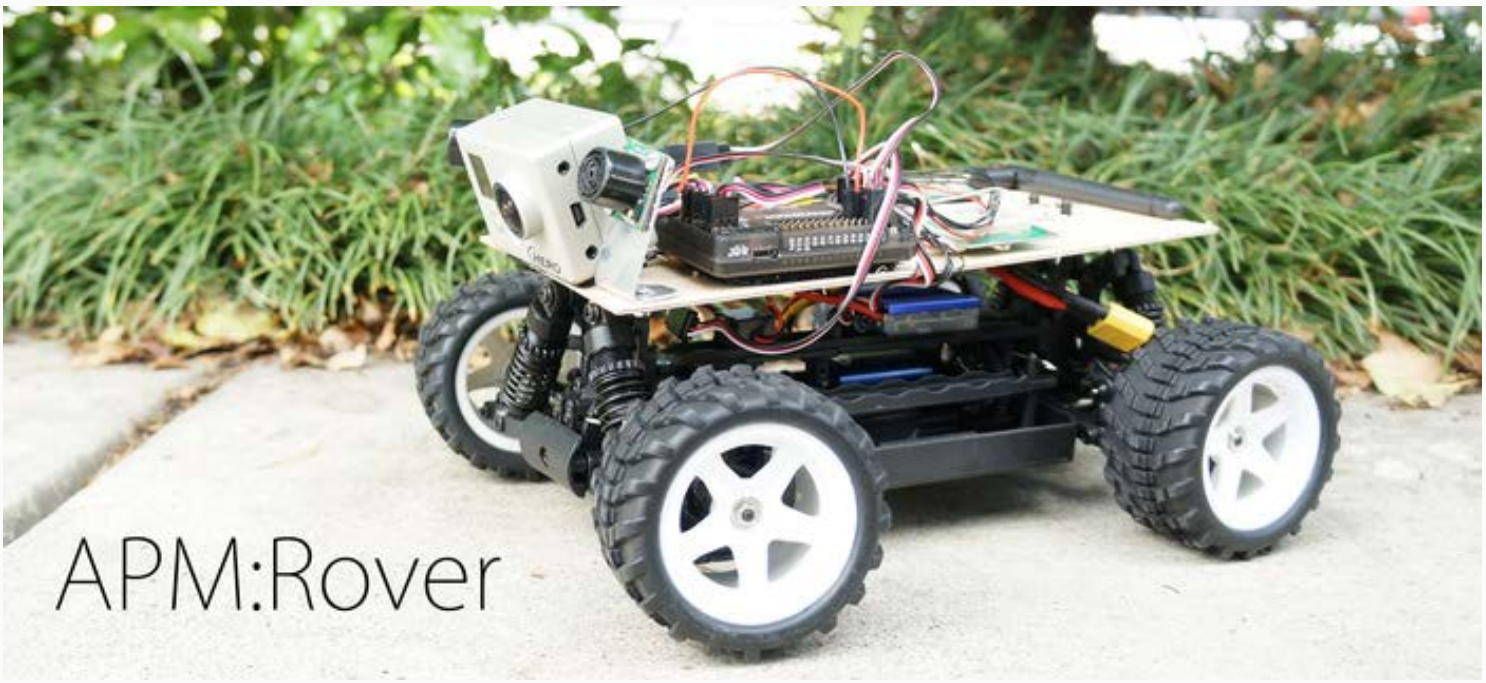
Rover

Rover Home

Tip

The ArduPilot Developer Ecosystem is Evolving! Find out more here …



Rover is an advanced open source autopilot for guiding **ground vehicles** and **boats**. It can run fully autonomous missions that are defined using mission planning software or pre-recorded by the driver during a manual run.

Rover runs on the Pixhawk and a number of other supported autopilot boards.

This is the platform that won the 2013 and 2014 Sparkfun Autonomous Vehicle Competition! That said,

Rover is far more than just a toy - it is already being used in numerous applications (you can see a few examples including agricultural automation in our Video Demos).

Table of Contents

Introduction to Rover

This article describes the main components of a Rover system, including the ground vehicle, autopilot hardware, and the software/firmware.

Rover Components

While it is is possible to build a vehicle with Rover from scratch we highly recommend starting from an existing RC vehicle (this comes with a frame, escs and power supply). You will need to add the autopilot, GPS, and possibly some other hardware. Most importantly you will need to obtain an RC

Transmitter that has spare channels needed for autopilot mode control and learning.

We like cheap 4-wheel-drive off-road vehicles with brushless motors and no RC (the RC units that come with most ready-to-run RC cars don't have the spare channel needed for autopilot mode control and learning). One such good platform is this RC off-roader from HobbyKing, which is just $72 (don't worry, we won't use the goofy plastic body shell):

Autopilot Hardware

Pixhawk is highly recommended for general use.

Developers creating robot vision applications should consider using a separate Companion Computer, or a Linux based autopilot board (e.g. NAVIO+ or Erle-Brain) which is capable of running both Rover and the image processing code.

For more options, see the topic Choosing a Flight Controller.

Note

You will need at least four female-to-female servo extension cables to connect the autopilot to your RC receiver (choose length to suit)

4+ channel RC transmitter and receiver

You'll need a radio control transmitter to manually control your Rover and to activate its control modes. You can use any RC transmitter/receiver system with at least 4 channels.

Don't get one designed for cars (with a steering wheel and throttle trigger); we won't be driving the

Rover manually much at all. Ideally, it will have at least two toggles switches, and one of those switches will have three positions. If you're on a budget, the Turnigy 9x ($54) is a popular choice. If you'd like better quality, we like the Taranis FrSky Reciever.

Some other options are discussed in the topic Compatible RC Transmitter and Receiver Systems (Pixhawk/PX4).

GPS module

Your Rover will *require* a GPS module. The recommended module is 3DR UBlox GPS + Compass Module which also includes a compass. You can check out other GPS solutions here.

LiPo batteries and charger

You'll also need batteries and a charger. Almost any 2S (7.2v) LiPo under 2600 mAh will do, but the recommend one for the above buggy is this one. A simple LiPo charger like this one will work fine.

Optional hardware

Telemetry Radio

A telemetry radio allows your Rover to communicate with your ground station remotely using the MAVLink protocol. This allows you to interact with your missions in real time and receive streaming data from your vehicle's cameras and other components. This adds considerable convenience to your missions!

Sonar/IR Sensors

Sonar/IR sensors are recommended for obstacle avoidance.

Ready to Use Rovers

At time of writing, the only Ready-to-Run (RTR) Rover is Erle-Rover.

Erle-Rover: Ready to Use Rover from Erle-Robotics

This RTR Rover contains all the needed components for getting started: the frame, Erle-Brain Linux autopilot, 4 Channels 2.4Ghz RC, Power Module (to power up the autopilot), NIMH battery and charger.

Note

This Rover uses a powerful Linux autopilot that can run more computationally intensive operations than a traditional autopilot (like Pixhawk).

Ground Control Station

The (free and open source) Mission Planner is required if you're going to be loading new versions of Rover onto the autopilot controller, and for first-time tuning and calibration. It runs on a PC and can also be used for planning missions.

Once your Rover is configured, you may find it more convenient to choose a different ground station - running on the tablet, phone or computer of your choice. The main options are discussed in the topic Choosing a Ground Station.

Note

This wiki exclusively uses Mission Planner as the reference GCS.

Rover Project News

This topic contains milestones and announcements related to the Rover project.

June 19, 2015: Rover 2.50 released:

The ArduPilot development team has released version 2.50 of APM:Rover. This release is mostly a back-end improvement to ArduPilot but a few new features and bug fixes are included.

Re-do Accelerometer Calibration

Due to a change in the maximum accelerometer range on the Pixhawk all users must re-do their accelerometer calibration for this release.

Only 3D accel calibration

The old “1D” accelerometer calibration method has now been removed, so you must use the 3D accelerometer calibration method. The old method was removed because a significant number of users had poor experiences.

Changes in this release are:

- CLI_ENABLED parameter added so the CLI can now be accessed in Rover
- PID logging for the steering controller. It its now possible to graph what the P, I and D are doing as your driving the rover around to enable much better tuning of the vehicle.
- Transition from .pde file to .cpp files for improved development.
- GIT Submodules created for PX4Firmware, PX4Nuttx and uavcan git repositories for improved development.
- Followme mode now works for Rover
- GUIDED mode significantly improved. If you have a GCS which is in Followme mode if the user then changes mode with the RC transmitter to HOLD or anything else then the Rover will STOP listening to the
- Followme updated guided mode waypoints.
- When going into GUIDED mode the rover went into RTL - this is fixed.
- Added EKF_STATUS_REPORT MAVLink message
- 64-bit timestamps in dataflash logs
- Numerous EKF improvements
- Added support for 4th Mavlink channel
- Added support for raw IMU logging
- Updated Piksi RTK GPS driver
- Improved support for GPS data injection (for Piksi RTK GPS)
- The SITL software in the loop simulation system has been completely rewritten for this release. A

major change is to make it possible to run SITL on native windows without needing a Linux virtual machine. (thanks Tridge)

March 4, 2015: Rover 2.49 released:

Hi everyone. The ardupilot team is delighted to announce the release of version 2.49 of Rover.

In case anyone needs to manually download the beta release its here:

http://firmware.ardupilot.org/Rover/stable/

This release is a bug fix release with two important bugs found by Marco Walther - Thanks Marco!

The bug fixes in this release are:

- Fixed a sonar problem where objects to the left wouldn't be identified - thanks Marco Walther!
- Fixed the ordering of the AP_Notify call so the main indicator light would be correct on startup - thanks Marco Walther!

February 20, 2015: Rover 2.48 released:

Hi everyone. The ardupilot team is delighted to announce the release of version 2.48 of Rover.

In case anyone needs to manually download the beta release its here:

http://firmware.ardupilot.org/Rover/stable/

The ardupilot development team has released version 2.48 of Rover. This release is a bug fix release with some important bugs found by the users of ardupilot.

The changes in this release are:

- fixed a bug that could cause short term loss of RC control with some receiver systems and configurations
- allowed for shorter sync pulse widths for PPM-SUM receivers on APM1 and APM2
- fix an issue where battery reporting could be intermittent (thanks Georgii Staroselskii!)
- fixed a mission handling bug that could cause a crash if jump commands form an infinite loop (thanks to Dellarb for reporting this bug)
- improved support for in-kernel SPI handling on Linux (thanks to John Williams)
- support UAVCAN based ESCs and GPS modules on Pixhawk (thanks to Pavel, Holger and and PX4

dev team)
- Multiple updates for the NavIO+ cape on RaspberryPi (thanks to Emlid)
- Lots of EKF changes
- added support for MAVLink packet routing
- added detection and recovery from faulty gyro and accel sensors
- added support for BBBMini Linux port
- increased number of AVR input channels from 8 to 11
- auto-set system clock based on GPS in Linux ports
- added SBUS FrSky telemetry support (thanks to Mathias)
- Added AK8963 MAG support (thanks Staroselskii Georgii)
- Added support for second battery
- Auto formatting of SDCard if it cannot be accessed on startup
- A number of significant performance improvements for the PX4 platform

The most important bug fix is the one for short term loss of RC control. This is a very long standing bug which didn't have a noticible impact for most people, but could cause loss of RC control for around 1 or 2 seconds for some people in certain circumstances.

The bug was in the the AP_HAL RCInput API. Each HAL backend has a flag that says whether there is a new RC input frame available. That flag was cleared by the read() method (typically hal.rcin->read()). Callers would check for new input by checking the boolean hal.rcin->new_input() function.

The problem was that read() was called from multiple places. Normally this is fine as reads from other than the main radio input loop happen before the other reads, but if the timing of the new radio frame exactly matched the loop frequency then a read from another place could clear the new_input flag and we would not see the new RC input frame. If that happened enough times we would go into a short term RC failsafe and ignore RC inputs, even in manual mode.

The fix was very simple - it is the new_input() function itself that should clear the flag, not read().

Many thanks to MarkM for helping us track down this bug by providing us with sufficient detail on how to reproduce it. In Marks case his OpenLRSng configuration happened to produce exactly the worst case timing needed to reproduce the issue. Once Tridge copied his OpenLRS

settings to his TX/RX he was able to reproduce the problem and it was easy to find and fix.

A number of users have reported occasional glitches in manual control where servos pause for short periods in past releases. It is likely that some of those issues were caused by this bug. The dev team would like to apologise for taking so long to track down this bug!

The other main change was also related to RC input. Some receivers use a PPM-SUM sync pulse width shorter than what the APM1/APM2 code was setup to handle. The OpenLRSng default sync pulse width is 3000 microseconds, but the APM1/APM2 code was written for a minimum sync pulse width of 4000 microseconds. For this release we have changed the APM1/APM2 driver to accept a sync pulse width down to 2700 microseconds.

Auto format of SD Card

From time to time the SD cards in the PX4 autopilots get corrupted. This isn't a surprise considering what we do to them. Your all familiar with the windows "please unmount or eject your SDCard before removing" process. Well we don't do that. In fact normal operation is to just pull the power on the SDCard - whilst its being written too!! Not to mention the horrible vibration rich environment the SDCard exists in. If the autopilot is setup in the internal innards of your plane/copter/rover this can be a nightmare to get to. To resolve that problem Tridge has added code at startup so when ArduPilot tries to mount to SDCard to access it - if that fails it will then try to format the SDCard and if successful mount the card and proceed. If the format fails then you will get the usual SOS Audio that makes most of us want to find the buzzer and rip its heart out.

I mention this in case anyone has precious logs saved on the SDCard or they are using the SDCard out of their phone with their wedding photo's on it. Probably best not to do that and assume any data on the SDCard can be deleted.

We are also looking to add a parameter to control whether the card is auto formatted on startup or not but it isn't in there yet.

November 15, 2014: Rover 2.47 released:

This is a minor bug fix release. The most important change in this release is the fixing of the skid steering support but there have been a number of fixes in other areas as well.

Full changes list for this release:

- add support for controlling safety switch on Pixhawk from ground station
- prevent reports of failed AHRS during initialisation
- fixed skid steering that was broken in the last release
- report gyro unhealthy if gyro calibration failed
- fixed dual sonar support in CLI sonar test
- fixed Nuttx crash on Pixhawk with bad I2C cables
- added GPS_SBAS_MODE parameter - turns on/off satellite based augemtation system for GPS
- added GPS_MIN_ELEV parameter - specifiy the elevation mask for GPS satellites

- added RELAY_DEFAULT parameter to control default of relay on startup
- fixed bug in FRAM storage on Pixhawk that could cause parameters changes not to be saved
- better handling of compass errors in the EKF (Extended Kalman Filter)
- improved support for linux based autopilots
- added support for PulsedLight LIDAR as a range finder

Many thanks to everyone who contributed to this release, especially

Tom Coyle and Linus Penzlien for their excellent testing and feedback.

August 26, 2014: Rover 2.46 released:

This release is based on a lot of development and testing that happened prior to the AVC competition where APM based vehicles performed very well. Details here.

Full changes list for this release:

- added support for higher baudrates on telemetry ports, to make it easier to use high rate telemetry to companion boards. Rates of up to 1.5MBit are now supported to companion boards.
- new Rangefinder code with support for a wider range of rangefinder types including a range of Lidars (thanks to Allyson Kreft)
- added logging of power status on Pixhawk
- added PIVOT_TURN_ANGLE parameter for pivot based turns on skid steering rovers
- lots of improvements to the EKF support for Rover, thanks to Paul Riseborough and testing from Tom Coyle. Using the EKF can greatly improve navigation accuracy for fast rovers. Enable with AHRS_EKF_USE=1.
- improved support for dual GPS on Pixhawk. Using a 2nd GPS can greatly improve performance when in an area with an obstructed view of the sky
- support for up to 14 RC channels on Pihxawk
- added BRAKING_PERCENT and BRAKING_SPEEDERR parameters for better breaking support when cornering
- added support for FrSky telemetry via SERIAL2_PROTOCOL parameter (thanks to Matthias Badaire)
- added support for Linux based autopilots, initially with the PXF BeagleBoneBlack cape and the Erle robotics board. Support for more boards is expected in future releases. Thanks to Victor, Sid and Anuj for their great work on the Linux port.
- added StorageManager library, which expands available FRAM storage on Pixhawk to 16 kByte. This allows for 724 waypoints on Pixhawk.
- improved reporting of compass (magnetometer) and barometer errors to the GCS
- fixed a bug in automatic flow control detection for serial ports in Pixhawk

- fixed use of FMU servo pins as digital inputs on Pixhawk
- imported latest updates for VRBrain boards (thanks to Emile Castelnuovo and Luca Micheletti)
- updates to the Piksi GPS support (thanks to Niels Joubert)
- improved gyro estimate in DCM (thanks to Jon Challinger)
- improved position projection in DCM in wind (thanks to Przemek Lekston)
- several updates to AP_NavEKF for more robust handling of errors (thanks to Paul Riseborough)
- lots of small code cleanups thanks to Daniel Frenzel
- initial support for NavIO board from Mikhail Avkhimenia
- fixed logging of RCOU for up to 12 channels (thanks to Emile Castelnuovo)
- code cleanups from Silvia Nunezrivero
- improved parameter download speed on radio links with no flow control

January 26, 2014 Rover 2.45 released.

Details on Forum Here and the associated Blog Here.

Rover Developer Test Track and Typical Mission Test Course

Good news from Andrew Tridgell and the rest of the Rover (Rover) Development Team: they've pushed out a major new release (2.45). Major changes include: numerous updates for PX4 and Pixhawk, dual sensor support on the Pixhawk and reduced GPS latency for all three navigation controllers.

The following is from Tridge's release announcement. Please post any questions or comments there.

Pixhawk updates¶
The biggest set of changes are for the PX4 and Pixhawk boards. The Pixhawk production boards started shipping a few weeks ago and include some nice hardware features that didn't make it into the 2.44 release. The most important of these is dual sensor support. The Pixhawk has two gyros and two accelerometers, and (if you have an external GPS/compass combo) dual compass as well. In the 2.44 release only one of each of these sensors could be used. With the 2.45 release the health of each sensor is monitored and if one sensor fails the second sensor can take over. In addition to failover support the code logs both sets of sensor values both to MAVLink and to the logs on the microSD card, which is very useful for diagnostics. We are also working on dual GPS support which we hope to get into the next release.

GPS navigation improvements¶
A very simple change to our GPS configuration made a big difference for this release. In previous rover releases we setup the uBlox as ENGINE_AUTOMOTIVE as we thought that would be the best match

for a rover. What we discovered recently is that using the AUTOMOTIVE mode led to much more time delay (lag) in the GPS velocity data, which led to poor handling of sharp turns. We have now switched to ENGINE_AIRBORNE_4G and found that we get much better navigation at higher speed with sharp turns.

Parameter storage¶

Another major change for this release is the way parameters are stored on PX4 and Pixhawk. In previous releases we stored parameters on a file on the microSD card. That usually worked fine, but recently there have been a few too many issues with FAT filesystem corruption of microSD cards, especially when powering off while writing to the SD card. For this release we have moved all parameters to the EEPROM on the PX4 and the FRAM chip on the Pixhawk. This makes parameter storage independent of the microSD card, avoiding parameters becoming corrupt due to microSD card problems. Parameters from a microSD card will be automatically copied to EEPROM/FRAM when you upgrade to Rover 2.45.

Improved relay code¶

The relay and servo set code has had a major overhaul, with up to 4 relays now supported for MAVLink control and much better support for the DO_SET_SERVO, DO_SET_RELAY, DO_REPEAT_SERVO and DO_REPEAT_RELAY MAVLink commands. Along with these changes is a new parameter BRD_PWM_COUNT which allows you to specify how many auxillary PWM outputs to enable, with the remaining outputs being available as digital relays. This allows you to re-assign some of the aux servo outputs on Pixhawk for use as relays, by setting the RELAY_PIN, RELAY_PIN2, RELAY_PIN3 and RELAY_PIN4 parameters. The pin numbers for these pins start at 50 for the first aux servo pin, and go to 55 on Pixhawk.

Improved logging¶

There have been logs of logging improvements in this release. Apart from the dual sensor logging, we can now transfer on-board log files to the ground station over MAVLink, which makes it much easier to get detailed logs without having to pull a microSD card out, or boot to the CLI. We're hoping to remove the need for the CLI completely in a future release, doing everything over MAVLink.

The new logging code also includes the git version number of APM and (if needed) the PX4Firmware and PX4NuttX repositories used to build the firmware in the logs. This makes it easier to track down any issues to the exact code used.

More telemetry ports¶

On Pixhawk and PX4 you can now have a 3rd telemetry port, allowing USB and two serial telemetry ports to be active at the same time, which is very useful when you have an onboard computer.

Regards,

TCIII Rover2 Developer

Rover 2.44 Released

- December 29 2013 Rover 2.44 released. Details here
- The Rover development team are delighted to announce the 2.44 release of the rover firmware.
- The main change in this release is better support for the new Pixhawk autopilot, but there are also a lot of smaller changes that improve the Rover code.
- The Pixhawk changes include support for dual sensors with failover, support for auto-update of the IO firmware and lots of small performance improvements.
- Other changes include:
  - Added support for the MAV_CMD_DO_SET_CAM_TRIGG_DIST MAVLink command
  - Added SKIP_GYRO_CAL option
  - Added support for MAVLink system time from GPS time
  - Improved GPS timing for AHRS
  - Lots of improvements to dataflash logging
  - Support for MAVLink log download
  - Support for reverse in STEERING mode
- Added support for 2nd telemetry port on Pixhawk
- This release is parameter compatible with previous releases, so it should be an easy update for all users.

Rover 2.43 Released

- October 10 2013 Rover 2.43 released. Details here
- Tom and Tridge have just released APMrover2 2.43 for your driving pleasure!
- This is a major new release with a lot of improvements, especially in the area of more precise steering and navigation. Highlights include:
- adoption of L1 navigation code from Plane
  - new more precise steering controller
  - support for the new Pixhawk autopilot
  - conversion to centimeter level precision throughout the code
- I'll only be able to give a rough summary of the changes here as we have had more than 1500 changes in total go into the git tree since the last release.
- Only some of those were related exclusively to the rover code, but a lot of them do affect rover.
- The most significant change is in the navigation controller.
  - Earlier this year the Plane code adopted a new L1 navigation system.
  - That was highly successful, and led to the fixed-wing plane code improving its navigation performance enormously, with much smoother and more accurate turns, and much better handling of wind, along with simpler tuning.

- Since then Tom and I have been working to bring the same benefits to the Rover code.
  - It was a bit of a rocky start, with initial attempts not working well on real rovers.
  - With a single set of tuning values we were able to get good results either on straight driving or in tight turns, but not in both with the same settings.
- We finally made progress when we wrote a new STEER2SRV steering controller that takes the output of the L1 navigation code (as a demanded lateral acceleration) and produces the steering controls.
  - The new controller was based initially on the fixed-wing plane roll controller, but adapted to the physics of wheeled vehicles.
  - That controller replaces the old classic PID based steering controller in the rover code, and provides much smoother and more adaptable steering.
  - After we got the tuning for it right, steering on both of our rovers is now much better than it has been previously.
- The new steering code has been so successful in fact that I have now ported it back for use in Plane to control the ground steering on planes when taxiing and in automatic takeoff and landing.
  - It has really helped with one of my planes that tends to be difficult to control during takeoff.
  - So Rover and Plane now use the same ground steering code, just with different tuning parameters.
- The next big change that you may notice in this release is the internal move to 32 bit integer positions instead of 32 bit floating point positions.
  - The reason is that 32 bit floats that represent a global latitude/longitude only have a precision of a bit over 1 meter in some parts of the world.
  - On a plane flying at 60 knots this isn't a problem, as 1m accuracy is great, but for a rover it is not good.
  - Switching to 32 bit integers gives us a precision of about 1cm, which is a good level for a rover.
  - This lowers the amount of weaving of the rover as it no longer tends to jump across 1m boundaries.
- That change to 1cm accuracy had impacts in a lot of different parts of the code.
  - For example, the code which takes account of GPS lag to predict the actual position of the rover needs to do its math carefully to avoid ending up with floating point rounding errors.
  - The NMEA GPS driver needed to be modified quite a lot to read the floating point numbers in the NMEA protocol in a way that preserves centimetre level precision.
- The centimeter precision change doesn't mean that every rover will now navigate with cm precision.
  - The rover will still be limited by the precision of its GPS and how good its steering linkages are.
  - But if you have a good (expensive!) GPS and good hardware then Rover should be able to get much better results with this release than it could previously.
- Support for the new Pixhawk autopilot is another major feature of this release.
  - Most of my testing for this release has been done with a Pixhawk and it has worked very well.
  - We still support the APM1, APM2 and PX4 of course, it is just nice to also work with the latest hardware.

    - We also are working on supporting Rover on embedded Linux boards like the BeagleBone black, and Mike McCauley is working on a port to the Flymaple board.
- For this release you will need to read the new navigation tuning guide for Rover as quite a few parameters have changed.
    - The full parameter list is also useful as a reference.
- There have been a lot of other smaller changes in this release. Here is a partial list of the more important ones:
    - added new AP_BattMonitor library for common battery handling with copter and plane
    - use GPS projected position based on estimated GPS lag for more precise waypoint turns
    - changed STEERING more to allow for throttle nudging
    - added RCMAP support for mapping RC channels in any order
    - switched to using new AP_Scheduler scheduling
    - enabled RELAY support
    - added LEARN_CH parameter for setting which channel triggers waypoint learning
    - support camera stabilization and camera mount controls
    - adopted new AP_Notify system for controlling LEDs on different boards
    - lowered default accel/gyro filter frequency to cope with rough surfaces (eg. grass)
- Tom and I wish you a lot of happy rovering with this release!
- Cheers, Tridge

Rover 2.42 Released

- July 1, 2013: Rover version 2.42 released. Details here
- This release contains numerous small improvements that have accumulated over the last couple of months.
- Many of the improvements came out of the period of intensive development and testing that Tom and I put into the code before the ACV2013 competition (which we were delighted to win!)
- The main improvements are:
    - fixed voltage calculation for RSSI receiver signal logging
    - use projected GPS position for waypoint calculations. This helps a lot for tight missions, such as around the barrels at AVC
    - disable cross-tracking over last 3 meters of each mission leg, to prevent fishtailing as a waypoint is approached
    - fixed change of speed command in missions (it was off by a factor of 100x, leading to some crazy speeds!)
    - fixed MAVLink logging of sonar distances
    - made it possible to change AHRS_ORIENTATION at runtime without a reboot
    - added new RCMAP_ parameters to allow for having RC input channels in any order
    - added new internal task scheduler for more accurate timing
    - fixed HIL simulation code

    - fixed handling of PX4 servo arming
    - changed the AUTO_TRIGGER_PIN to be push-on/push-off
- Meanwhile Tom and I are working towards adding the L1 navigation controller for the next release.
    - We didn't put that in this release as we aren't yet happy enough with the performance. We hope to get that fixed soon!
- The other thing that has changed since the last release is that we have switched to the new rover documentation site.
    - The docs have improved a lot, although there is still more to do!

Happy rovering!

Previous Releases

- May 15, 2013: Project retitled Rover to conform with the rest of the APM projects, which are now no longer just Arduino-based.
- May 6, 2013: Rover 2.41 released. Bug fixes and performance improvements. Details here
- April 6, 2013: Rover 2.40 released. Major enhancement of the Rover code, adding many performance enhancements and bringing it in sync with the latest Plane code and current APM and Mission Planner features. Details here
- June 16, 2012: Rover successfully finishes the Sparkfun Autonomous Vehicle Competition
- May 2, 2012: Jean-Louis Naudin ports APM code to rovers

Rover Video Demonstrations

This article shows a small number of videos demonstrating the Rover autopilot in real use use.

Tip

Please let us know if you find other great videos/use cases we can show here!

Ground Vehicles

Driverless tractor¶
Driverless automated grain cart¶
RC car¶

Boat

ArduBoat MK4 - RC Boat¶
This video originates from this DIYDrones blog post.

Rover Control Modes

This topic lists Rover's control modes.

Autopilot control modes are controlled through a radio transmitter switch, via mission commands, or using commands from a ground station (GCS) or companion computer.

For a 3 position switch initially set 5 and 6 to MANUAL, 3 and 4 to LEARNING and 1 and 2 to AUTO.

The control modes supported for Rover are:

MANUAL mode

Basic radio pass through drive your vehicle with the RC transmitter.

- Control steering and throttle manually.
- One dedicated mode is always Manual mode (Flight Mode 6).
- For all modes other than just driving around in Manual Mode you must have GPS lock and a Home position.
  - After vehicle is first powered up, GPS lock is achieved (when GPS blue LED solid on) and Home position is set to that location.
- Momentary toggle of (CH7) switch clears the waypoints list.
  - Steering hard right when CH7 switch is toggled resets resets the vehicles current position to the Home position.

LEARNING Mode

Drive your vehicle with radio pass through but additionally you can record waypoints for use in auto mode.

- Control steering and throttle manually.
- Momentary toggle (CH7) stores the current GPS position as a new waypoint.

There is more information on this mode in the topic Learning a Mission.

AUTO Mode

Follow a course of Waypoints generated in while driving in Learning mode or from a loaded waypoint list.

- Waypoint lists can also be generated with APM mission planner and uploaded to APM via the Flight Planner screen.
- Rover loops the waypoint lists stored in memory.
- Momentary toggle (CH7) triggers immediate RTL (return to launch point: Home position), enters HOLD mode after completing RTL.
- To regain manual control of the rover, switch back to LEARNING or MANUAL mode.
- In Auto mode if RC reception is lost, the controller will automatically switch to HOLD mode.

STEERING Mode

Learning mode (sort of) plus obstacle avoidance.

- Steering mode is a bit like learning mode, except that the controls use the same steering, throttle and obstacle avoidance code as AUTO mode does.
  - The steering controls the “navigation bearing”, which is what AUTO uses to navigate.
  - The throttle controls the target speed, just like AUTO does.
  - When it sees an obstacle it tries to avoid it in the same way that AUTO does.
  - It even does turn speed scaling.
- So you can use STEERING mode to test your rover while remaining fully in control.
- Drive it at a obstacle, and it will avoid it, which is not only fun, it’s great for tuning.
- Momentary toggle (CH7) stores the current GPS position as a new waypoint.

HOLD Mode

All Stop.

- When you switch to Hold mode, the vehicle should stop with its wheels pointed straight ahead.
- It is necessary to set your throttle (`RC3_Trim`) parameter to the correct position for your vehicles throttle off PWM.

- If you have a reversing vehicle with a center neutral point set `RC3_TRIM` parameter to 1500: (May need adjustment for your vehicle).
- If you have a non reversing vehicle RC3_TRIM can be left to it's default low (~1100 PWM) setting or adjusted to it if necessary.
- In Auto mode if RC reception is lost, the controller will automatically switch to Hold mode.
- If you enable Throttle Failsafe and set the Failsafe action to HOLD it will go to HOLD if radio reception is lost in all modes.
- Normally you do not have to install this as a switch option.
- If it failsafes to HOLD mode, switching to another mode and back will clear it.

GUIDED mode

- Does not require a dedicated switch, so don't select it as a flight mode.
- It may currently require the "flight altitude" to be set to zero.
- Allows you to use a **:ref:`Telemetry Radio <common-telemetry-landingpage>`** to specify waypoints on a map in mission planner.
- And "tell" the Rover to go to those waypoints.
- See the **:ref:`APMCopter Guided Mode Description <copter:ac2_guidedmode>`** for a overview of how it works.

Rover Setup and Configuration

This section provides a step-by-step guide for building an Unmanned Ground Vehicle (UGV) using the Rover autopilot running on Pixhawk, PX4 or APM 2.x.

Rover Assembly Instructions

This section contains the wiring instructions for assembling the flight controller and other *essential components* of Rover (for several flight controllers):

The instructions for adding other hardware are covered in Optional Hardware.

Mandatory Hardware Configuration

As part of first time setup, you'll need to configure some mandatory hardware components.

The linked articles explain how to calibrate the RC settings and compass, assign control modes to the remote control switches, and set the most important autopilot parameters.

In addition to mandatory calibration, you may also choose to Configure Optional Hardware.

Throttle Arming in Rover

A software safety feature that requires the throttle to be explicitly armed by a driver is now available for Rover. It was introduced in version 3.0.0 and it is enabled by default. The idea is to prevent an unexpected throttle up and having the rover randomly accelerate away. It also performs a number of system checks to ensure the Rover ArduPilot system is healthy and ready to go.

Parameters governing throttle arming are introduced in the ARMING section of the parameters wiki page. This page discusses throttle arming in greater detail and walks through the typical procedures for using this safety feature.

A Safety Note About Arming

A safety feature of PX4 / Pixhawk is that all servo output is kept at at minimum values until the safety button is pushed. This has the nice effect of (usually) disabling electric motors and it is quite possible to use only this safety feature for propeller safety. However:

1. This feature is not available for the APM.
2. Generally during pre-arm checks it is desirable to verify servo operation (e.g., to check steering is deflecting in the expected direction). This can more safely be done with the throttle disabled. When the safety button is pushed **all** servos are enabled, including the motor – unless this additional software throttle arming safety is used.

A note about APM1 and channel 8

The APM1 has a hardware multiplexor on channel 8 which forces pass-thru of the first 4 channels if channel 8 is above 1750. This means the arming code is bypassed if you use channel 8 for your mode switch. So if you want to use arming on an APM1 then you should change FLTMODE_CH to another channel (say channel 5) and setup your transmitter/receiver to put the mode switch on that other channel.

IMPORTANT: RC Transmitter Calibration

It is essential that your RC radio transmitter be calibrated correctly before continuing. Please see the Calibrate your RC input wiki page if you don't know how to calibrate your radio.

Warning

During RC calibration throttle ups are possible, even with safeties enabled and you should therefore ensure the rover's wheels are not touching the ground when you do the calibration.
Note that if you have RCMAP_THROTTLE set to something other than 3, then the RCn_MIN etc values used will be the one for the channel you have selected as the throttle channel.

When calibrating your RC input you should also be careful to set the deadzone (DZ) value of the throttle (usually RC3_DZ) to a non-zero value when in normal control. If you have the RC3_DZ set to 0 and your transmitter doesn't output exactly the same value as RC3_TRIM you won't be able to arm using the steering as the APM will think you are at a non-zero throttle level.

Simplest Solution: Use Only ARMING_REQUIRE

The simplest way to use the throttle arming feature is to require the user to request the throttle to arm. ARMING_REQUIRE has three possible values, 0, 1 and 2 (the default value is 1, enabled). They have the following effect:

- ARMING_REQUIRE=0: No effect. Throttle arming safety is not employed.
- ARMING_REQUIRE=1: Before the user arms throttle, send trim PWM to the throttle channel (which is usually channel 3). Therefore the RC3_TRIM PWM value is sent on the servo output rail.
- ARMING_REQUIRE=2: Before the user arms throttle, send no PWM signal to the throttle channel (usually channel 3). Some ESCs are not happy with this and will continuously beep, some will not. Some users may prefer this setting as it ensures no signal is sent to the ESC when disarmed.

When the ARMING_REQUIRE parameter is set to 1 or 2 (enabled), all that is required to arm the throttle is to either:

1. Arm the throttle via the ground control software.

   OR

2. Arm the throttle by applying full right rudder input for several seconds.

To arm throttle in Mission Planner, use the Flight Data screen, then select the Actions tab. This provides an "Arm/Disarm" button that can be used to arm and disarm the throttle.

Location of the Arm/Disarm buttonin Mission Planner (button circled in red near the bottom of the image).

Alternatively, apply full right steering to your RC transmitter for several seconds to arm the throttle. Whichever method you choose should result in a message from your ground control software stating that throttle arming was successful. Mission planner displays a message in the artificial horizon on the HUD:

Message on the artificial horizon inMission Planner stating that arming was successful.

Increased Safety: Perform System Checks Before Arming Throttle

By default the ARMING_CHECK parameter is set to 1 so the autopilot controller performs system health checks before arming throttle when a user attempts to arm. You can enable/disable any check with a bitmask. See wiki documentation on the ARMING_CHECK parameter for more information.

One thing to be aware of if you typically do not drive with a ground control station: **it will be difficult to determine why your autopilot is not arming if you are not connected to a ground control station when arming.** The ARMING_CHECK parameter should probably be left at 0 when without a ground control station.

The following are possible system health messages that may return if ARMING_CHECK is enabled and the autopilot rejects a request to arm the throttle:

- Message: “Hardware Safety Switch.” Solution: push the hardware safety switch on the PX4 or Pixhawk (does not apply to APM).
- Message: “Battery failsafe on.” Solution: Ensure your battery is charged. If it is, ensure your battery failsafe values are set correctly. For more information on failsafes, see the Failsafe Functions wiki page.
- Message: “Radio failsafe on.” Solution: Ensure that the RC transmitter is able to communicate with the RC receiver. For more information on this failsafe, see the documentation on the Throttle Failsafe.
- Message: “Bad GPS Pos.” Solution: Need to get a 3D fix with the GPS receiver. After ensuring your GPS receiver is functioning properly, ensure nothing on the rover or in the immediate environment is interfering with GPS satellite signals.
- Message: “No GPS detected. Solution: ensure your GPS receiver is functioning.

- Message: “No compass detected.” A rover needs a compass. You’ll need to ensure your compass is installed and healthy.
- Message: “Compass not calibrated.” Solution: Calibrate compass. In Mission Planner this is accomplished in the Initial Setup screen, menu item Manadtory Hardware > Compass.
- Message: “Compass not healthy.” Solution: Ensure you do not have the compass installed near something that can induce a magnetic field, such as the motor. You also may try re-calibrating the compass.

Example Configuration

This topic provides additional recommended configuration for Rover vehicles during initial setup and testing. It assumes that the configuration is performed using *Mission Planner* (and that *Mission Planner* is already installed and connected).

1. Go to the **Config/Tuning** tab.
2. Select the **Planner** item in the left menu and ensure the **Advanced View** item on the bottom right is checked.
3. Click on the **Standard Params** item in the left menu and install some conservative starting parameters.
   - Set the **Channel 7** Option to **Learn Waypoint** if it is not already set to it.
   - Reduce **Target cruise speed in auto** from 5.00 to **2.50**.
   - Reduce **Base throttle percentage** from 50 to **30**.
4. Set a failsafe to turn your vehicle off (put it in HOLD mode) if you lose RC reception for 5 seconds.
   - Set **Failsafe Action** to **HOLD**.
   - Set **GCS failsafe enable** to **Disabled** unless you are using a 3DR Telemetry radio.
   - Set **Throttle Failsafe Enable** to **Enabled**
   - Failsafe is preset to 5 seconds but can be adjusted in the **Full Parameter List** with the **FS_TIMEOUT** parameter.
5. Select the **Advanced Params** item from the left menu and ensure the **Board Orientation** Item matches your setup.
   - If your autopilot is mounted upside down (e.g. for improved connector access), set the orientation to **Roll 180** (8).
   - If your autopilot is mounted right side up select **None**.
6. Scroll down to **Compass Orientation** and ensure it is set correctly for your compass. This should already have been set in Compass Calibration.
7. Scroll down to **RC trim PWM (RC3_TRIM)** and set it up for (reverse - centre - forward) throttle or not.
   - Throttle is normally channel 3, if you have changed it to another channel go there instead.
   - If you have a centre loaded throttle with reverse, set this value to **1500**. If not then leave it at

1100.

8. The Minimum Throttle parameter (`THR_MIN`) is mostly useful for rovers with internal combustion motors, to prevent the motor from cutting out in auto mode.
9. The Maximum Throttle parameter (`THR_MAX`) can be used to prevent overheating a ESC or motor on an electric rover.

Arming Rover

Arming your rover

Before you can driver your rover you need to arm it. Arming the rover before flight has two purposes:

- prevent the motor from turning when the user is not ready to driver (a safety feature)
- prevent movement before the autopilot is fully configured and ready to go

In past releases of APM:Rover arming was optional, and the requirement to arm (controlled by the ARMING_REQUIRED parameter) was disabled by default. This was changed for the 3.0.0 release to require arming by default.

The key thing that arming does is to enable the motor. You will not be able to start the motor (ie. control the throttle) until the rover is armed.

Note: If you have AHRS_EKF_USE enabled (you are using the EKF) then it is particularly important that you have arming enabled with arming checks enabled. Using EKF without arming checks may cause a crash.

Configuring Arming

There are two parameters which control how arming works:

- **ARMING_REQUIRE**: this controls whether an arming step is required. The default is 1, meaning that arming is required before moving. If set to 0 then arming is not required (the rover starts off armed).
- **ARMING_CHECK**: this controls what checks the autopilot does before arming is allowed. The default is 1, meaning all checks are done. Most users should leave it at 1, as the arming checks are important to ensure the autopilot is ready.

How to Arm

When you are ready to go you can ask Rover to arm. This can be done in two ways:

- **Steering Arming**. Hold the steering stick fully to the right and make sure the throttle stick is centred in the dead zone (DZ) for 2 seconds.
- **GCS Arming**. Press the arming button on your ground station

How to Disarm

This is whilst the throttle is in the DZ (normally centred with hands off) and steering to the left for 2 seconds. In ArduRover this condition could be accidentally triggered while driving so there are additional requirements prior to disarm:

- The ArduPilot controller needs to make sure that you are not actually driving. You need to be in a manual mode. This can be MANUAL, HOLD, LEARNING or STEERING. If your in AUTO, RTL or GUIDED you won't be able to disarm.

You can also disarm without using the transmitter with one of the following methods:

- use a ground station to issue a disarm command
- use the safety switch on your rover (on Pixhawk)

Visual and Audible signals

APM:Rover will provide visual and audio clues to the arming state if your autopilot has LEDs and a buzzer. The clues are:

- if the autopilot is disarmed, but is ready to arm then the large 3-colour LED will be flashing green
- if the autopilot is armed and ready to fly the large 3-colour LED is solid green
- when the autopilot is ready to arm it will play a "ready to arm" sound on the buzzer
- when the autopilot is armed or disarmed it will play the corresponding sound

See the sounds page to listen to what the buzzer sounds like for each state.

Arming Checks

Before allowing arming the autopilot checks a set of conditions. All conditions must pass for arming to be allowed. If any condition fails then a message explaining what failed is set to the GCS.

The checks performed are:

- Safety switch. The PX4 safety switch must be set to the safety-off state before arming is allowed. This is either done by pressing the safety switch for 2 seconds until it stops flashing, or you can disable the use of the safety switch by setting BRD_SAFETY_ENABLE=0
- Inertial Sensor Checks. The accelerometers and gyroscopes must all be healthy and all be calibrated. If you have more than one accel or gyro then they need to be consistent with each other.
- AHRS checks. The AHRS (attitude heading reference system) needs to be initialized and ready. Note that if you have the EKF enabled this may take up to 30 seconds after boot.
- Compass checks. All compasses must be configured and calibrated, and need to be consistent with each other (if you have more than one compass)
- GPS Checks. You need to have a 3D GPS fix.
- Battery checks. The battery voltage must be above the failsafe voltage (if configured)
- Logging checks. The logging subsystem needs to be working (ie. a microSD must be fitted and working on PX4)
- RC Control checks. You need to not be in RC failsafe

Throttle output when disarmed

When the rover is disarmed the throttle channel will not respond to input. There are two possible behaviours you can configure:

- ARMING_REQUIRE=1. When disarmed the trim value for the throttle channel (normally RC3_TRIM) will be sent to the throttle channel
- ARMING_REQUIRE=2. When disarmed no pulses are sent to the throttle channel. Note that some ESCs will beep to complain that they are powered on without a control signal

Diagnosing failure to arm

It can be frustrating if your rover refuses to arm. To diagnose arming issues follow this guide

Check it is ready to arm

If your board has a “ready to arm” LED (the large LED in the middle of the board on a Pixhawk) then that LED should be flashing green when the board is ready to arm. If it is flashing yellow then that indicates that one of the arming checks is not passing.

Try arming

Try sending an arm command with your GCS. If arming is refused then a message will be sent from the autopilot to the GCS indicating why it is refusing to arm.

Steering arming

If you are using right-steer + trim-throttle to arm and you don't get a message on your GCS giving a arming failure reason then it may be that your RC calibration is a bit off and the autopilot is not quite seeing trim throttle or isn't quite seeing full right steering.

Reasons for refusing to arm

When the autopilot refuses to arm it sends a STATUSTEXT MAVLink message to the GCS explaining why it is refusing. The possible reasons why the autopilot can refuse to arm are:

- **logging not available**. If your microSD card has failed or is corrupt then logging won't be available and you cannot arm.
- **gyros not healthy**. If the gyros have failed the autopilot will refuse to arm. This is rare, and if it happens repeatedly then you may have a hardware failure.
- **gyros not calibrated**. This happens when the automatic gyro calibration at startup didn't converge. Try rebooting the autopilot with the rover still.
- **accels not healthy**. If the accelerometers have failed the autopilot will refuse to arm. Try recalibrating your accelerometers.
- **GPS accuracy errors**. There are 4 types of GPS arming errors that can be reported. They are "GPS vert vel error", "GPS speed error", "GPS horiz error", "GPS numsats". Try moving your rover for better GPS reception or switching off any RF sources (such as a FPV transmitter) that may be interfering with your GPS.
- **Mag yaw error**. This happens when your compass is badly out of alignment. Check your compass orientation and re-do your compass calibration or move your rover further away from any magnetic materials.
- **EKF warmup**. This happens when the EKF is still warming up. Wait another 10 seconds and try again.
- **AHRS not healthy**. This means the EKF is not healthy. If the error persists then try rebooting your board.
- **3D accel cal needed**. This happens when you have not done a 3D accelerometer calibration.
- **Inconsistent accelerometers**. This happens when you have multiple IMUs (such as the Pixhawk which has two) and they are not consistent. This can be caused by temperature changes. If the error doesn't clear itself after a minute you will need to redo your accelerometer calibration.
- **Inconsistent gyros**. This happens when you have multiple gyros and they are not reporting consistent values. If the error does not clear itself after 30 seconds then you will need to reboot.
- **GPS n has not been fully configured**. This happens with a uBlox GPS where the GPS driver is unable to fully configure the GPS for the settings that are being requested. This can be caused by a bad wire between the autopilot and GPS, or by a bad response from the GPS. If the message is

about “GPS 0” then it is the first GPS. If it is “GPS 1” then it is the 2nd GPS. If you get a failure for the 2nd GPS and don’t have two GPS modules installed then set GPS_TYPE2 to zero to disable the 2nd GPS

Tuning steering and navigation for a Rover

When setting up a Rover there are 6 key parameters you need to get right. This guide provides a set of steps to get those parameters right, so that your rover will accurately navigate in AUTO mode.

The key parameters

- `STEER2SRV_P`: This tells the rover code what the turning circle (as a diameter in meters) is for your rover. It is critical that you get this parameter right, as it tells the code what steering angle to choose to achieve a desired turn rate.
- `TURN_MAX_G`: This tells the rover the maximum G force (in multiples of one gravity) that your rover can handle while remaining stable. If you set this too high then your rover may flip over in turns, or skid. If you set it too low then your rover will not be able to turn sharply enough.
- `NAVL1_PERIOD`: This controls the aggressiveness of the navigation algorithm. A smaller number means more aggressive turning in AUTO mode, a large number means larger, smoother turns.
- `SPEED_TURN_GAIN`: This controls how much the rover should slow down while turning, as a percentage of current target speed. To not slow down at all in turns set this to 100. When doing initial tuning it is recommended you use 100, and lower this later to improve handling of tight courses.
- `CRUISE_SPEED`: This controls the target speed in meters/second in AUTO.
- `CRUISE_THROTTLE`: This sets the initial guess at what throttle is needed to achieve CRUISE_SPEED when driving straight ahead. This needs to be right for the rover to achieve good speed control.

Step 1: Setting initial parameters

To start the tuning process set the following values:

- STEER2SRV_P = 2
- TURN_MAX_G = 1
- NAVL1_PERIOD = 6
- SPEED_TURN_GAIN = 100
- CRUISE_SPEED = 2
- CRUISE_THROTTLE = 20

These are conservative values that should give you slow, gentle behaviour for most rovers.

Step 2: Setting CRUISE_THROTTLE

We need to get `CRUISE_THROTTLE` right so we know what throttle level will give us a speed of 2 meters/second.

- Switch to MANUAL mode
- Slowly advance the throttle until the ground station shows a speed of 2 meters/second
- note the throttle percentage shown in the HUD
- set CRUISE_THROTTLE to the percentage needed for straight driving at 2 meters/second

Step 3: Setting the STEER2SRV_P

To set your STEER2SRV_P parameter you need to measure the diameter of the turning circle of your rover.

Put your rover into MANUAL mode, and put the steering hard over to one side. Then very slowly drive your rover in a circle. Use a tape measure to measure the diameter of that circle and set STEER2SRV_P to that value in meters.

Step 4: tuning TURN_MAX_G

The `TURN_MAX_G` parameter can now be tuned so that your rover can drive more aggressively, without turning over.

- put into STEERING mode
- turn at maximum turn rate and maximum throttle
- adjust TURN_MAX_G to highest level that rover remains stable, doesn't start to roll over and doesn't skid

Step 5: tuning NAVL1_PERIOD

Now you can finally start tuning your steering in AUTO mode. To tune in AUTO you will need to create a mission for your rover to navigate. A simple rectangular mission is the best to start with.

- create a rectangular course for the rover
- run the course in AUTO mode
- if the rover weaves along the straights, then raise NAVL1_PERIOD in increments of 0.5
- if the rover doesn't turn sharply enough then lower NAVL1_PERIOD in increments of 0.5
- if the rover is turning too late in corners then raise WP_RADIUS in increments of 1.0

You may also find that you may need to raise NAVL1_DAMPING slightly to improve navigation in tight courses.

Final Setup

Now that you have the basic parameters tuned you may find you can go back and raise CRUISE_SPEED for faster driving. If you do, then make sure you adjust CRUISE_THROTTLE to suit.

You may also like to lower SPEED_TURN_GAIN to ask the rover to slow down a bit when turning. That would allow for higher speeds when driving straight ahead.

There are a number of other parameters that can be used to fine tune your steering once you have completed the basic ones above. Please see the full parameter list for details.

Some parameters to pay particular attention to are STEER2SRV_TCONST, STEER2SRV_I, STEER2SRV_D and STEER2SRV_IMAX.

Fixing problems with weaving

A common issue with rovers is that the steering ‘weaves’, turning from side to side rather than turning smoothly. There can be several possible reasons for this happening.

The first thing you need to work out is if the problem is confined to low speed driving or also affects higher speed driving. If the problem only happens at very low speed (such as when first entering auto on a mission) then the most likely problem is that STEER2SRV_MINSPD is too low. The default is 1.0 m/s, which is quite low, and if your GPS heading isn’t very reliable at low speed then you may need to raise that number. Try 2.0 and see if that helps with low speed weaving.

If the problem also happens at higher speeds then it is likely to either be the L1 navigation tuning or the steering controller tuning. To fix it you will need to understand how these parameters work.

The NAVL1_PERIOD controls how rapidly the L1 navigation controller changes the demanded steering direction. Making this number larger will reduce weaving in AUTO, but it will also mean that you can’t steer as accurately around tight corners. Try raising NAVL1_PERIOD in steps of 0.5 until the weaving stops. For most rovers a value between 6.0 and 8.0 is good, but some rovers may need higher values.

If raising the NAVL1_PERIOD fixes the weaving but leaves you unable to handle sharp turns in missions then you will instead need to tune the steering controller.

In the steering controller there are 3 key parameters that will control weaving:

- A smaller STEER2SRV_P will reduce weaving, try reducing it by 0.1 at a time
- A larger STEER2SRV_D will “damp” the weaving, but if you make it too large then you will get high speed oscillation. For example you may find that a value of 0.1 reduces the damping, but a value of 0.2 could cause a high speed oscillation in the steering servo. If you get fast oscillation then reduce the STEER2SRV_D value by 50%.
- A larger STEER2SRV_TCONST will slow down the steering controller, which will reduce weaving. Try raising it in steps of 0.1.

With the current controller system you do need to experiment a bit with these values to get the behaviour you want. We hope to introduce an automatic tuning system in future, but for now manual tuning is needed.

Parameter Tuning

This section contains topics about tuning Rover parameters:

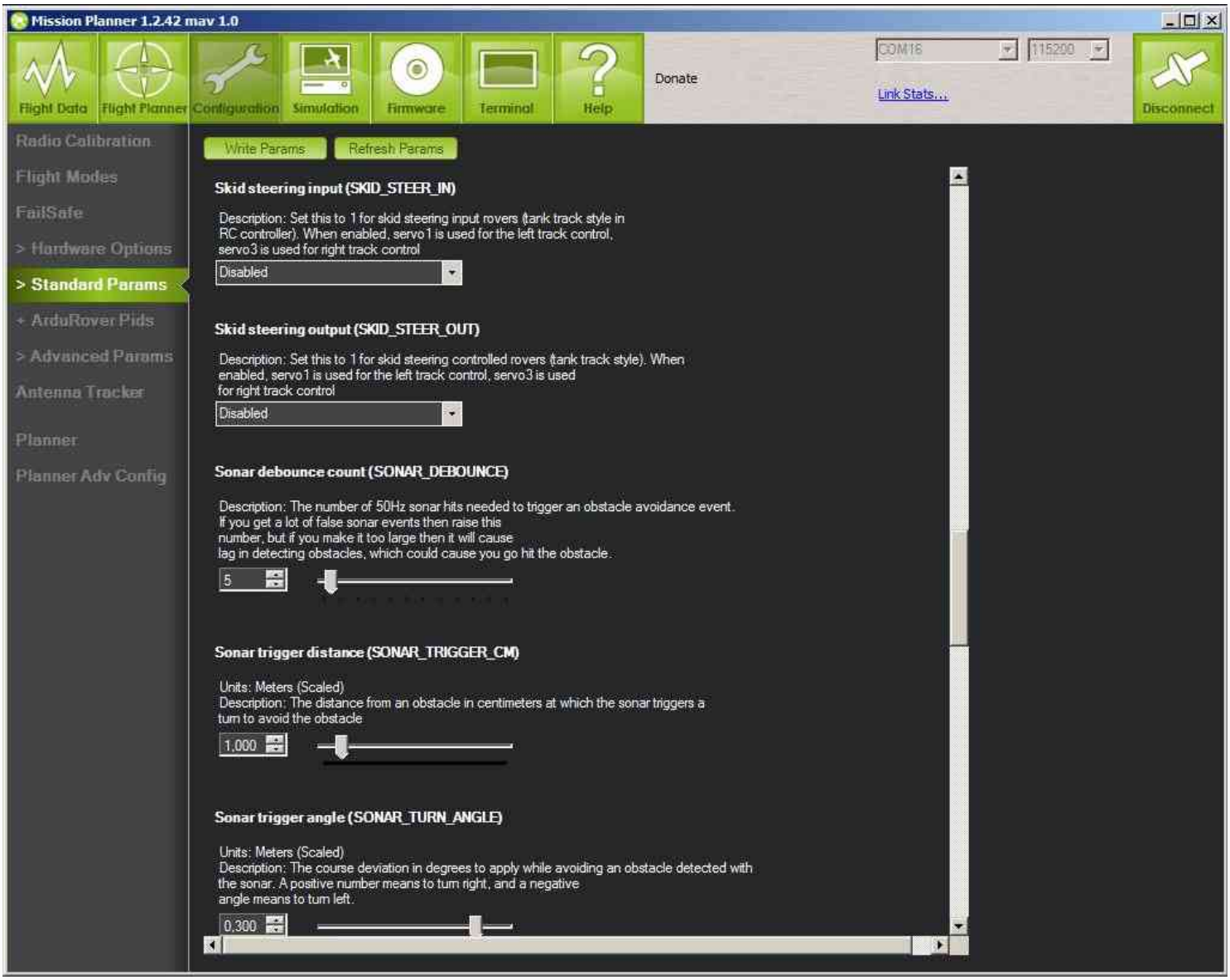
Skid Steer Parameter Tuning

The skid-steer parameters are used to configure whether a Skid Steer vehicle uses a dual throttle skid steer transmitter or a conventional single throttle RC transmitter.

There are two parameters, only one of which should be enabled (depending on the type of transmitter used).

SKID_STEER_OUT	0=Disabled 1=SkidSteer	Set this to 1 for skid steering controlled rovers (tank track style). When enabled, servo1 is used for the left track control, servo3 is used for right track control
SKID_STEER_IN	0=Disabled 1=SkidSteerRC	Set this to 1 for skid steering input rovers (tank track style RC Transmitter). When enabled, servo1 is used for the left track control, servo3 is used for right track control

The options are set in the *Mission Planner* **CONFIG/TUNING | Standard Params** page:



Mission Planner: Rover Skid-SteerParameters

Note

All parameters may be adjusted in the “Advanced Parameter List” in Mission Planner). All the Rover user settable parameters are listed in APMrover2 Parameters.

Sonar Parameter Tuning

This topic explains how to tune the SONAR/RANGEFINDER parameters.

Overview

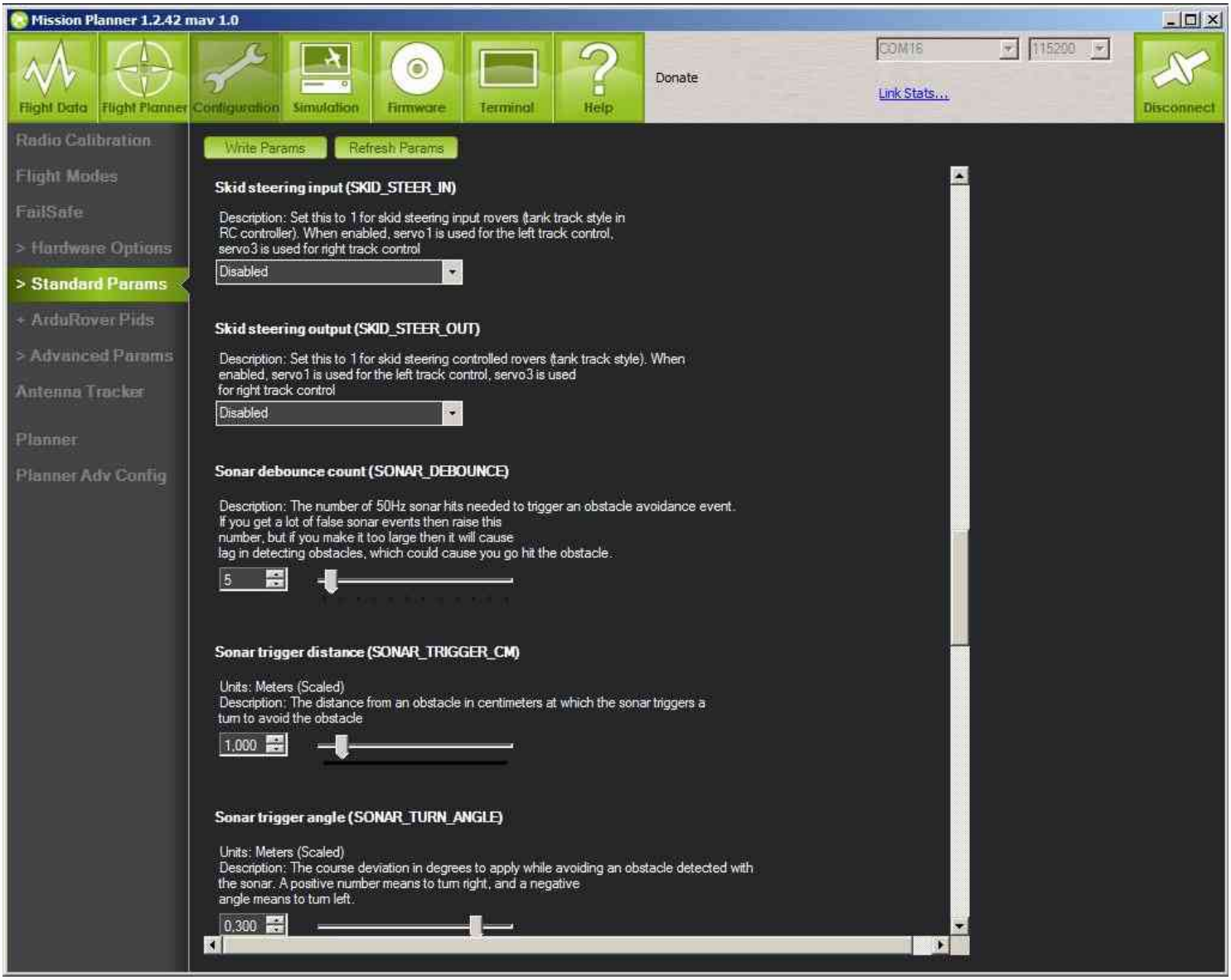
- Up to 4 RNGFNDs may be used.
- Debounce lets you specify the number of returns before triggering an obstacle avoidance event,

good for filtering false returns.
- Maximum and Minimum range of the RNGFND can be set.
- The SONAR readings can be scaled and adjusted for your use.
- When an obstacle is detected you can set the correcting steering angle and the time to hold it.

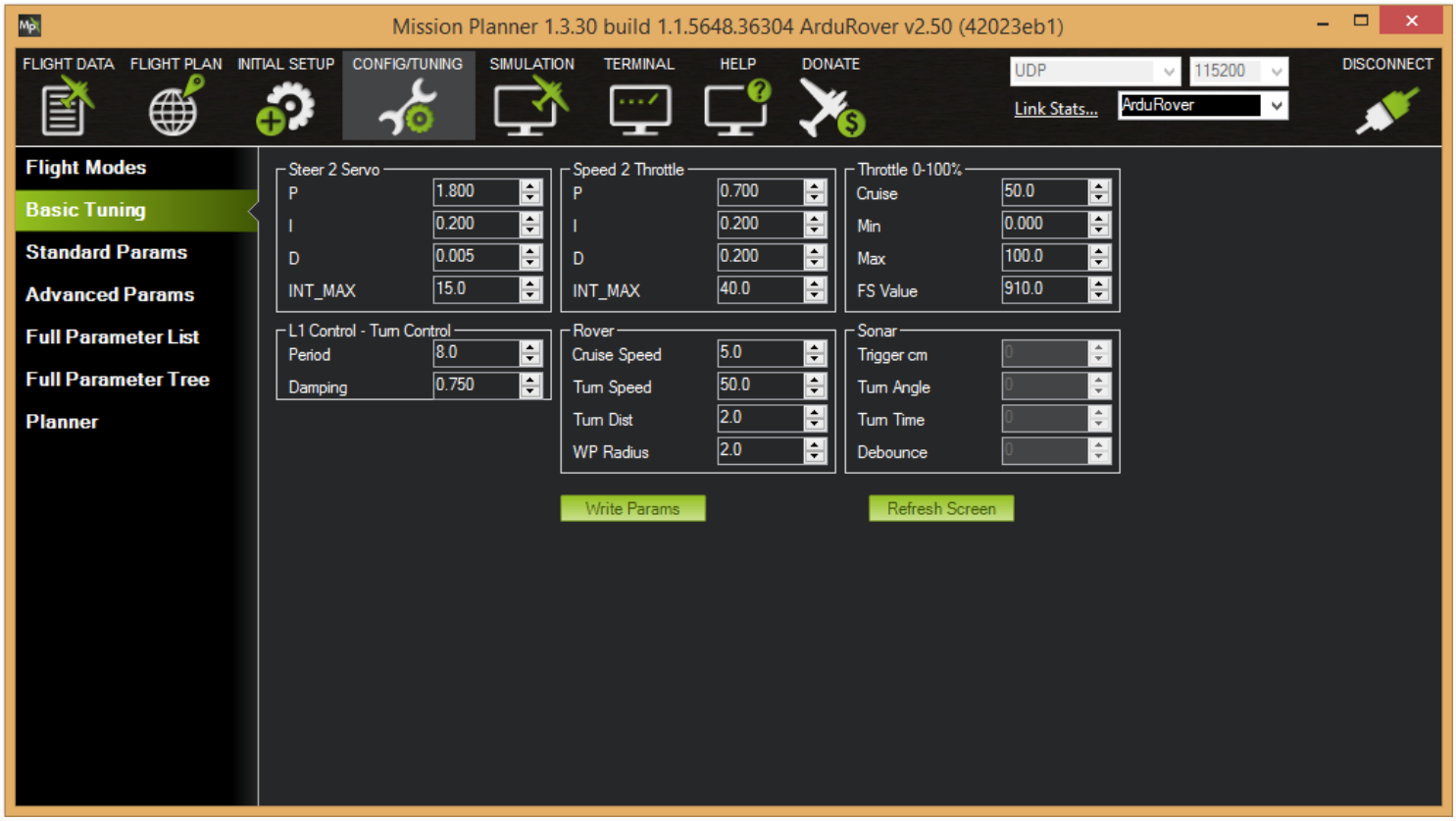
RNGFND parameters

The parameters that are relevant to the rangefinder(s) are listed here.

The parameters are set in the *Mission Planner* **CONFIG/TUNING| Basic Tuning** or **CONFIG/TUNING | Standard Params** pages):



Mission Planner: Rover SonarParameters



Mission Planner: Rover BasicTuning

Note

All parameters may be adjusted in the “Advanced Parameter List” in Mission Planner. All the Rover user settable parameters are listed in APMrover2 Parameters.

Speed Turn Parameter Tuning

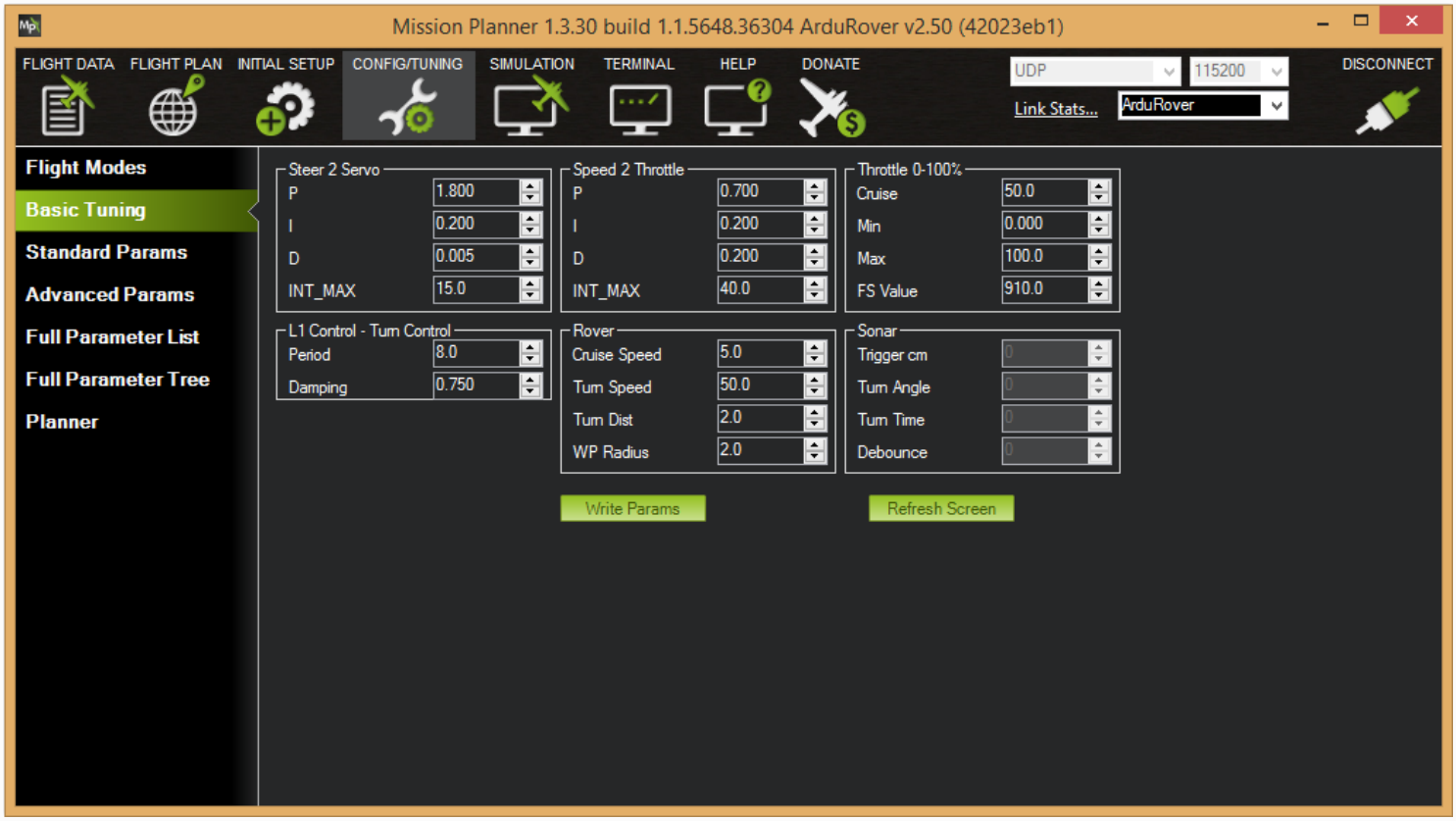
The speed-scaling parameters are used to set the percentage reduction in throttle when turning around a waypoint and the distance before a target when the Rover will start to slow down.

The two parameters are listed below:

- rover:SPEED_TURN_GAIN
  - Range 0-100 Units: percent Increment 1 (50)
  - The percentage to reduce the throttle while turning. If this is 100% then the target speed is not reduced while turning. If this is 50% then the target speed is reduced in proportion to the turn rate, with a reduction of 50% when the steering is maximally deflected.
- rover:SPEED_TURN_DIST
  - Range 0-100 Units: meters Increment .1 (50)
  - The distance to the next turn at which the rover reduces its target speed by the

SPEED_TURN_GAIN. |

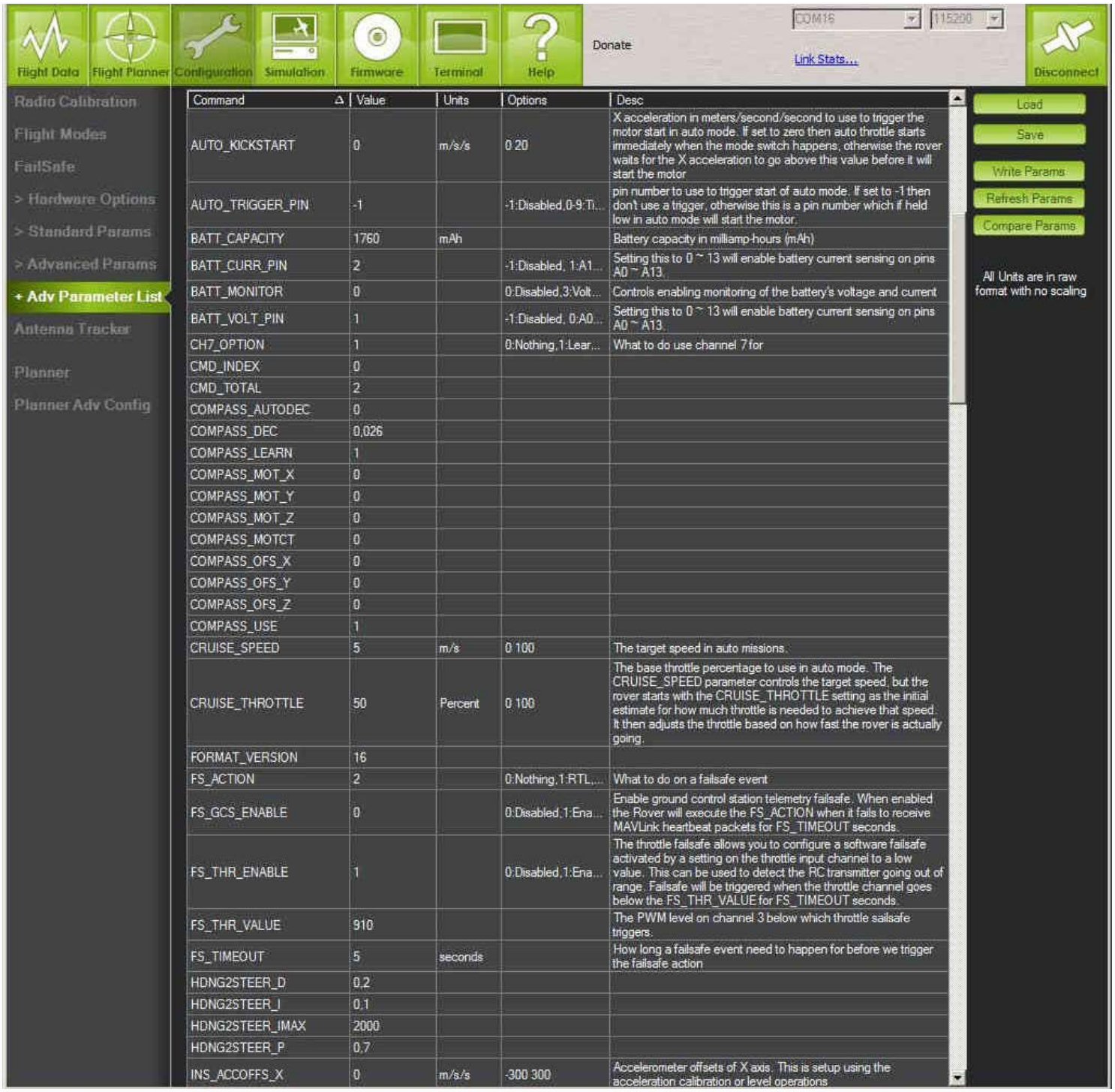
The parameters are set in the *Mission Planner* **CONFIG/TUNING | Basic Tuning** page:



Mission Planner: Rover BasicTuning

Note

All parameters may be adjusted in the “Advanced Parameter List” in Mission Planner). All the Rover user settable parameters are listed in APMrover2 Parameters.
All Rover parameters are listed here. The parameters can be adjusted in *Mission Planner* in the specific screens described in each topic or in the **Advanced Parameter List**.



Command	Value	Units	Options	Desc
AUTO_KICKSTART	0	m/s/s	0 20	X acceleration in meters/second/second to use to trigger the motor start in auto mode. If set to zero then auto throttle starts immediately when the mode switch happens, otherwise the rover waits for the X acceleration to go above this value before it will start the motor
AUTO_TRIGGER_PIN	-1		-1:Disabled,0-9:Ti...	pin number to use to trigger start of auto mode. If set to -1 then don't use a trigger, otherwise this is a pin number which if held low in auto mode will start the motor.
BATT_CAPACITY	1760	mAh		Battery capacity in milliamp-hours (mAh)
BATT_CURR_PIN	2		-1:Disabled, 1:A1...	Setting this to 0 ~ 13 will enable battery current sensing on pins A0 ~ A13
BATT_MONITOR	0		0:Disabled,3:Volt...	Controls enabling monitoring of the battery's voltage and current
BATT_VOLT_PIN	1		-1:Disabled, 0:A0...	Setting this to 0 ~ 13 will enable battery current sensing on pins A0 ~ A13
CH7_OPTION	1		0:Nothing,1:Lear...	What to do use channel 7 for
CMD_INDEX	0			
CMD_TOTAL	2			
COMPASS_AUTODEC	0			
COMPASS_DEC	0,026			
COMPASS_LEARN	1			
COMPASS_MOT_X	0			
COMPASS_MOT_Y	0			
COMPASS_MOT_Z	0			
COMPASS_MOTCT	0			
COMPASS_OFS_X	0			
COMPASS_OFS_Y	0			
COMPASS_OFS_Z	0			
COMPASS_USE	1			
CRUISE_SPEED	5	m/s	0 100	The target speed in auto missions.
CRUISE_THROTTLE	50	Percent	0 100	The base throttle percentage to use in auto mode. The CRUISE_SPEED parameter controls the target speed, but the rover starts with the CRUISE_THROTTLE setting as the initial estimate for how much throttle is needed to achieve that speed. It then adjusts the throttle based on how fast the rover is actually going.
FORMAT_VERSION	16			
FS_ACTION	2		0:Nothing,1:RTL,...	What to do on a failsafe event
FS_GCS_ENABLE	0		0:Disabled,1:Ena...	Enable ground control station telemetry failsafe. When enabled the Rover will execute the FS_ACTION when it fails to receive MAVLink heartbeat packets for FS_TIMEOUT seconds.
FS_THR_ENABLE	1		0:Disabled,1:Ena...	The throttle failsafe allows you to configure a software failsafe activated by a setting on the throttle input channel to a low value. This can be used to detect the RC transmitter going out of range. Failsafe will be triggered when the throttle channel goes below the FS_THR_VALUE for FS_TIMEOUT seconds.
FS_THR_VALUE	910			The PWM level on channel 3 below which throttle sailsafe triggers.
FS_TIMEOUT	5	seconds		How long a failsafe event need to happen for before we trigger the failsafe action
HDNG2STEER_D	0,2			
HDNG2STEER_I	0,1			
HDNG2STEER_IMAX	2000			
HDNG2STEER_P	0,7			
INS_ACCOFFS_X	0	m/s/s	-300 300	Accelerometer offsets of X axis. This is setup using the acceleration calibration or level operations

Mission Planner: Rover AdvancedParameters List