0000000000000
0000001000000
0000011100000
0000010100000
0000011100000
0000111110000
0001111111000
0011011101100
0110011100110
0100011100010
0000111110000
0001101011000
0000000000000
Aerial Robotics

FLEXIPILOT 1.35

Using On Screen Display add-ons, wireless modems, RVOSD as visual telemetry display and GPS output

The document describes connection and setup of the on-screen-display from RVOSD, which is and optional autopilot assist responsible for adding text and graphics to the video signal coming from the surveillance camera and sending it to the video transmitter.
RVOSD needs special firmware update for autopilots.
The document also discusses GPS configuration that uses shared data pin with OSD output.

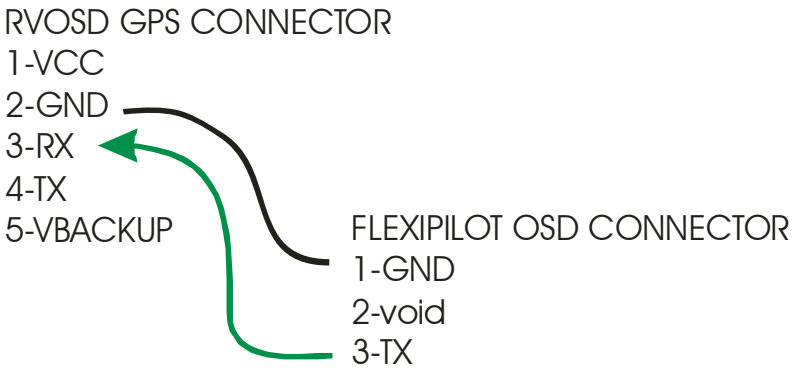
The autopilot is using RVOSD (RangeVideo On-Screen Display) as display processor. RVOSD navigation algorithm, servo outputs and thermopiles are not used. In particular, the AHI (Artificial Horizon Indicator) and waypoint sequencing is handled entirely by FLEXIPILOT. Therefore, before each flight, you don't need to reconfigure RVOSD itself in any way.
RVOSD, once connected to FLEXIPILOT, is not using its own GPS. Instead, all data is sent directly from FLEXIPILOT (which uses its own GPS).
As a result, only the main RVOSD dual-PCB 'box' has to be mounted in the fuselage, besides wireless camera and video transmitter.

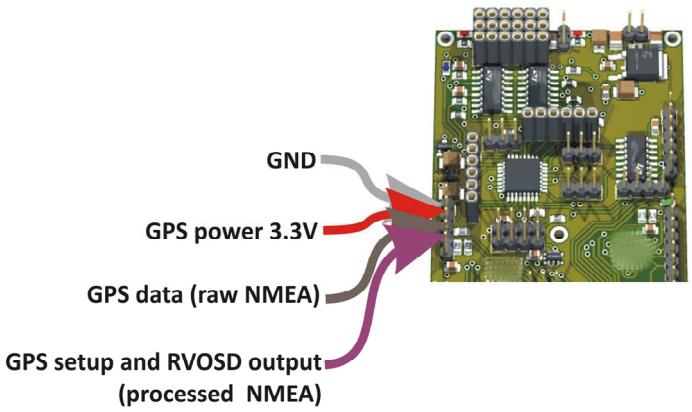
Using OSD from different brand is possible, but many advanced features of RVOSD will be not present, notably the AHI, TSD (Tactical Situation Display or Navigation Screen).

Secondary use of the OSD connector during mission simulation is also explained.

Data connection

RVOSD connector is essentially a serial port, 115200 baud, 8N1, asynchronous.
The data transmitted uses 2-wire cable, of which darker one is ground, the other is signal.
No other connections are necessary; however powering the autopilot directly from RVOSD is also possible.
The connector is sending GPGGA, GPVTG and proprietary GPOSD commands.
It is possible to connect this output to wireless modem and obtain classic telemetry.



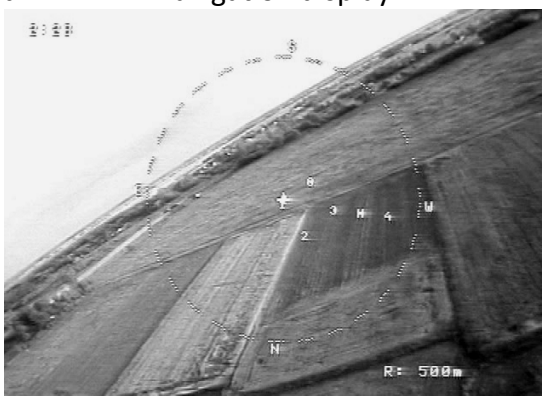


Display screens

RVOSD can display information like roll, pitch, actual position, home position, waypoints, altitude, climb rate, course and speed. The main display also contains 'radar' display with short lines pointing home and to the next waypoint. In case of problems, you can use this display for using manual override flying on distance in order to recover from potentially dangerous situation, this is called FPV flying. You can also fly in manual RC mode at close distance using the display enhanced by the autopilot, particularly AHI (Artificial Horizon Indicator). The AHI is relying entirely on autopilot's IMU and no thermopile sensor is necessary. Also, the autopilot contains its own GPS and sends all the GPS data to the OSD.

Two types of display screens can be automatically toggled by the autopilot:

HUD display and navigation display



For description of the display elements, refer to RVOSD user's manual.
The navigation screen is using up to 5 waypoints:

```
0=next waypoint
1=last met waypoints
2=older waypoint
…
4=the oldest waypoint
```

The waypoint list is updated each time the autopilot meets a new waypoint.
The home position is calculated entirely by the OSD.
You can also use the display in simulation mode.

Note: the MSL altitude in all GPS data send to the OSD is replaced by AGL altitude from barometric sensor. The WGS altitude field contains climb rate.

Setup variables

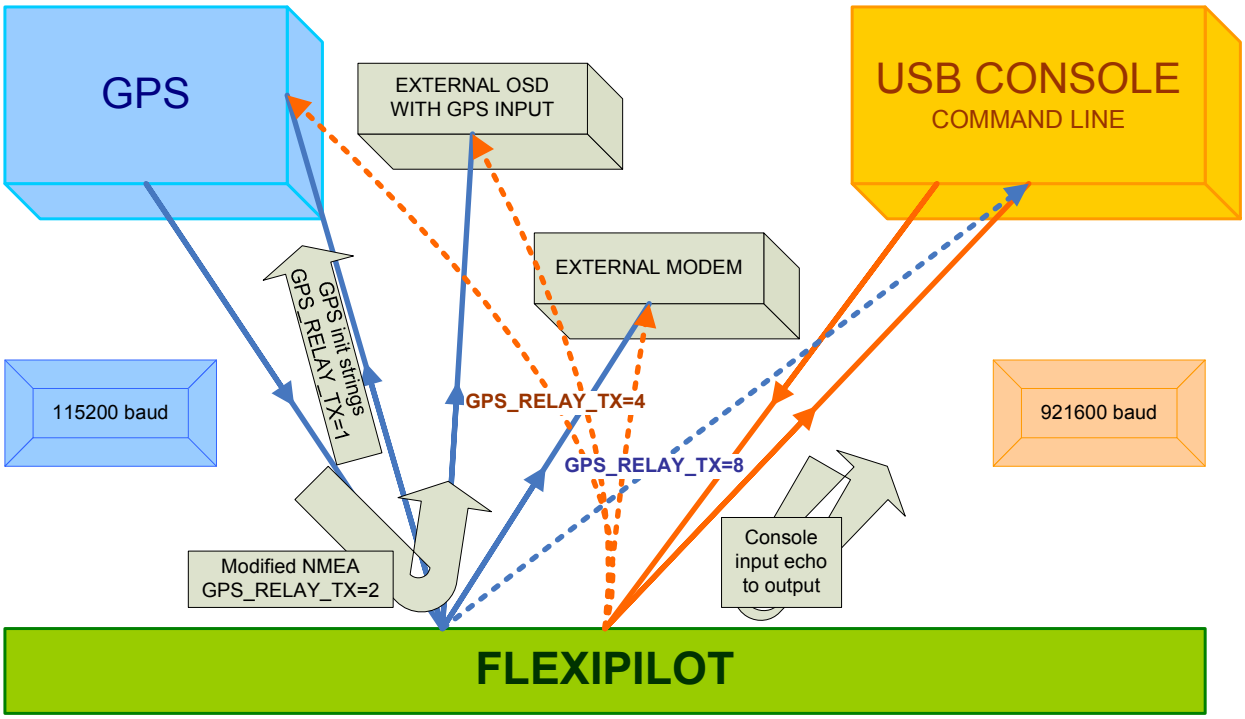
Note: in order to log OSD output, use
LOG_SOURCE=2 (GPS/OSD OUTPUT) and LOG_DUMP_MODE=0 (RAW)

GPS_RELAY_TX
Possible values: 0-7
Default: 1
Enables the OSD output. This parameter controls the TX signal coming both to GPS receiver and OSD display unit. Use the flag values that can be summed together in order to give selected functionality.
Note: the output is interleaved with OSD_PROTOCOL data.

Possible flags:
0 no extra output
1 send GPS initialization strings
2 reserved for future use
4 mirror console output to OSD/modem (useful during tuning)
8 mirror OSD/modem output to console

Example values:
1 standard setting
5 during tuning with wireless modem

Note: what you see on the OSD/modem output can be one of the following:
-GPS initialization strings $PMTK (if GPS_RELAY_TX has flag 1)
-all console output (if GPS_RELAY_TX has flag 4)
-all console input (if GPS_RELAY_TX has flag 4), because console input is acknowledged to console output and as a consequence you will see it also on OSD output
-if OSD_PROTOCOL>0, you will see one-way telemetry synchronous with autopilot



Interactions of the command console input/output,
GPS input/output and modem/OSD output

GPS_MODE

Possible values: 0-7

Default: 3

1 enable DGPS (WAAS/EGNOS/MSAS)

2 enable internal clock correction using GPS

4 enable all NMEA messages (may delay the autopilot, use only for GPS testing)

DGPS correction:

Disable DGPS on southern hemisphere or when there are known DGPS region-wide failures/outages.

Clock correction:

When using critical logging during trials with full rate logging (or rockets), disable clock correction because arriving occasional 1s correction might alter realtime data alignment for a few percent during a period of a few seconds. Such correction is desirable for long missions that require accurate timestamps, but undesirable for analysis of events that might happen anytime and endure less than a half minute.

GPS_INIT_BAUDRATE

Default: 115200

Possible values:

2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, 76800, **115200**

This is the baudrate used for sending GPS configuration strings at boottime.

During GPS initialization, if this variable is different than 115200 and GPS_BAUDRATE is 115200, additional setup is performed trying to switch GPS to the new baudrate.

GPS_BAUDRATE
Default: 115200
Possible values:
2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, 76800, **115200**
This variable alters the baudrate used by OSD output connector, and is necessary for emulating generic GARMIN devices. Note however, that the onboard GPS has fixed baudrate and cannot communicate with the autopilot with another baudrate than 115200. Therefore you can change this only for simulation, after which you must switch it back to 115200, otherwise the GPS will never lock and the flight-ready led will remain blinking.

OSD_TRIGGER
Before takeoff, the user can change the OSD screen using TV remote supplied with RVOSD.
After takeoff, the FLEXIPILOT takes control of the screen switching if the OSD_TRIGGER is not 0. In this case, the HUD display will be selected by default, switching to the navigation screen when either of the 2 triggers is active, or when heading/arrived home.
As the triggers are general mechanism that can be also enabled/disabled by the RC receiver and support programming periodic action, you can obtain any desired pattern of the display switching. For example, you can switch to navigation screen over specific flight leg, upon arrival on waypoint using trigger logic, or request periodic screen toggling.

Possible flags enabling transition from HUD to NAVI display:
0 manual screen switching by RVOSD remote
1 switching based on Trigger 1 (when active)
2 switching based on Trigger 2 (when active)
4 display navigation when going home
8 continue to display navigation when reached home
The values 4 and 8 are necessary for full control as there is separate setting **TRIGGER_HOME_DISABLE** that optionally disables triggers when going home.

The GPS positions sent to OSD output can be hidden or altered in order to avoid unwanted observers of the video signal to guess the flying field location. The GPS position displayed on the video channel will become encrypted.
The variables OSD_OBFUSCATE_LAT and OSD_OBFUSCATE_LON control the offset added to the true position before it is send to the OSD.

OSD_OBFUSCATE_LAT
Latitude offset added to the accurate GPS position, in decimal format,
floating-point values between -90..90deg.
Example: OSD_OBFUSCATE_LAT=12.336922

OSD_OBFUSCATE_LON
Longitude offset added to the accurate GPS position, in decimal format,
floating-point values between -180..180deg.
Example: OSD_OBFUSCATE_LON= -113.141261

OSD_OBFUSCATE
Possible values:
0 send true position
1 use fixed offset
2 select random value for position offsets once upon startup, since you know your true takeoff position, you can use it to find the offset having the video recording and find the lost UAV
3 change random offset values all the time, it will be impossible to record flight path from the video

OSD_OBFUSCATE_SEED
Random number generator value, changed every time a new random value is being used.
Example: OSD_OBFUSCATE_SEED=1234 will ensure every time the random number sequence will start with the same key value.

OSD_PROTOCOL
Default: 0
Possible values: 0…2047
Possible flags:
1 Synthetic NMEA sequence emulating Garmin GPS 1Hz ($GPRMC, $GPGGA, $GPVTG)
2 AerialRobotics BEACON protocol (various messages up 1…20Hz)
4 RVOSD mixed output (16Hz $GPOSD, 4Hz: $GPVTG, $GPRMC, $GPVTG, $GPGGA)
8 RVOSD AHI/IMU output (20Hz: $GPOSD)
16 Attopilot 2009 output 4Hz ($ATTO, $GPRMC, $GPGGA, $GPVTG)
32 Ardupilot output
64 ArduIMU output
128 RVOSD: display roll-corrected IMU/AHI angles (camera mounted on roll head)
256 Attopilot 2010 ($A1…$A4)
512 MavLink v1 QGroundStation (4Hz: #0 #32 #29 #35 #36 #72 #34 #28 #33 | 20Hz: #30)
1024 MavLink v2 QGroundStation (4Hz: #0 #32 #29 #35 #36 #37 #34 #28 #33 | 20Hz: #30)

Note: you can combine and send multiple protocols at once, but MavLink v2 replaces v1 if selected simultaneously.

Beacon Protocol output

A custom binary protocol with correction codes is available.
The name Beacon reflects the fact it is one-way protocol for reporting UAV situation.
Depending on protocol version, different binary encodings are possible:
BEACONv1 – output is ASCII-readable, end-line after every message, fixed bandwidth per message
BEACONv2 – pure binary protocol, more compact, token-delimited, slightly oscillating bandwidth depending on content (special token encoding)

OSD_SENDER_ID
Default: 1
Possible values: 0-255
Identifies a sender.

OSD_BEACON_PIDSEL
Default: 0
Possible values: 0-11
Selects a PID loop used for sending PID feedback loop tuning message.
0 send nothing
1 FB_RUDD2TURN
2 FB_TURN2HEAD
3 FB_THR2ALTI
4 FB_THRB2ALTI
5 FB_RUDD2ROLL
6 FB_ROLL2HEAD
7 FB_ELEV2ALTI
8 FB_ELEV2PITCH
9 FB_PITCH2ALTI
10 FB_ELEV2CLIMB
11 FB_CLIMB2ALTI
Note that the loops are usually used in groups, examples:
FB_RUDD2TURN + FB_TURN2HEAD
or
FB_RUDD2ROLL + FB_ROLL2HEAD
Unused PIDs will display steady output.

OSD_BEACON_BURSTLEN
Default: 32
Possible values: 1-32 or 129 to 160
While BEACON protocol slots in FLEXIPILOT allow a custom chain of up to 32 messages, this chain can consist of only 2 or 3 messages that need to be repeated over time. This variable defines the span. For example, BURSTLEN=2 means that actually only the 1st two nibble of OSD_BEACON_MSG1 will be used. Adding 128 to this value means each burst is packed and sent as a group during each autopilot iteration – this of course uses significantly more bandwidth and the data amount should be precisely calculated for given modem type.

Example:
OSD_BEACON_MSG1=0x4A000000
OSD_BEACON_BURSTLEN=2
OSD_BEACON_BURSTSPACING=1
Iteration1: send packet 0x04
Iteration2: send packet 0x0A
Iteration3: nothing
Iteration4: send packet 0x04
Iteration5: send packet 0x0A
Iteration6: nothing
…

OSD_BEACON_MSG1=0x4A000000
OSD_BEACON_BURSTLEN=130 *(note: 2+128)*
OSD_BEACON_BURSTSPACING=1
Iteration1: send packet 0x04 & 0x0A
Iteration2: nothing
Iteration3: send packet 0x04 & 0x0A
Iteration4: nothing
…

OSD_BEACON_BURSTSPACING
Default: 0
Possible values: 0-36000
Using specific burst spacing, you can fine-tune bandwidth usage and update rate of BEACON message chain to be sent. At 20Hz, *36000* is 30min. The actual time spacing may change with different autopilot main loop frequencies.

OSD_BEACON_MSG1
OSD_BEACON_MSG2
OSD_BEACON_MSG3
OSD_BEACON_MSG4
Possible values: 0-0xFFFFFFFF

There are 15 possible BEACON messages. Each message ID can be written as hexadecimal 0x1…0xF. One digit represents one message sent during 1/20th of a second following the scheme:

Example	**OSD_BEACON_MSG1**	**OSD_BEACON_MSG2**	**OSD_BEACON_MSG3**	**OSD_BEACON_MSG4**
Message 1 Every 1s	**1000 0000**	**0000 0000**	**0000 0000**	**0000 0000**
Message 1 every 1s, message 15 every 0.5s	**1000 0000**	**F000 0000**	**0000 0000**	**F000 0000**
All messages	**1020 3040**	**5060 7080**	**90A0 B0C0**	**D0E0 F000**

Note: the user has to ensure that enough bandwidth is possible. Wireless modem's baudrate (due to number of lost frames) drops significantly with distance.
It is possible to group the selection of packets using OSD_BEACON_BURSTSPACING.

As an indication,
BEACONv1 messages 1 & 2 at 1Hz are using 940bps of modem bandwidth
BEACONv1 messages 1 to 13 at 1Hz are using 6360bps of modem bandwidth
*Note: you can verify actual values pressing '**r**' on the console and check USARTGPS_sendbuf_bps, then adjust message order and composition in order to achieve optimal bandwidth.*

BEACONv2 messages 1 to 13 at 1Hz are using approx. 3150bps of modem bandwidth
BEACONv2 messages 1 to 13 at 20Hz (packed) are using approx. 63250bps of modem bandwidth

Example:
@@@OSD_BEACON_MSG1=0x12345678
@@@OSD_BEACON_MSG2=0x9ABCD000
@@@OSD_BEACON_MSG3=0
@@@OSD_BEACON_MSG4=0
@@@OSD_BEACON_PIDSEL=3 //thr2alti
@@@OSD_BEACON_BURSTLEN=13
@@@OSD_BEACON_BURSTSPACING=0
...will send all types of messages in succession. If additionally OSD_BEACON_ BURSTLEN = 141 *(13+128)*, the whole selection will be sent in a single pack, at 20Hz rate since OSD_BEACON_BURSTLEN=0.

Beacon message IDs

0 Ping/heartbeat mark (not sent by the autopilot)
1 Situation (position&speeds)
2 Orientation (angles)
3 Navigation (waypoints and their positions)
4 PID – feedback control loops – values
5 Meteo report estimated by autopilot
6 DateTime from GPS or RTC
7 MissionSafety (endurance estimations, heading to best home and takeoff positions)
8 Performance (airframe performance factors)
9 Engine #N status
10 Formation flying data, publishing own navigation plans
11 Stabhead – stabilized head virtual look-at position
12 Impact point (when gliding or using a parachute)
13 Reception quality

OSD_BANDWIDTH
Default: 0
Possible values:
0 – 921600bps (local USB or unlimited)
1 – 115200bps (short range, typically 2.4GHz)
2 – 9600bps 100% duty cycle (long range 900MHz USA)
affects protocols ARDU, ARDUIMU, ATTO2010
3 – 9600bps 10% hourly duty cycle (long range 868MHz EU)
affects protocols GARMIN, ARDU, ARDUIMU, ATTO2010
This variable affects specific protocols and lowers their update rate accordingly in order to meet maximum bandwidth requirements for specific modem types. For example, GARMIN protocol with setting 3 under EU law is divided into 3 portions at 1Hz each – this is less data than from standard 1Hz GARMIN receiver.
This setting affects GARMIN, ARDU, ARDUIMU and ATTO2010 protocols.
BEACON protocol bandwidth can be finely adjusted using OSD_BEACON_MSGn.
RVOSD protocols are local (not designed to be connected to the modem)
and consume the whole 115200bps bandwidth – this cannot be changed.

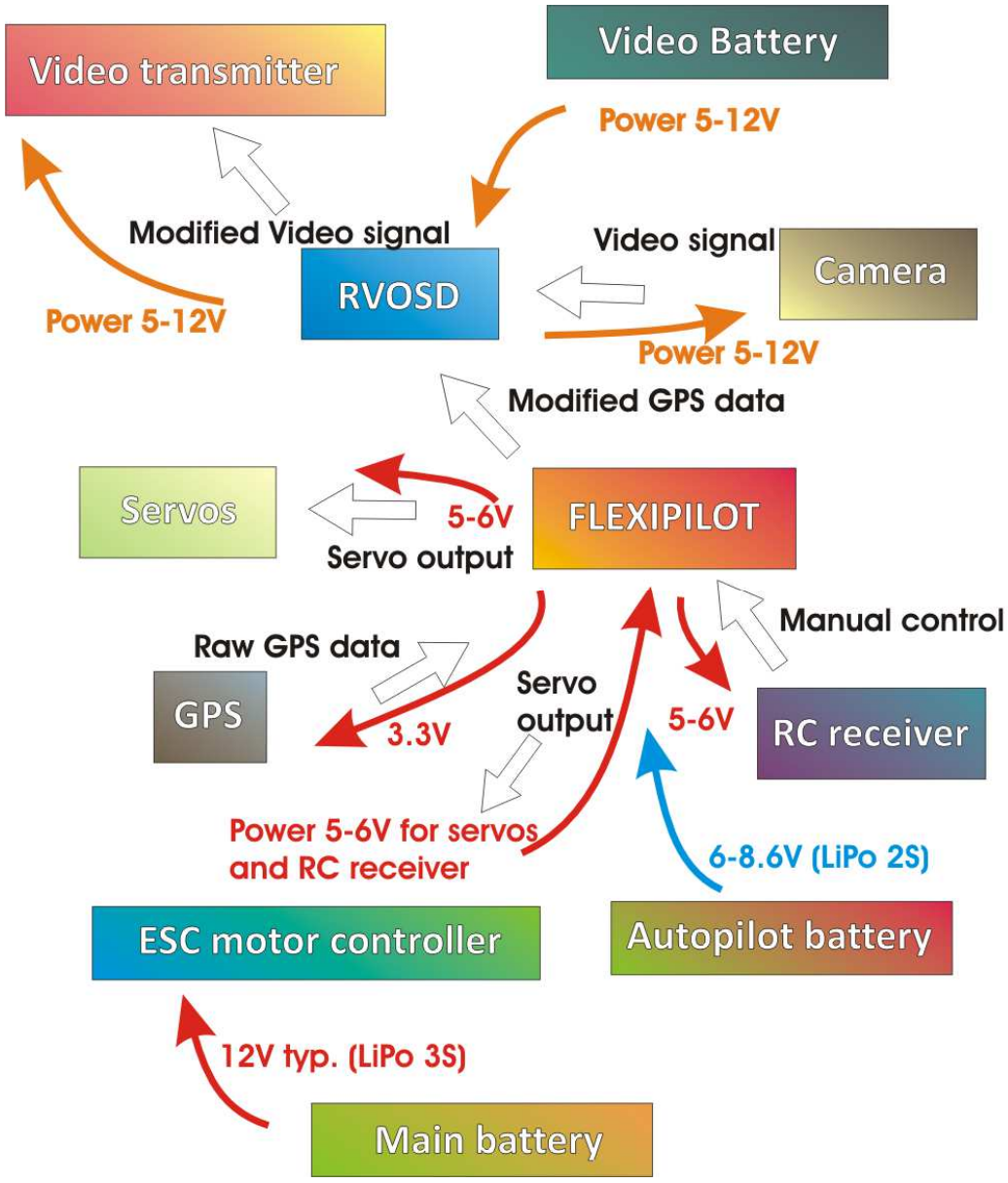
MAVLINK protocol is designed for short range use and its bandwidth is in theory unlimited, the FLEXIPILOT emits MAVLINK at around 14Kbps emitting each supported message at around 4Hz.

OSD_ACTIVE_PERIOD
Default: 1
Possible values: 0-3600s (0s – 1h)
OSD_INACTIVE_PERIOD
Default: 1
Possible values: 0-3600s (0s – 1h)
The two parameters define modem duty cycle. The modem power supply is powered off during inactive cycle. Periodically powering off the modem can:
-increase battery life up to several days (ACTIVE_PERIOD=5s, INACTIVE_PERIOD=55s)
-assure that EU regulated 10% duty cycle limit will not disturb the transmission (ACTIVE_PERIOD=59s, INACTIVE_PERIOD=1s)
-Increase modem reliability in case of hang-up (ACTIVE_PERIOD=59s, INACTIVE_PERIOD=1s)
-Disable modem (ACTIVE_PERIOD=0, INACTIVE_PERIOD>0)
-Keep modem enabled (ACTIVE_PERIOD>0, INACTIVE_PERIOD=0)

Note: the modem can be also powered down/silenced using RADIOSILENCE trigger flag at specific waypoint - until the next waypoint is met; consult navigation manual. After re-enabling at waypoint, the state of ACTIVE_PERIOD starts counting from 0.

Power layout

The following diagram shows the safest connection, where both video and autopilot domains are powered independently. Also, the autopilot logic is using completely separate battery.

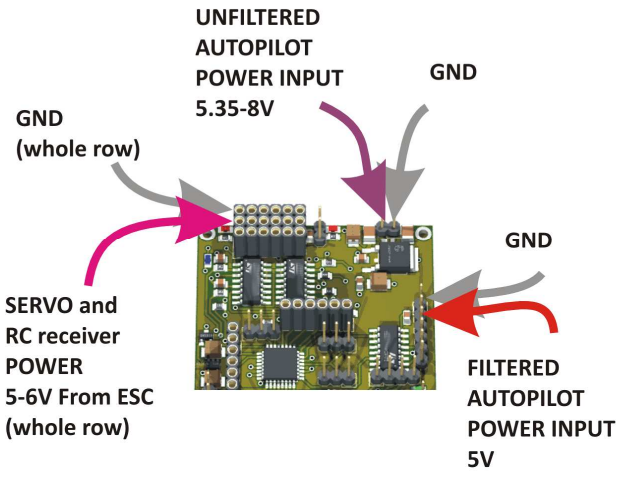


You can observe the following rules:
-RVOSD is used only for display and power filtering purposes, but the autopilot is using all its advanced capabilities
-Power domains are separate
-Never send the power from ESC to the autopilot logic power,
as the voltage is too unstable when the servos are moving

Optimizing electrical connections for reduced weight

In order to save weight, 3 solutions are possible:

1. Remove autopilot battery, and use step-down switching converter between Main battery and autopilot Unfiltered Power Input, providing at least 5.35V. This will be efficiently filtered by linear low-drop regulator onboard the autopilot and the operation is 100% safe, also step-down regulator is very efficient in suppressing motor noise, not letting it enter the autopilot. Use voltages slightly above 5.35V for power usage efficiency.
2. Main battery and Video battery can be the same, as RVOSD has power filtering capabilities.
3. Once RVOSD uses main battery and there are no noise issues, you can also try powering the autopilot using efficient switched power regulator from RVOSD. You can completely bypass the autopilot's voltage regulators and provide clean 5V for the logic using middle pin of the Auxiliary Input in RVOSD (please verify the possibility in current version RVOSD manual)



Special commands

You can simulate the whole flight at home. However when the GPS has problems obtaining lock, use **@@@ SIMOK_OSDGPS** command which will enable artificially correct lock for the OSD output so you can launch the usual flight simulation without using the RC transmitter with the commands:

@@@SIMOK_OSDGPS
@@@RXOVROFF
@@@SIMENABLE

For simulation purposes you might want to disable GPS position scrambling and enable the full RVOSD output:

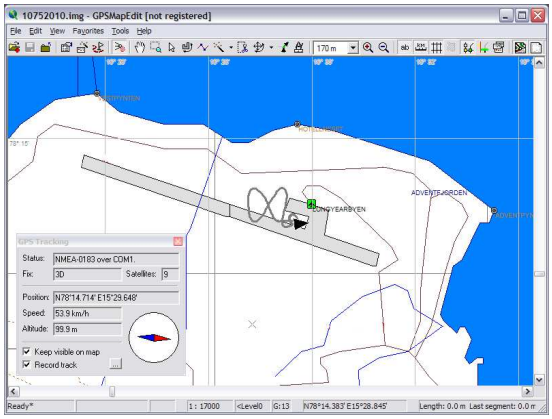
@@@OSD_OBFUSCATE=0
@@@OSD_PROTOCOL=4

Using OSD connector for realtime simulation

The OSD connector is a powerful tool for simulating the flight. As discussed in FLEXIPILOT navigation manual, you can launch the simulation and observe the situation on the OSD. However, you can use the level translator (from TTL to RS-232 levels) and RS-232 cable in order to connect this output to your PC. Then, you can use the extra port as generic GPS input to any GPS mapping software.
As an example we can use the following settings:

Garmin GPSMapEdit

GPS_BAUDRATE=4800
GPS_RELAY_TX=0
OSD_PROTOCOL=1 //Garmin 1Hz
SIM_TAKEOFF_LAT=78.245416
SIM_TAKEOFF_LON=15.494785
Using **GPSMapEdit** software in order to simulate the flight using free maps at exotic latitudes over 75 degrees, which are not allowed by MapSource software itself (yet vector-based MapSource maps support them).



Google Maps with GPSTracker

GPS_BAUDRATE=115200
GPS_RELAY_TX=0
OSD_PROTOCOL=1 //Garmin 1Hz