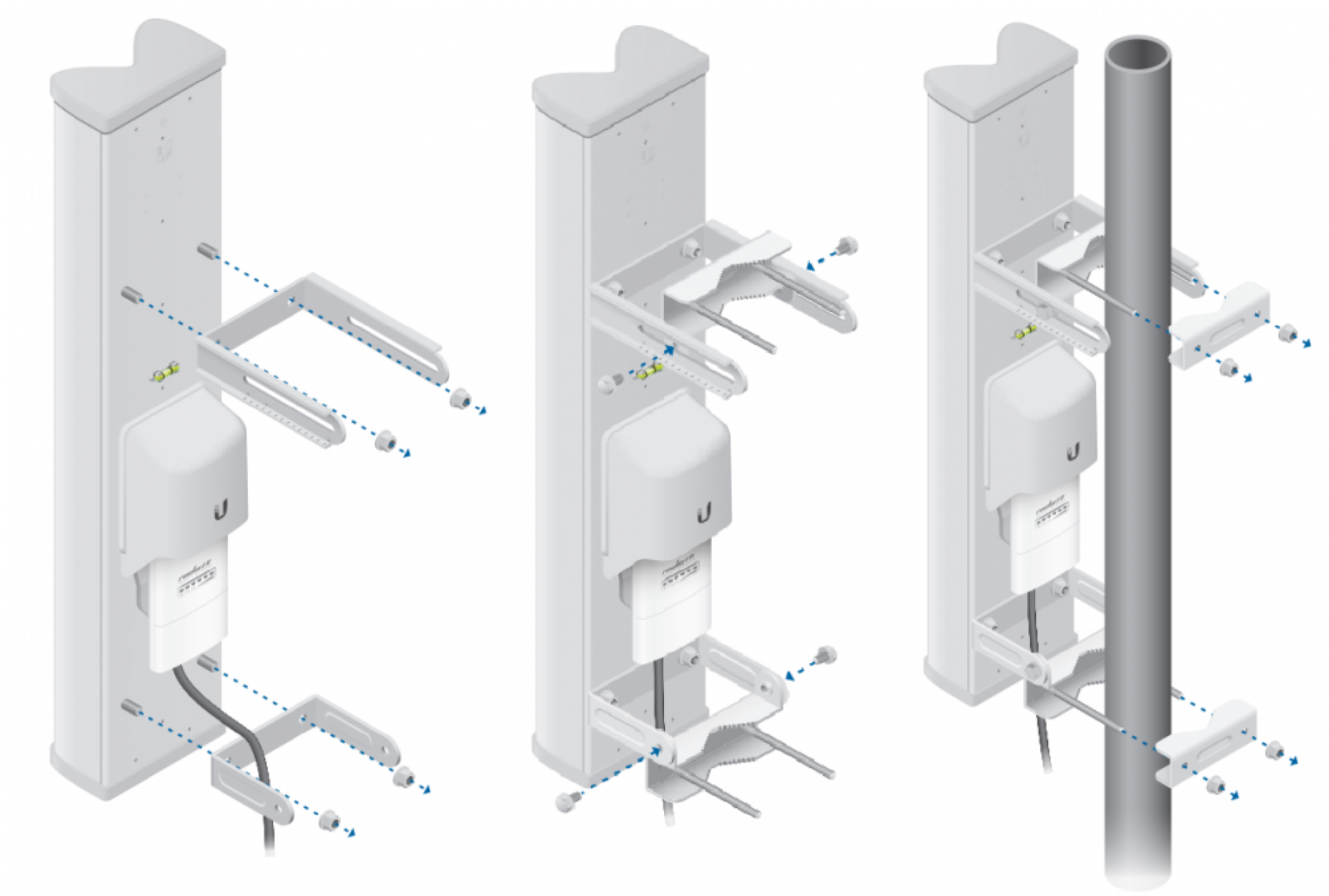
2. Mount black distance bracket with the protective earth conductor on the case.

Note:

It is important that the cable is connected to the appropriate hole, which is marked on the case with following electrical marking.

3. After installing black distance bracket, box should looks like this.

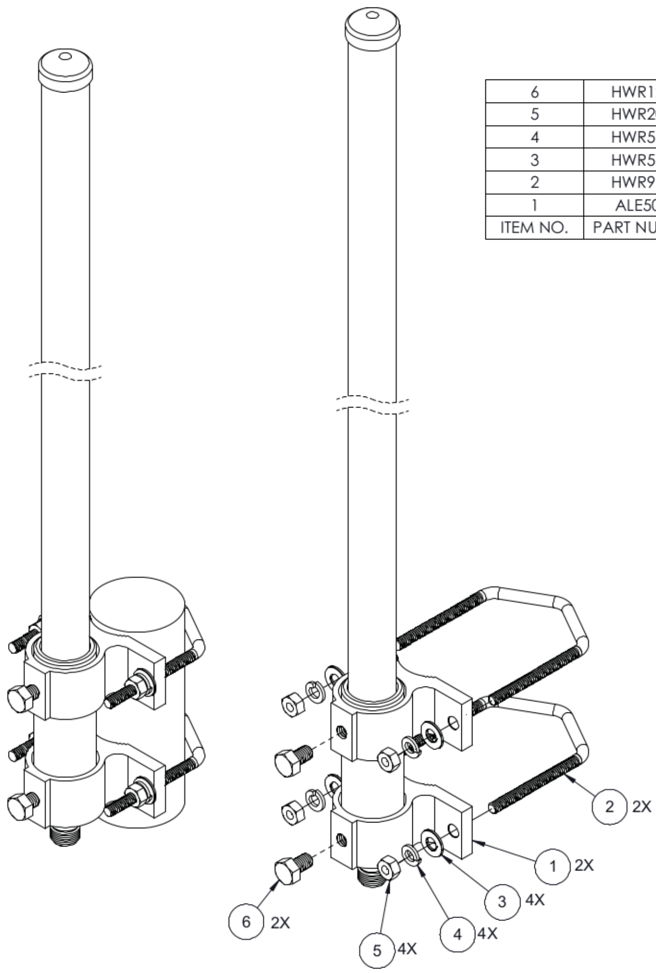
4. At last mount antenna, sector:



Note:

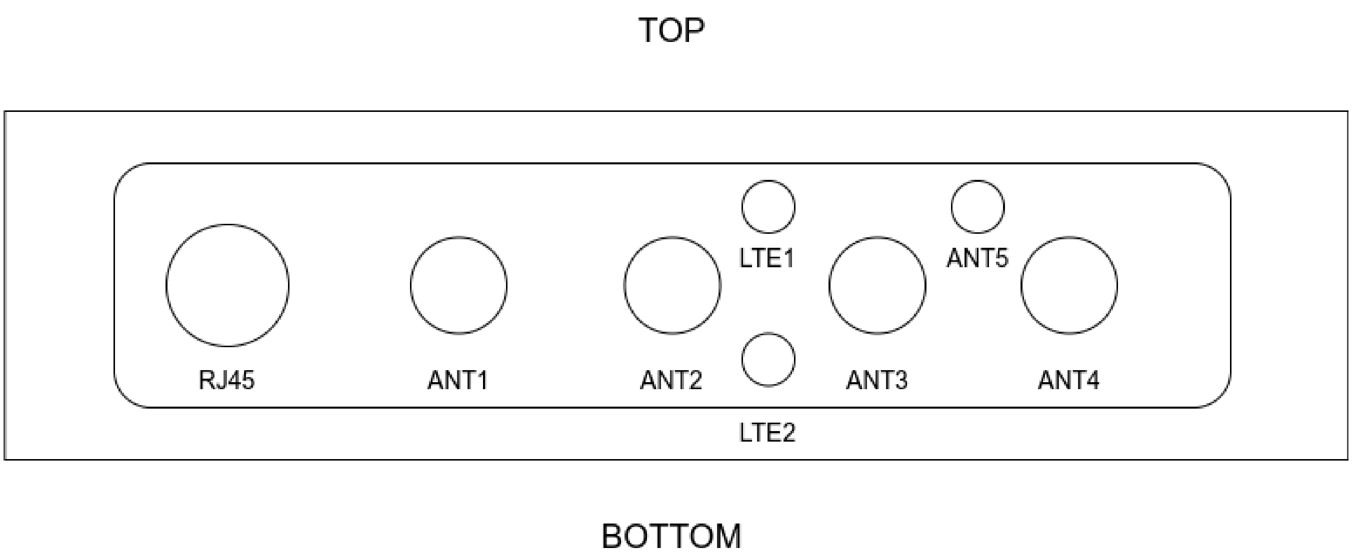
It is important that the cable is connected to the appropriate polarization, for sector antennas Aerobits devices always use vertical polarization (V in antenna outlet description).

5. Or omnidirectional:



ITEM NO.	PART NUMBER	DESCRIPTION	QTY
6	HWR1200	BOL,HEX,5/16-18x1/2,SS,NF	2
5	HWR2000	1/4" - 20 NUT	4
4	HWR5100	1/4" - 20 LOCK WASHER	4
3	HWR5150	Washer, Flat for 1/4" Hardware	4
2	HWR9150	CLAMP,VBOLT,1/4-20,ST,NF	2
1	ALE5000	BKT,EXT,AL,FOR FM2	2

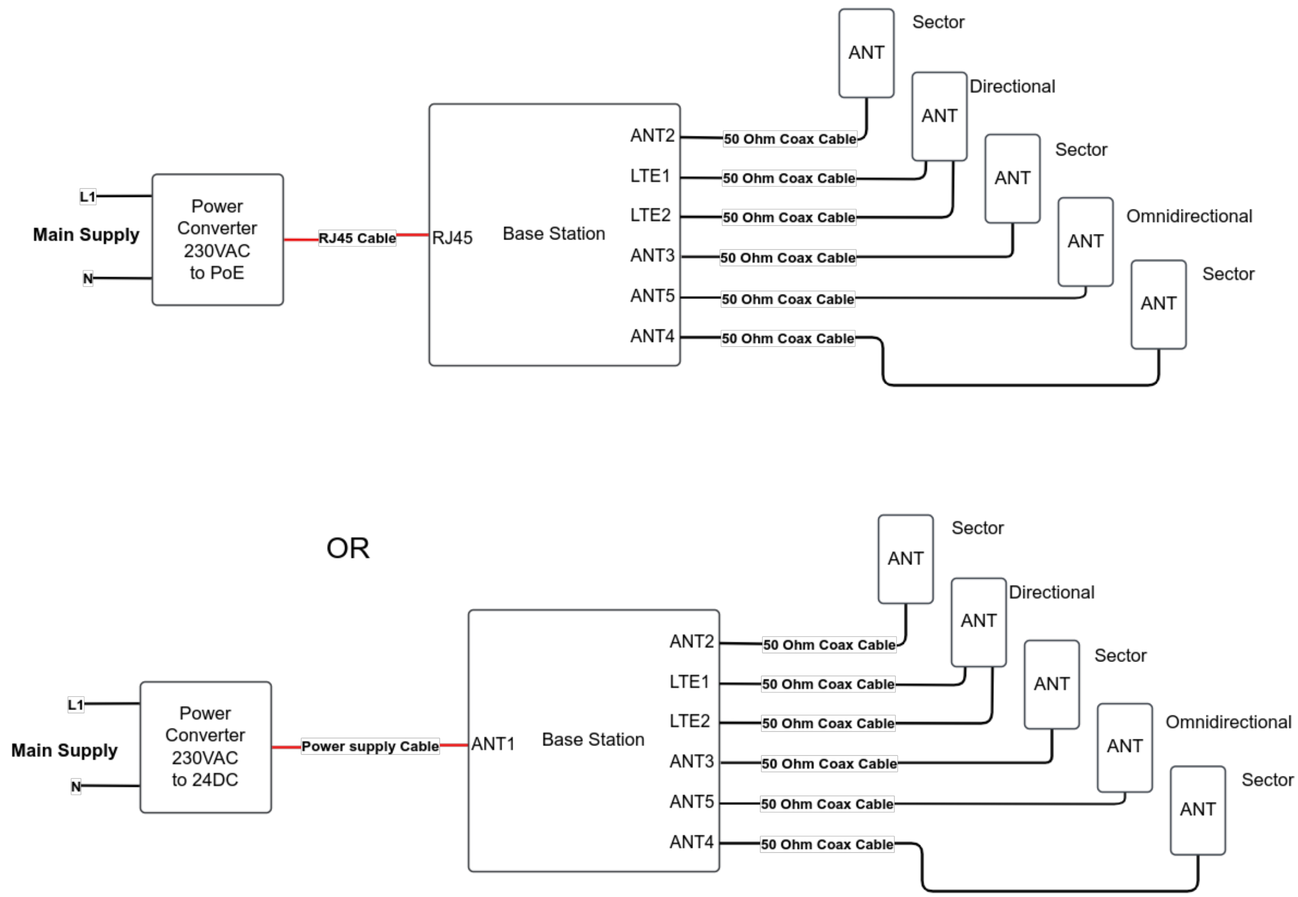
6. After the mechanical part of installation, connect the antennas to the device and the device to the ethernet cable as shown



1. RJ45 - Ethernet/PoE
2. ANT1 - Power delivery
3. ANT2 - ADS-B/FLARM (sector)

4. LTE1 - LTE antenna outlet
5. LTE2 - LTE antenna outlet
6. ANT3 - ADS-B/FLARM (sector)
7. ANT5 - RemoteID (BT/Wi-Fi)
8. ANT4 - ADS-B/FLARM

4.2.2 Electrical connection



4.2.3 Power supply connection PoE

First, connect ethernet cable between PoE supply unit OUT socket and GS Linux.

Warning:

This cable will have PoE supply on it, so do not connect to it other devices that cannot handle it.

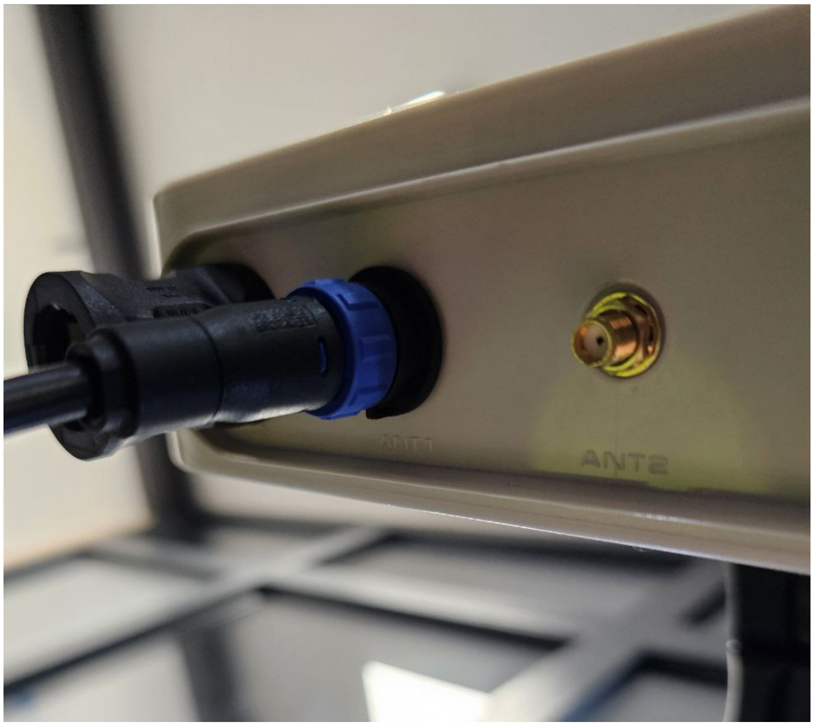
Warning:

Power up PoE supply only when everything is connected, do not switch connections when supply is on.

Optionally, simply connect other socket labeled IN to your regular router/switch, and GS Linux will be connected to it just as with any regular ethernet connection.

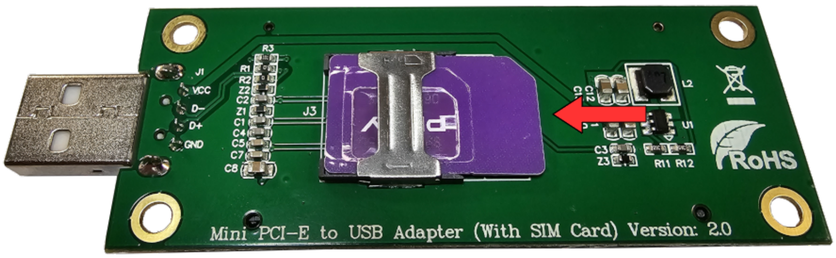
4.2.4 Power supply connection with Bulgin cable

Connect the power cable to the ANT1 socket and supply power with an AC/DC converter.



4.2.5 Inserting a SIM/chip card

Depending on configuration there is a LTE USB stick, that requires a SIM card. Insert card from back just like below.



4.3 Software configuration

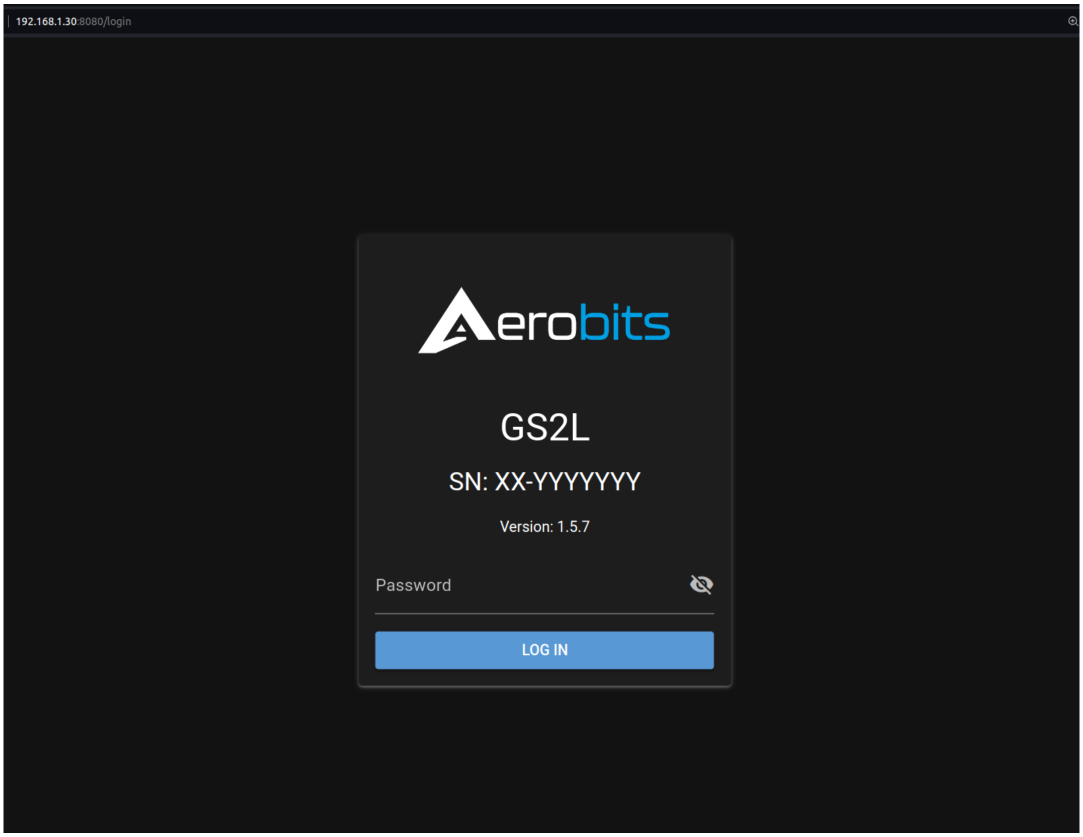
4.3.1 Connection using user interface

Connect station in local network, find its IP address. Start connection using browser and this parameters:

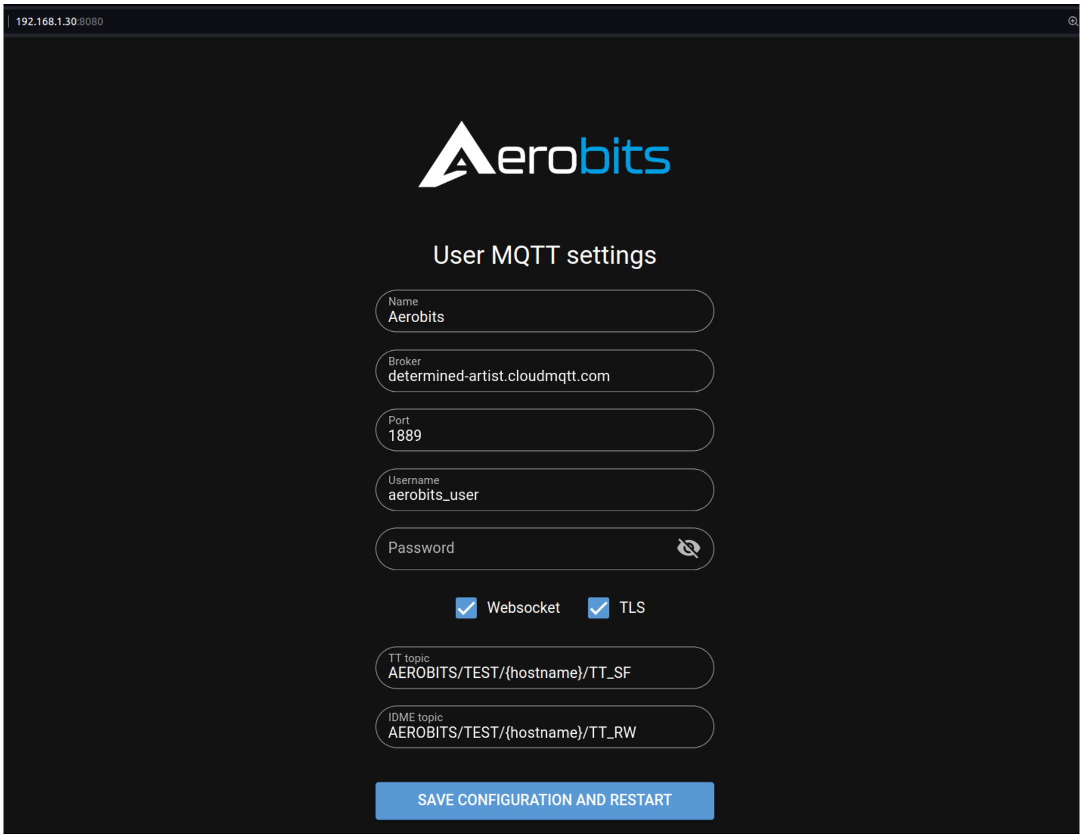
- **IP**: local device IP
- **PORT**: 8080

Example connection shown below:

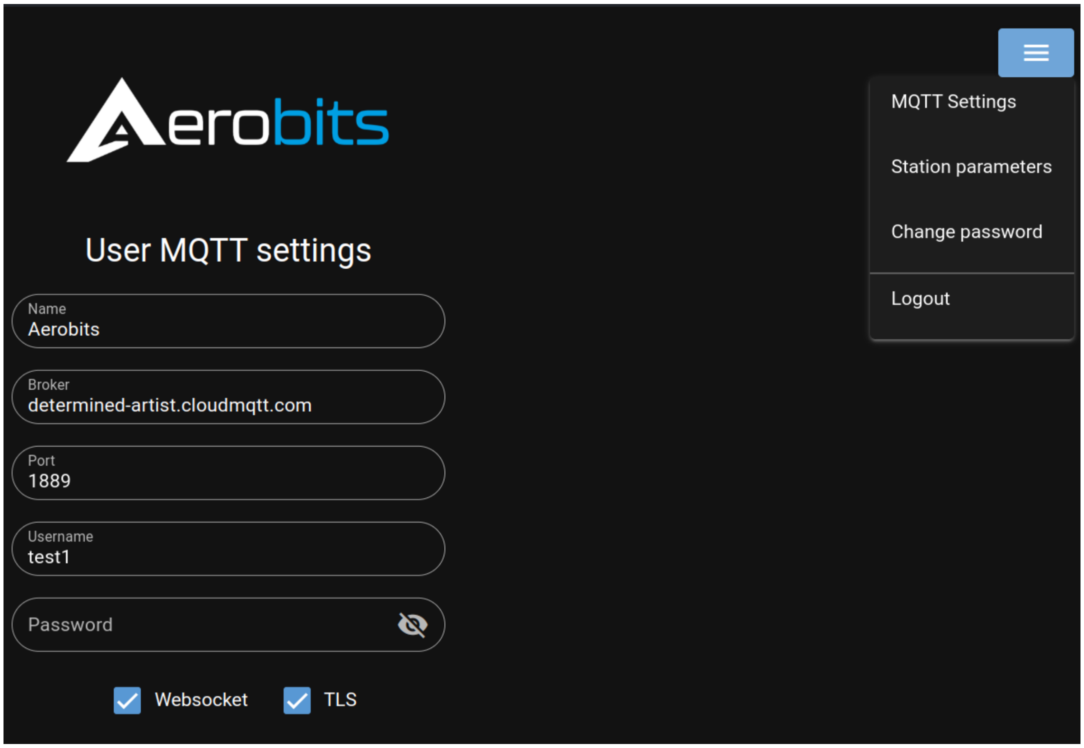
1. Connect to Your device typing IP:PORT in browser. Type password supplied with the device and refresh page after login.

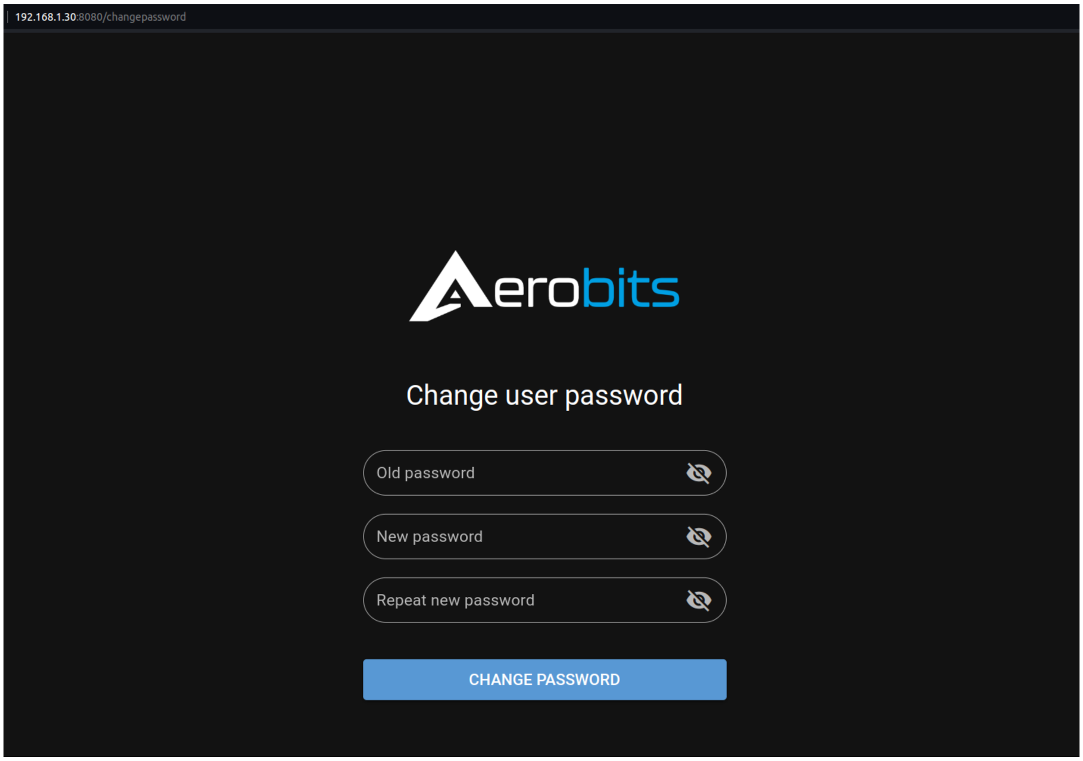


2. Setup Your MQTT connection properties and save configuration.



3. Open menu in the top right corner and change Your custom password and save it.


Aerobits
User MQTT settings
Name
Aerobits
Broker
determined-artist.cloudmqtt.com
Port
1889
Username
test1
Password
Websocket
TLS
MQTT Settings
Station parameters
Change password
Logout


192.168.1.30:8080/changepassword
Aerobits
Change user password
Old password
New password
Repeat new password
CHANGE PASSWORD

4.3.2 Receiving with PlaneMap

1. install PlaneMap, version at least 0.6.9
2. click **Add Connection** and fill:
    (a) IP:PORT
    (b) MQTT
    (c) login
    (d) password
3. click **Apply and Use**
4. click **Connect**
5. status bar at the bootom should have both **online** and **connected**
6. click **Add Topic**
7. fill topic
8. field **Type** selects frames:
    (a) **AeroJSON:STATUS** is sending data constantly, use it to check if station has connection
    (b) **AeroJSON:GNSS** gives position if it can be received
    (c) **AeroJSON:STATION PARAM** gives sensor reports, every few seconds
9. select **AeroJSON:ADSB** to receive planes
10. **subscribe** to receive data
11. **unsubscribe** to change topic or type
12. click **View Data** to confirm that data is received
13. to show planes switch tabs to **Map**

5 Protocols

Each system has protocols unique to it, but protocols common to all systems such as the CSV protocol are also used. The GS2L has only JSON protocol and structure will be presented below.

5.1 Json protocols

JSON (JavaScript Object Notation) is a lightweight, text-based data interchange format that is easy for humans to read and write and easy for machines to parse and generate. JSON is widely used for transmitting data between a server and a web application, as well as for configuration files, data storage, and APIs.

Each message is encoded as separate JSON object, without any excess whitespace, consisting of always present main fields described in a table below and data field depending on message type. All examples shown in this manual have additional spaces for readability.

```
{
    "src": "ID-0000001",
    "ts": 69061337,
    "ver": 1,
    "gnss": {
        "fix": 0, "track": 0, "vVelo": 0
    }
}
```

Depending on message type, data field can have one JSON object (like in **gnss**), or an array of objects (like in **adsb**):

```
{
    "src": "ID-0000001",
    "ts": 69061337,
    "ver": 1,
    "adsb": [
        { "icao": "78A001", "sigStr": -80, "sigQ": 5, "fps": 0 },
        { "icao": "78A002", "sigStr": -81, "sigQ": 5, "fps": 0 },
        { "icao": "78A003", "sigStr": -82, "sigQ": 5, "fps": 0 }
    ]
}
```

Table 4: Description of main JSON fields.

JSON Field	Unit	Example	Description
src	–	ID-0000001	OEM TT serial number.
ts	milliseconds	69061337	Timestamp in milliseconds, relative to last UTC midnight. Value 69061337 encodes 19:11:01.337. Omitted if unknown.
ver	–	1	JSON protocol version. See details below.
gnss	–	{…}	Data fields, described in subchapters below.
adsb	–	[{…},{…},{…}]	Array of multiple sets of data fields, described in subchapters below.

Note:

The order of JSON object fields in any part of message may vary between firmware revisions and messages.

Some JSON objects have fields, of which values may sometimes be unknown. In this case, they are skipped in JSON output. In following chapters, each of those fields are explicitly marked as ommitable.

Note:

In case of JSON objects consisting of only ommitable fields, if none of them are set, the whole object may be omitted.

The *src* field has name of the station. It can be also configured to have name of module that received message.

The *ver* field indicates JSON protocol version. Future ICD versions may introduce additional fields without changing the version number. If a breaking change occurs in Ground Station with Linux JSON specification, the version number is guaranteed to be incremented.

Note:

The version number of JSON protocol described in this document is 1.

6 ADS-B receiver subsystem

6.1 ADS-B JSON protocol

The “adsb” section contains aircraft information determined by OEM TT-Multi-RF internal ADS-B processing engine. The messages are encoded as JSON array with at least one entry. Each entry is an object consisting of fields denoted in table *adsb* (page 22).. Reports for each ADS-B aircraft are updated once every second.


```
{
    "src": "33-0000683",
    "ts": 69061337,
    "ver": 1,
    "adsb": [
        {
            "icao": "780A3F",
            "sigStr": -67,
            "sigQ": 9,
            "fps": 5,
            "lat": 34.39696,
            "lon": -85.1055,
            "baroAlt": 35000,
            "geoAlt": 36975,
            "track": 143.78,
            "hVelo": 528,
            "vVelo": 0,
            "ident": "CPA3174",
            "squawk": "5730",
            "ecat": 5,
            "nacp": 9,
            "nacv": 1,
            "nicBaro": 1,
            "nic": 8,
            "surf": 1
        }
    ]
}
```


Table 5: Descriptions of JSON ADS-B section fields.

JSON Field	Unit	Example	Description
src	–	ID-0000001	See table *Description of main JSON fields.* (page 20).
ts	milliseconds	69061337	See table *Description of main JSON fields.* (page 20).
ver	–	1	See table *Description of main JSON fields.* (page 20).
adsb	–	type of message	
icao	–	DABABE	ICAO address, 24-bit value encoded in uppercase hexadecimal, with leading zeros.
sigStr	dBm	-95	Signal strength, in dBm.
sigQ	dB	2	Signal quality, in dB.
fps	–	5	Frames received per second
lat	–	53.42854	Latitude. Omitted if position is unknown.
lon	–	14.55281	Longitude. Omitted if position is unknown.
baroAlt	ft	1725	Barometric altitude, in feet. Omitted if unknown.
geoAlt	ft	1712	Geometric altitude, in feet. Omitted if unknown.

continues on next page

Table 5 – continued from previous page

JSON Field	Unit	Example	Description
track	degree°	72.18	Track angle, 0°..360°. Omitted if unknown.
hVelo	knots	10.5	Horizontal velocity, in knots. Omitted if unknown.
vVelo	ft/min	50	Vertical velocity, in ft/min, positive value is upwards. Omitted if unknown.
ident	–	TEST8	Callsign, up to 8 chars. Omitted if unknown.
squawk	–	7232	Squawk, 8 octal digits. Omitted if unknown.
ecat	–	13	Emitter category code, see table *ecat* (page 23).. Omitted if unknown.
nacp	–	3	NAC_P value, as described in ED-102A. Omitted if value is 0 (unknown).
nacv	–	1	NAC_V value, as described in ED-102A. Omitted if value is 0 (unknown).
nicBaro	–	1	NIC_{BARO} value, as described in ED-102A. Omitted if value is 0 (unknown).
nic	–	2	NIC value, as described in ED-102A. Omitted if value is 0 (unknown).
surf	–	2	1 if on ground, else omitted. Omitted if unknown.

The emitter category values returned in *ecat* field is shown in table below:

Table 6: ADS-B emitter category values in JSON protocol.

“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 Balloons).
20	Cluster obstacle.
21	Line obstacle.

7 FLARM receiver subsystem

7.1 FLARM JSON protocol

The “flarm” section contains aircraft information determined by OEM TT-Multi-RF internal FLARM processing engine. The messages are encoded as JSON array with at least one entry. Each entry is an object consisting of fields denoted in table *flarm* (page 24).. Reports for each FLARM aircraft are updated once every second.


```
{
    "src": "ID-0000001",
    "ts": 69061337,
    "ver": 1,
    "flarm": [
        {
            "idType": 1,
            "id": "DABABE",
            "type": 13,
            "danger": 1,
            "lat": 53.42854,
            "lon": 14.55281,
            "alt": 1725,
            "track": 72.18,
            "hVelo": 10.5,
            "vVelo": 50,
            "movMode": 5,
            "stealth": 1,
            "notrack": 1
        }
    ]
}
```


Table 7: Descriptions of JSON FLARM section fields.

JSON Field	Unit	Example	Description
src	–	ID-0000001	See table *Description of main JSON fields.* (page 20).
ts	milliseconds	69061337	See table *Description of main JSON fields.* (page 20).
ver	–	1	See table *Description of main JSON fields.* (page 20).
flarm	–	type of message	
idType	–	1	Aircraft id type. 0: randomized, 1: ICAO, 2: FLARM. Omitted if invalid.
id	–	DABABE	Aircraft id, 32-bit value encoded in uppercase hexadecimal, with leading zeros.
type	–	13	Aircraft type, see table *ecat-flarm* (page 25)..
danger	–	2	Alarm level, 0: no danger, 3: high danger. Omitted if unknown.
lat	degree°	53.42854	Latitude. Omitted if position is unknown.
lon	degree°	14.55281	Longitude. Omitted if position is unknown.
alt	m	1725	Barometric altitude, in meters.
track	degree°	72.18	Track angle, 0°..360°. Omitted if unknown.
hVelo	m/s	10.5	Horizontal velocity, in m/s. Omitted if unknown.
vVelo	m/s	50	Vertical velocity, in m/s, positive value is upwards. Omitted if unknown.
movode	–	5	Movement mode.1: stationary, 4: circling right, 5: flying,7: circling left.
stealth	–	1	Set to 1 if target has Stealth flag set, otherwise omitted.

continues on next page

Table 7 – continued from previous page

JSON Field	Unit	Example	Description
notrack	–	1	Set to 1 if target has Notrack flag set, otherwise omitted.

The list of possible FLARM "Aircraft type" values returned in *type* field is shown in table *ecat-flarm* (page 25).

Table 8: FLARM aircraft type category values in JSON protocol.

"ecat" value	Description
0	Reserved.
1	Glider, Motor glider.
2	Tow plane, tug plane.
3	Helicopter, gyrocopter, rotocraft.
4	Skydiver, parachute.
5	Drop plane for skydivers.
6	Hang glider (hard).
7	Hang glider (soft).
8	Aircraft with reciprocating engine.
9	Aircraft with jet / turboprop engine.
10	Reserved.
11	Balloon (hot, gas, weather, static).
12	Airship, blimp, zeppelin.
13	Unmanned Aerial Vehicle (UAV).
14	Reserved.
15	Static obstacle.

8 GNSS receiver subsystem

8.1 GNSS JSON protocol

The *gnss* section contains basic GNSS information. This message is sent once per second. The example JSON message with "gnss" section fields described:


```
{
    "src": "ID-0000001",
    "ts": 69061337,
    "ver": 1,
    "gnss": {
        "fix": 1,
        "lat": 53.42854,
        "lon": 14.55281,
        "altWgs84": 499.6,
        "altMsl": 508.6,
        "track": 127.3,
        "hVelo": 10.5,
        "vVelo": 25,
        "gndSpeed": [
            5.2,
            2.1
        ],
        "acc": {
            "lat": 5.2,
            "lon": 2.1,
            "alt": 3.6
        },
        "nacp": 12,
        "nacv": 2,
        "nic": 12
    }
}
```


Table 9: Descriptions of JSON GNSS section fields.

JSON Field	Unit	Example	Description
gnss			Type of message
fix	–	1	Set to 1 if onboard GNSS currently has fix, otherwise 0.
lat	degree °	53.42854	Last known latitude. Omitted if there was no GNSS fix since device boot.
lon	degree °	14.55281	Last known longitude. Omitted if there was no GNSS fix since device boot.
altWgs84	m	499.6	Last known WGS-84 Altitude, in meters. Omitted if there was no GNSS fix since device boot.
altMsl	m	508.6	Last known MSL Altitude, in meters. Omitted if there was no GNSS fix since device boot.
track	–	127.3	Track angle, 0°..360°, relative to true north. Omitted if unknown.
hVelo	–	10.5	Horizontal velocity, in knots. Omitted if unknown.
vVelo	–	25	Vertical velocity, in m/s. Positive value is upwards. Omitted if unknown.
gndSpeed	knots	[5.2,2.1]	Ground speed in east-west and north-south axes respectively, in knots. Positive value is East and North. Derived from track / hVelo values. Omitted if unknown.
acc	–	struct	Accuracy in all 3 dimensions

continues on next page

Table 9 – continued from previous page

JSON Field	Unit	Example	Description
lat	m	5.2	Accuracy of latitude, in meters. Omitted if unknown.
lon	m	2.1	Accuracy of longitude, in meters. Omitted if unknown.
alt	m	3.6	Accuracy of altitude, in meters. Omitted if unknown.
nacp	–	12	Navigational Accuracy Category for Position value, as defined in ED-282. Omitted if unknown.
nacv	–	2	Navigational Accuracy Category for Velocity value, as defined in ED-282. Omitted if unknown.
nic	–	12	Navigation Integrity Category as defined in ED-282. Omitted if unknown.

9 RemoteID receiver subsystem

9.1 RemoteID JSON protocol

The *remoteID* section contains aircraft information determined by Ground Station with Linux internal Remote ID processing engine. The messages are encoded as JSON array with at least one entry. Each entry is an object consisting of fields denoted below, if field is unknown will be omitted (empty). Reports for each remoteID aircraft are updated once every second.


```
{
    "src": "ID-0000001",
    "ts": 69061337,
    "ver": 1,
    "remoteid": [
        {
            "framePrefix": "B4",
            "aircraftID": "18099300000132",
            "idType": 1,
            "uasType": 2,
            "lat": 53.42854,
            "lon": 14.55281,
            "height": 1,
            "baroAlt": 17,
            "geoAlt": 17,
            "track": 72.00,
            "hVelo": 10.5,
            "vVelo": 50,
            "statusFlag": 0,
            "operator":
            {
                "id": "AAABBBBBBBBBBBBC-DDD",
                "idType": 2,
                "lat": 53.42854,
                "lon": 14.55281,
                "locType": 0
            }
            "times": 350,
            "rssi": -50,
            "selfIdType": 1,
            "selfId": "Test",
            "frameType": 15,
            "mac": "df:a5:c3:84:78:66",
            "channel": 6
        }
    ]
}
```


Table 10: Descriptions of JSON RemoteID section fields.

JSON Field	Unit	Example	Description
src	–	ID-0000001	See table *Description of main JSON fields.* (page 20).
ts	milliseconds	69061337	See table *Description of main JSON fields.* (page 20).
ver	–	1	See table *Description of main JSON fields.* (page 20).
framePrefix	–	B4	frame prefix, see description *frame-prefix* (page 29).
remoteid	–	type of message	

continues on next page

Table 10 – continued from previous page

JSON Field	Unit	Example	Description
aircraftID	–	18099300002137	Aircraft ID represented by string value
idType	–	1	ID type see table 12
uasType	–	2	Callsign of aircraft, see table 13 .
lat	–	53.42854	Latitude in degrees, accuracy 0.6 degree
lon	–	14.55281	Longitude in degrees, accuracy 0.6 degree
height	m	1	Height based on start up altitude, in meters .
baroAlt	m	17	Barometric altitude, in meters.
geoAlt	m	17	Geometric altitude, in feet. Omitted if unknown.
track	degree°	72.18	Track angle, 0°..360°. Omitted if unknown.
hVelo	m/s	10.5	Horizontal velocity, in m/s, accuracy 0.1 m/s.
vVelo	m/s	50	Vertical velocity, in m/s, positive value is upwards, accuracy 0.1 m/s.
operator	–	struct	Operator status.
id	–	AAABBBBBBBBBBBBC-DDD	The operator number from local FAA department.
idType	–	2	Specific type of Operator ID.
lat	–	53.42854	The operator latitude in degrees, accuracy 0.6 degree.
lon	–	14.55281	The operator longitude in degrees, accuracy 0.6 degree.
locType	–	0	The operator location type.
times	s	350	Timestamp of the sent frame expressed in seconds since current hour, accuracy 0.1 s-1.5 s.
rssi	dBm	-50	Set to 1 if plane is on earth surface. Omitted if plane is in air or unknown.
selfIdType	–	1	Self id type *self-id* (page 30).
selfId	–	Test	Self ID status
frameType	–	15	Frame type *frame-type* (page 30).
mac	–	df:a5:c3:84:78:66	MAC address
channel	–	6	Wi-Fi channel

Whereby the following prefixes mean:

Table 11: Descriptions of RemoteID frame prefix.

Frame prefix value	Description
B4	Bluetooth 4.0 (Legacy) frame.
B5	Bluetooth 5.0 frame.
WN	Wi-Fi NaN frame.
WB	Wi-Fi beacon frame.

Note:

Referring to the ASD-STAN prEN 4709-002 standard, our product displays all the required information(ASD-STAN prEN 4709-002 Table 1 - Data Dictionary), optional data is only available upon special request.

Below is a list off self Id types returned in Self Id value field.

Table 12: Descriptions of RemoteID Self Id value field.

Self Id Type value	Description
0	Text Description.
1	Emergency Description.
2	Extended Status Description.
3–200	Reserved.
201–255	Available for private use.

Below is a list off frame types returned in Frame Type value field.

Table 13: Descriptions of RemoteID Frame Type value field.

ID Type value	Description
0	Basic ID.
1	Location.
3	Self ID.
4	System.
5	Operator ID.
15	Packed all in one.

10 UAT receiver subsystem

10.1 UAT JSON protocol

The "uat" section contains aircraft information determined by OEM TT-Multi-RF internal UAT processing engine. The messages are encoded as JSON array with at least one entry. Each entry is an object consisting of fields denoted in table *Descriptions of JSON UAT section fields.* (page 32). Reports for each UAT aircraft are updated once every second.


```
{
    "src":"33-0000687",
    "ts": 69061337,
    "ver":1,
    "uat":[
        {
            "icao":"777888",
            "flags":
            {
                "groundState":false,
                "updAltBaro":false,
                "updPosition":false,
                "updTrack":false,
                "updVeloH":false,
                "updVeloV":false,
                "updAltGeo":false
            },
            "call":"N61ZP",
            "squawk":7232,
            "lat":90.0000,
            "lon":180.0000,
            "altBaro":150,
            "track":142,
            "velH":10,
            "velV":60,
            "sigStr":0,
            "sigQ":0,
            "fps":11,
            "nicnac":
            {
                "nacP":0,
                "nacV":0,
                "nicBaro":0
                },
            "altGeo":150,
            "ecat":14,
            "emergency":0,
            "uatFlags":
            {
                "utcCoupling":false,
                "cdti":false,
                "acasOperational":false,
                "acasActive":false,
                "identActive":false,
                "atcActive":false,
                "headingMagnetic":false,
                "reservedMS1":0
            }
```


(continues on next page)

(continued from previous page)

```
            }
        ]
}
```

Table 14: Descriptions of JSON UAT section fields.

JSON Field	Unit	Example	Description
src	–	ID-0000001	See table *Description of main JSON fields.* (page 20).
ts	milliseconds	69061337	See table *Description of main JSON fields.* (page 20).
ver	–	1	See table *Description of main JSON fields.* (page 20).
uat	–	type of message	
icao	–	3C65AC	ICAO number of aircraft (3 bytes, uppercase hexadecimal, with leading zeros)
flags	–	type of message	
groundState	bool	True	
updAltBaro	bool	True	
updPosition	bool	True	
updTrack	bool	True	
updVeloH	bool	True	
updVeloV	bool	True	
updAltGeo	bool	True	
call	–	N61ZP	Callsign of aircraft
squawk	–	7232	SQUAWK of aircraft (available when there is no Callsign)
lat	degree °	57.57634	Latitude, in degrees
lon	degree °	17.59554	Longitude, in degrees
altBaro	ft	5000	Barometric altitude, in feet
track	degree °	355	Track of aircraft, in degrees
velH	$knot$	464	Horizontal velocity of aircraft, in knots
velV	ft/min	1344	Vertical velocity of aircraft, in ft/min
sigStr	dBm	-70	Signal strength, in dBm
sigQ	–	1	Signal quality, number of errors corrected, 0(best)…6(worst)
fps	–	5	Number of raw frames received from aircrafts during last second
nicnac	–	struct	NIC/NAC *Descriptions of JSON UAT nicnac fields.* (page 32)
altGeo	$feet$	5000	Geometric altitude, in feet
ecat	–	14	Emitter category *UAT emitter category values in JSON protocol.* (page 33)
emergency	–	3	UAT emergency status *Descriptions of UAT emergency status field.* (page 33)
uatFlags	–	1F	UAT special status flags *Descriptions of UAT uatFlags status field.* (page 33)

Table 15: Descriptions of JSON UAT nicnac fields.

nicnac	Aircraft message start indicator	Example value
nacP	NAC_P value, as described in ED-102A. Omitted if value is 0 (unknown).	1
nacV	NAC_V value, as described in ED-102A. Omitted if value is 0 (unknown).	1

continues on next page

Table 15 – continued from previous page

nic-nac	Aircraft message start indicator	Example value
nicBaro	NIC_{BARO} value, as described in ED-102A. Omitted if value is 0 (unknown).	1

The emitter category values returned in `ecat` field is shown in table below:

Table 16: UAT emitter category values in JSON protocol.

"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 Balloons).
20	Cluster obstacle.
21	Line obstacle.

Table 17: Descriptions of UAT emergency status field.

Value	Description
0	No emergency/Not reported
1	General emergency
2	Lifeguard/medical emergency
3	Minimum fuel
4	No communications
5	Unlawful interference
6	Downed Aircraft
7	(Reserved)

Table 18: Descriptions of UAT uatFlags status field.

Value	Description
utcCoupling	Messages coupled to UTC
cdti	CDTI Traffic Display Capability

continues on next page

Table 18 – continued from previous page

Value	Description
acasOperational	TCAS/ACAS Installed and Operational
acasActive	TCAS/ACAS Resolution Advisory Active
identActive	IDENT Switch Active (equal or less than 20 seconds since activated by pilot)
atcActive	Receiving ATC Services
headingMagnetic	Heading according to magnetic north (true north default)
reservedMS1	Reserved for air quality

11 Network communication system

11.1 Network communication modes MQTT

The GS2L communicates through the **LTE** or **LAN** network using **MQTT** 3.1 protocol. Connection can be configured (GUI) to use username and password authentication, as well as **TLS** encryption. All data can be transmitted into multiple or single **MQTT** topic.

12 Sensors receiver subsystem

12.1 Sensor JSON protocol

The *sensor* section contains values acquired from miscellaneous sensors present in Aerobits device hardware and consists of fields shown below. This message is sent once per second. All fields are optional - they are sent only if appropriate sensor is enabled.


```
{
    "src": "ID-0000001",
    "ts": 69061337,
    "ver": 1,
    "StationParams": {
        "PowerSensor": {
            "BusVoltage": 4.6,
            "BusCurrent": 650.122,
            "Power": 3000.573,
            },
        "HumiditySensor": {
            "Temperature": 36.9,
            "Humidity": 19,
            },
        "PressureSensor":{
            "Pressure":1002.1,
            "Temperature":36.8,
            "MSLRelativeAltitude":120.76
        }
    }
}
```


Table 19: Descriptions of JSON Sensor section fields.

JSON Field	Unit	Example	Description
src	–	ID-0000001	See table *Description of main JSON fields.* (page 20).
ts	milliseconds	69061337	See table *Description of main JSON fields.* (page 20).
ver	–	1	See table *Description of main JSON fields.* (page 20).
StationParams	–	type of message	
PowerSensor	–	type of sensor	
BusVoltage	V	4.6	Current voltage sensor value
BusCurrent	mA	650.122	Current current sensor value
Power	mW	3000.573	Current power sensor value
HumiditySensor	–	type of sensor	
Temperature	°C	36.9	Current temperature sensor value
Humidity	%	19	Current humidity sensor value
PressureSensor	–	type of sensor	
Pressure	hPa	1002.1	Current pressure sensor value
Temperature	°C	36.8	Current temperature sensor value
MSLRelativeAltitude	m	120.76	Current MSL height (using standard reference 1013.25hPa at 0m)

13 Disclaimer

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.

Information in this document supersedes and replaces all previously supplied information.