

Kolibri Manual v0.1.0 (pre-release alpha etc etc. Longboi/Pyrocket edition)

yeah.... The plan was to make everything nice and documented with a "proper" user manual, but as expected this is just hastily written for a similar hastily written firmware... It "works" but it ain't pretty.

Anyways, this manual will go over firmware flashing, configuration, and operation of Kolibri rev1.1 for use on Longboi/Pyrocket.

Hardware, rocket side

In terms of hardware, you need:

- Soldered Kolibri Rev1.x (using active buzzer)
- Arming switch/key
- Battery connector
- 2s LiPo battery (500 mAh or so).
- L80RE GPS module (Optional, only for longboi)
- Breakwire
- h3lis331dl High G accelerometer
- Pyro simulator (LED-based)
- Antenna (wire) or SMA antenna **(Always keep this connected to be safe and avoid frying the RF module)**

Additional hardware to complete the setup (Just grab the Kolibri ground station kit for the launchday)

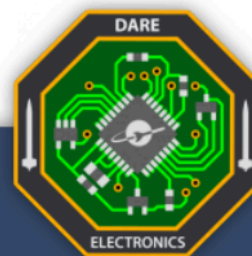
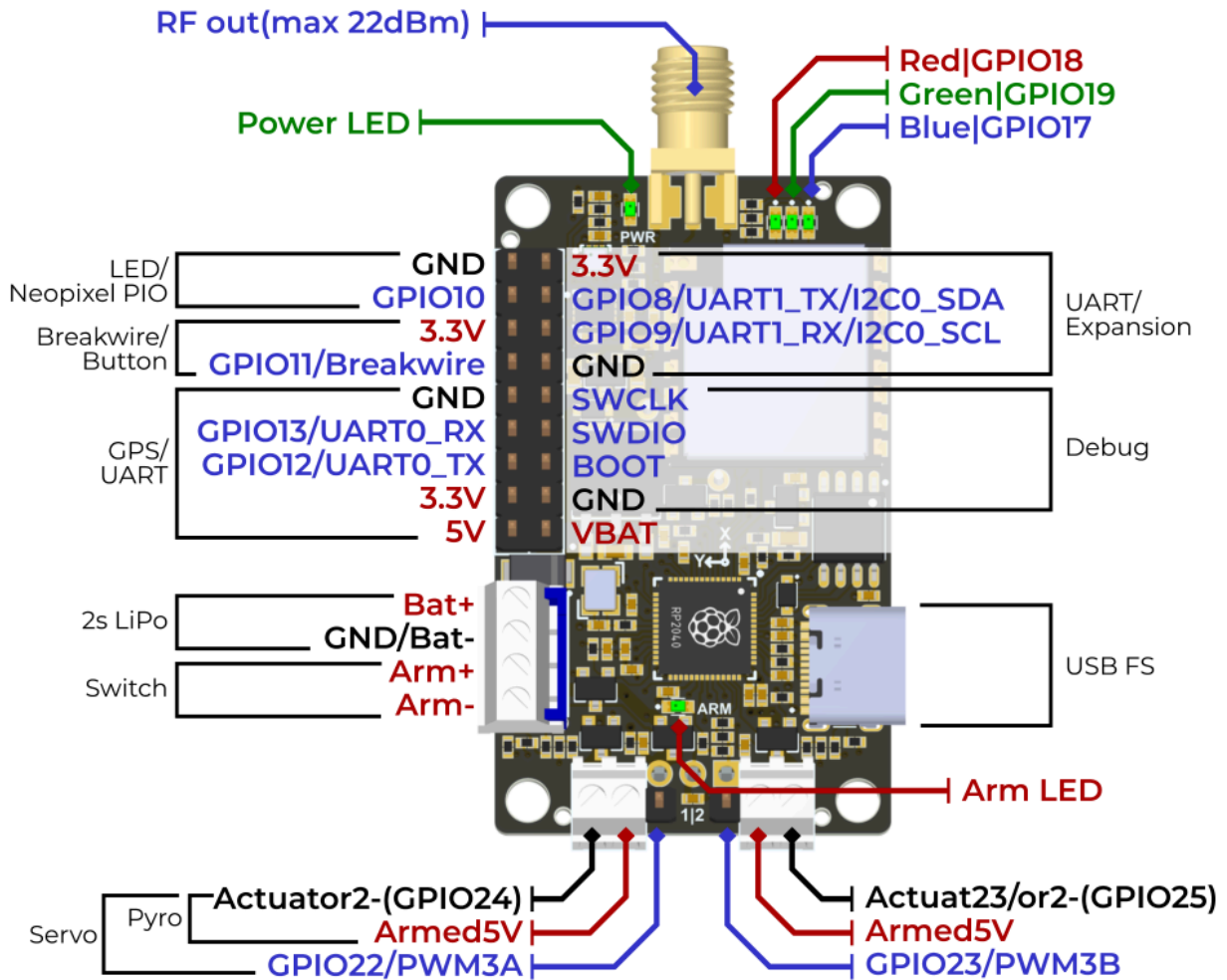
- Ground station receiver (Either OLED ground station or another Kolibri with an antenna connected)
- Tripod for receiver
- USB C to USB C cable
- USB C to USB A cable
- Laptop

Pinout and connections

This is the pinout, can be nice to print out a couple of these sheets:

KOLIBRI

Rev. 1.x
connection
diagram



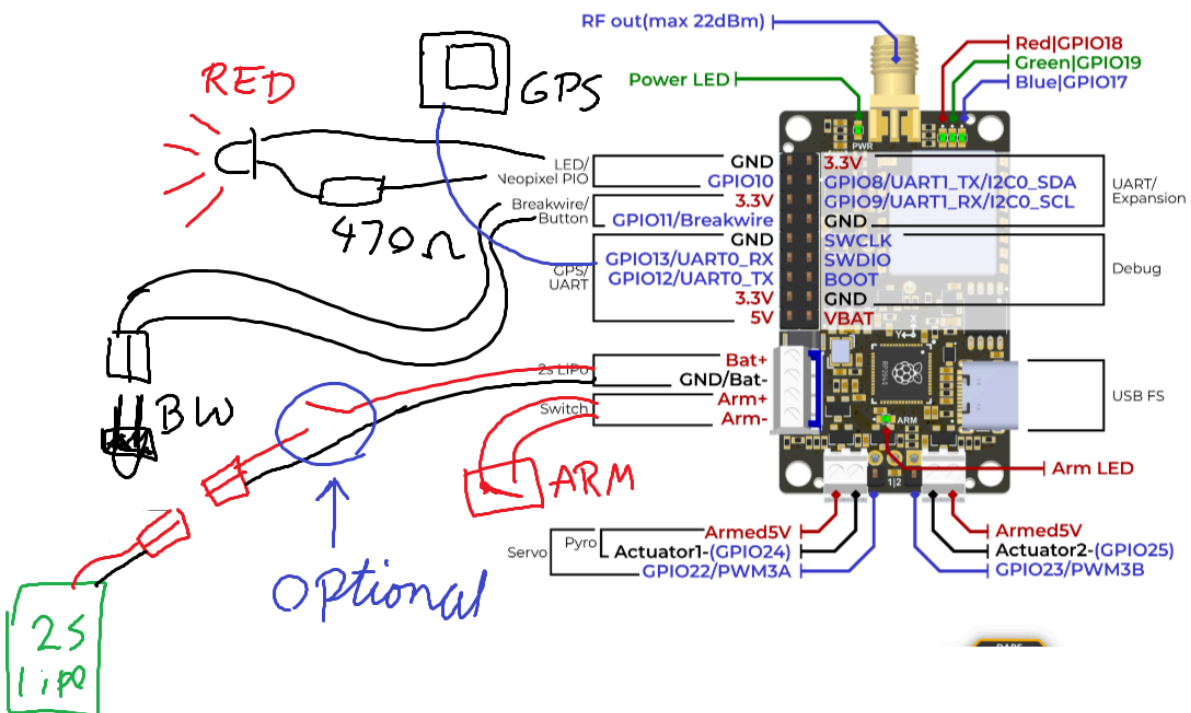
[pinout_r1.pdf](#)

The connection diagram is sketched below. Not shown is the high G accelerometer, which connects to the four pins labelled "UART/EXPANSION". GPS and the external LED are optional.

When referring to the arming switch, "ARM" refers to completing the arming connection while "DISARM" refers to breaking the connection.

Note: On Pyrocket and Longboi, connect the pyro to the SECOND actuation channel (Actuator2). This was due to my test setup using channel 1 for servo and channel 2 for pyro so that's what the flight code uses for now.

Required software



- Serial terminal emulator.
 - This tutorial uses Arduino IDE (Legacy)
Download the **Legacy IDE (1.8.X)** from here: <https://www.arduino.cc/en/software>.
 - For configuration using an android phone, install Serial USB Terminal:
https://play.google.com/store/apps/details?id=de.kai_morich.serial_usb_terminal&hl=nl
 - PuTTY: <https://www.putty.org/> (the arduino one works better)
- Serial Studio. Sorry Jesse, had an existing config for this and not (yet) for dare-mcs: <https://serial-studio.github.io/>

Make sure to have these installed.

Flash firmware

(WARNING: DO NOT FLASH A BOARD WITHOUT AN ANTENNA MOUNTED)

There are two versions of the firmware. One is for the ground station and one is for the flight board. The firmwares are "uf2" files. Make sure the indicated version of the flight firmware and ground station firmware match. You can find the firmwares here:

[kolibri_flight_0.1.1.uf2](#)

[kolibri_groundstation_0.1.1.uf2](#)

OLD VERSIONS WITHOUT DATA LOGGING:

[groundstation_v0.1.0.uf2](#)

kolibri_flightcode_v0.1.0.uf2

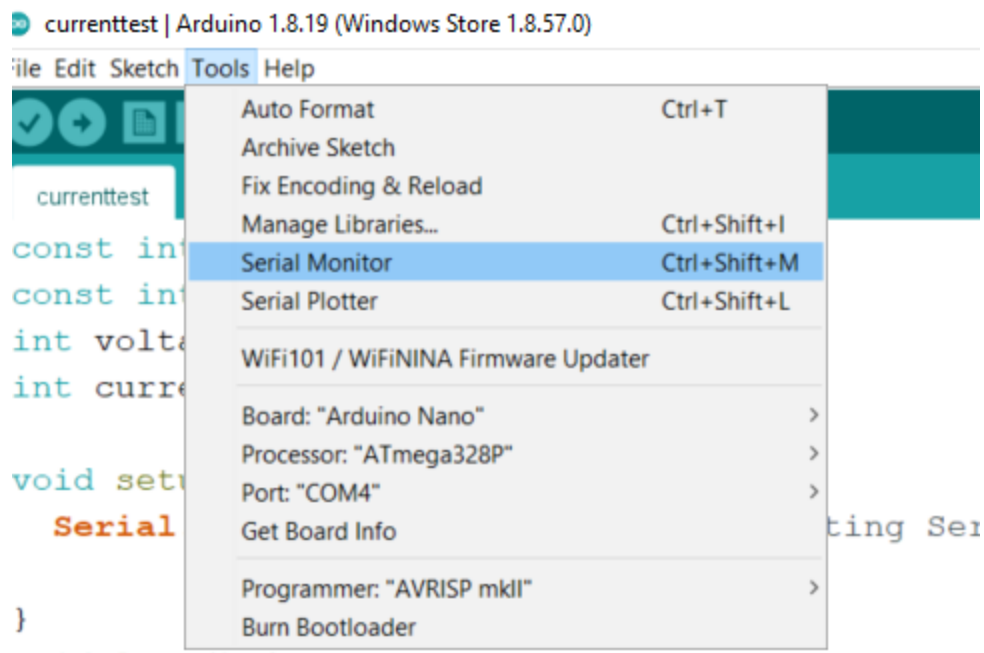
In order to flash the firmware, you need to put the board into "Bootloader" mode.

Manual way (always works):

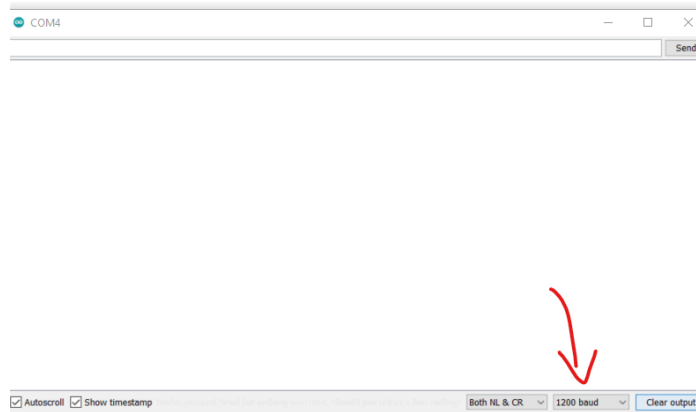
- Disconnect battery and USB power
- Grab a jumper or wire and bridge the "BOOT" pin to "GND"
- While bridged, connect USB to a laptop.

Software way (doesn't work if firmware is bad):

- Connect Kolibri to laptop
- Open Arduino IDE (Legacy)
- Select the Port corresponding to kolibri
- Open the serial monitor



- Change baud rate to 1200 (magic number)



- Or run the below python script which does the same thing

Either method should restart kolibri into bootloader mode, which mounts the board as a storage drive. In order to update the firmware, copy the correct UF2 file to the drive.

[usb-to-bootloader.py](#)

Warning: after updating the firmware, make sure to check that timer and configurations are correct.

Configure settings

The main settings you want to configure are the time for main deployment.

- Open the serial terminal
- Open Arduino and select the right com port
- Select 9600 baud

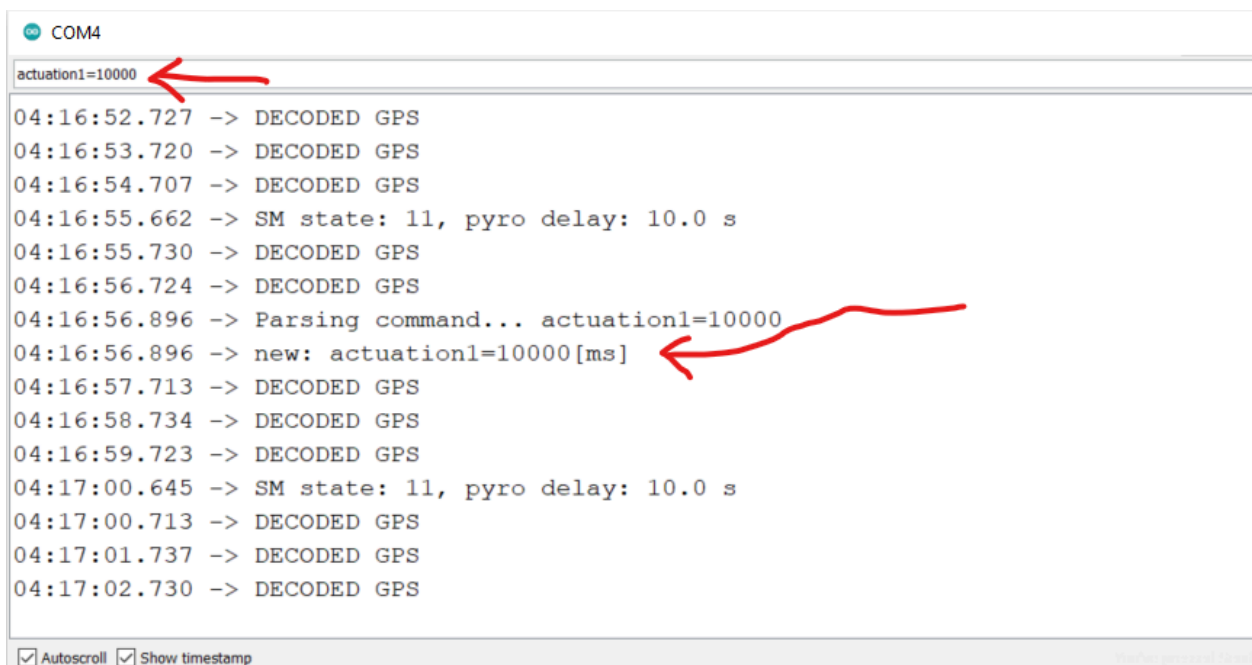
Type "?" followed by ENTER to see a list of commands. The ones you would want are:

- actuation1=13000 (set timer to 13 seconds)
- serial=6 (Example, use value written on back of board)
- readout (readout the blackbox data as hex formatted lines)

- readout_pretty (readout the blackbox data, formatted as plaintext)
 - The output is in CSV format, copy this over to a text file and use the following header:

```
time_ms, state, vbat, logging_percent_used, serial, baro_pressure_pa, baro_temperature_c,
baro_altitude_m, imu_acc[0], imu_acc[1], imu_acc[2], high_acc[0], high_acc[1], dummy, actuator2_enabled,
actuator2_continuity, actuator2_on, armed, actuator2_state, imu_gyro[0], imu_gyro[1], imu_gyro[2]
```

- erase_im_serious (erase the blackbox data. Beeps for 2 minutes until erase is complete, after which it restarts)

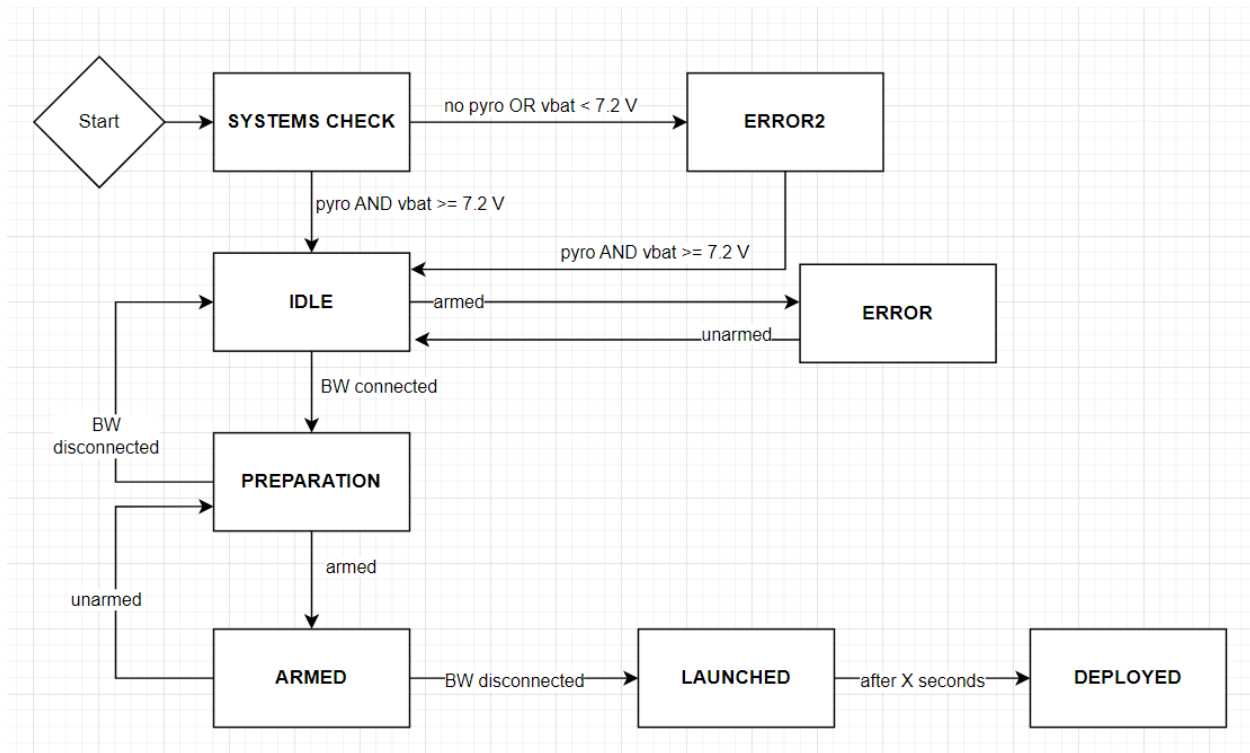


```
COM4
actuation1=10000
04:16:52.727 -> DECODED GPS
04:16:53.720 -> DECODED GPS
04:16:54.707 -> DECODED GPS
04:16:55.662 -> SM state: 11, pyro delay: 10.0 s
04:16:55.730 -> DECODED GPS
04:16:56.724 -> DECODED GPS
04:16:56.896 -> Parsing command... actuation1=10000
04:16:56.896 -> new: actuation1=10000[ms]
04:16:57.713 -> DECODED GPS
04:16:58.734 -> DECODED GPS
04:16:59.723 -> DECODED GPS
04:17:00.645 -> SM state: 11, pyro delay: 10.0 s
04:17:00.713 -> DECODED GPS
04:17:01.737 -> DECODED GPS
04:17:02.730 -> DECODED GPS
```

☒ Autoscroll ☒ Show timestamp

State machine

The state machine controlling recovery deployment is mostly based on the SRP state machine with some minor changes.



States and status beeps

Num	State	Description	Buzzer indicator
1	system_check	Just booted up	one blip
2	idle	Breakwire disconnected and pyro connected	two blips
3	preparation	Breakwire connected	Three blips
4	armed	Breakwire connected and armed	Four blips
5	launched	Breakwire disconnected, timer enabled	Fast beeps, once per second
6	deployed	Pyro charge has deployed	Alternating fast and slow beeps
7	landed	Changes to "landed" state 120 seconds after launch	Alternating fast and slow beeps
10-11	error	Either low battery voltage or no pyro continuity.	Long beeps (2 seconds)

Testing/launch procedure

The following sequence tests the main state machine of Kolibri for launch detection and deployment.

1. Gather everything
2. Make sure battery is disconnected
3. Measure battery voltage with multimeter, should be >8V (kolibri gives error under 7.2V)
4. Make sure breakwire is disconnected
5. Make sure pyro is not connected
6. Make sure board is DISARMED
7. Make sure antenna is present
8. Make sure the internal wiring are correct:
 - a. Breakwire connector is attached to Kolibri
 - b. Battery and arming connectors are attached to kolibri
 - c. High G accelerometer board is attached to Kolibri
9. Connect the battery, should get into an error state (no pyro continuity)
10. Connect pyro (or pyro simulator): Should get into idle state
11. Connect breakwire: Should get into preparation state
12. ARM the board. Should be in armed state
13. Disconnect breakwire. Should be in launched state
14. Wait until deployment (check if it matches configuration time)

Telemetry and tracking

Telemetry and GPS tracking work in parallel with the main launch code. So even if there's an error or if GPS is unconnected, the main deployment code still works.

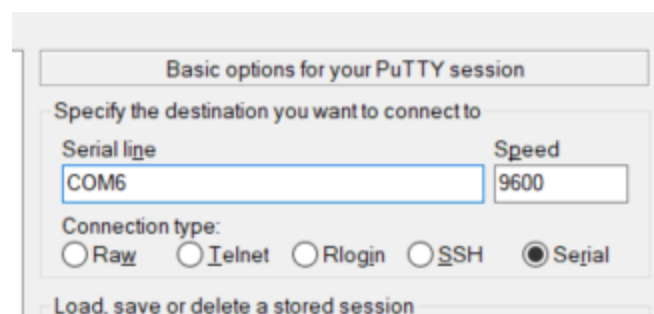
Currently two radio channels are available, channel 1 and channel 2. Channel 1 will be used for Pyrocket and Longboi, so make sure they don't fly at the same time 😊 (change if not the case)

Telemetry can be used to monitor the status of the state machine and onboard sensors.

In order to test the telemetry, the following serial studio config file can be used.

[kolibri_v0_1_0.json](#)

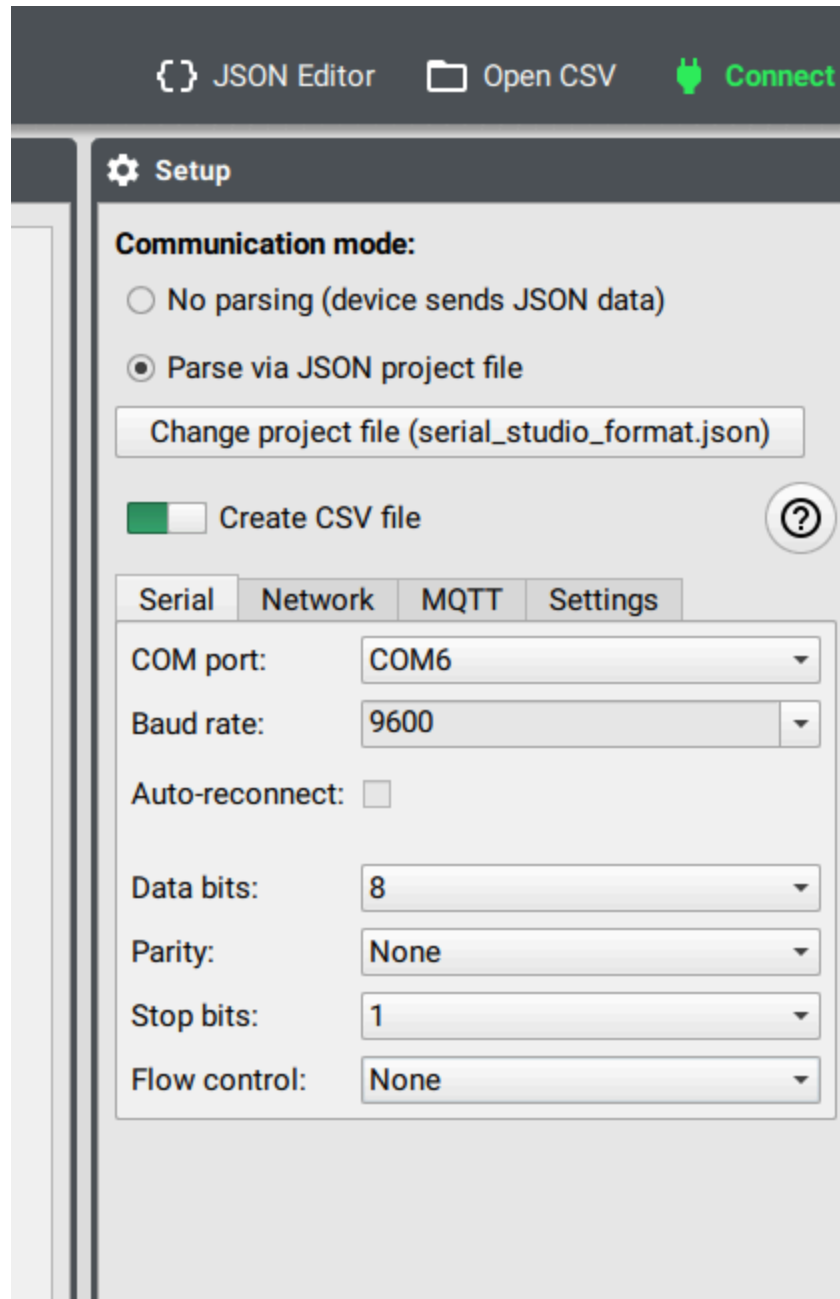
- Connect the ground station to a computer
- Open Serial studio (**v1.1.7**)
- Select the right com port, with 9600 baud
- Hit connect, status data should show up in the terminal. If no data appears, try this:
 - Open PuTTY (or arduino IDE)
 - Select Serial and the matching COM port (e.g. COM6) with 9600 baud
 - Connect and disconnect putty/arduino ide. Then try connecting in Serial studio again. This will send the signal for kolibri to start ending data.



- Connect power to flight Kolibri (with antenna)
- After a few seconds, the dashboard should show up with the telemetry data

- Afterwards, make sure to save the log somewhere.

Serial studio settings:



Should look something like this:



Note that the telemetry packets contain. The raw telemetry packets are sent to the serial terminal encoded as HEX, so even unsupported packages will be saved.

Data logging

Similar to telemetry, data logging works concurrently with the flight code. Data can be read out using the configuration commands over USB as shown above. Save this log somewhere

When launch is detected, data logging logs for 120 seconds at 100 Hz.

When not in flight, data logging occurs at 10 Hz.