AutoQuad6 & ESC32 CookBook

kinderkram, menno

Creative Commons - BY -- 2014

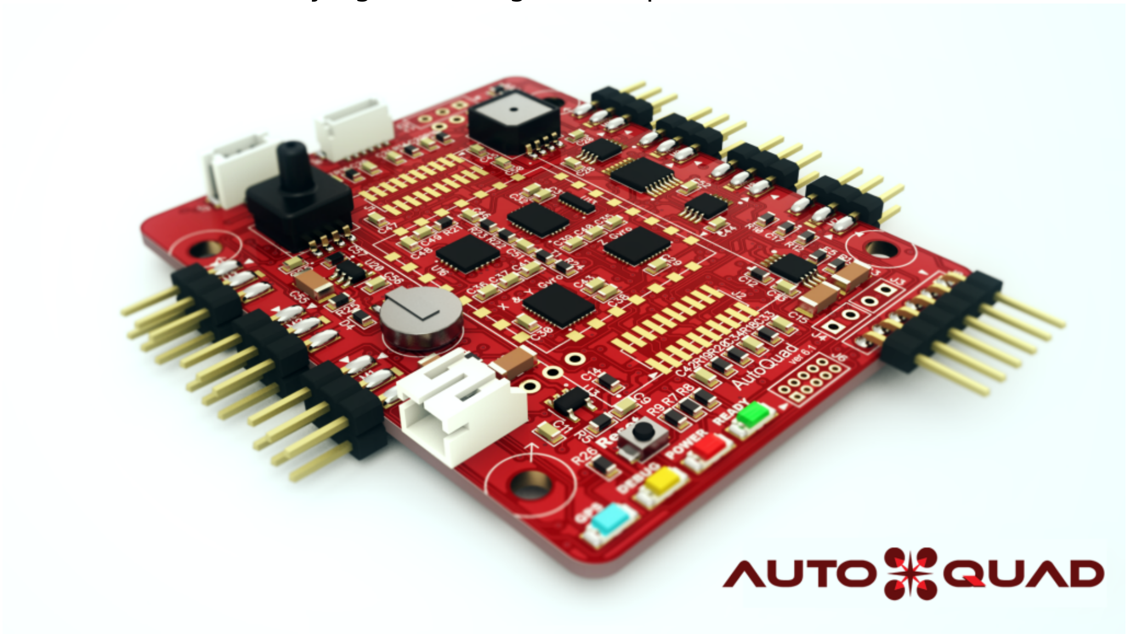
Table of Contents

Part 1 - Hardware & Software
AutoQuad 6

Introducing AutoQuad

What makes the AutoQuad flight controller special?

Besides using top quality sensors and MCU, AutoQuad differs in its approach of using sensor calibration techniques and calculating the magnetic profile of the entire multirotor creating the ultimate stable flying and navigation experience.



Summary of specifications of the AQ 6.x flightcontroller hardware

- 2″ x 2.5″ main board with 4,5×4,5cm mounting hole pattern
- Input voltage: 6.5v => 18v
- 2 High efficiency DC/DC converters, separate power to flightcontroller logics and accessories.
- STM32F407 32bit Cortex M4 microcontroller (1MB flash)
- Standard Arm 10 pin 0.05″ pitch SWD connector
- 14 general purpose PWM controllers / receivers (powered or un-powered)
- Dedicated Spektrum satellite (remote receiver) 2.4Ghz R/C radio connector
- uSD card slot driven by 4bit SDIO capable of 100Mb/s transfer (up to 32GB storage)
- onboard uBlox LEA-6T GPS module with battery backup and timepulse capture
- u.FL active GPS antenna connector
- optional external bi-directional telemetry radio via standard 6 pin FTDI connector – powered up to 1A
- I2C bus connector for I2C ESC’s (or other I2C devices)
- X,Y Mag: HMC6042
- Z Mag: HMC1041Z

- X,Y Gyro: IDG500
- Z Gyro: ISZ500
- Accel: ADXL325
- Pressure Sensor: MP3H6115A (optional 2. tube sensor)
- Battery voltage monitor

onboard IMU options:

- 9 DOF analog sensors (3x RATE, 3x ACC, 3x MAG) w/EMI hardening
- Optional DIMU (digital IMU w/ CAN Bus, new pressure sensor)
- Optional CAN bob as daughter board extension
- Optional VectorNav VN100

Summary of capabilities of the AQ 6.x firmware

- Fly extremely stable yet offer full dynamic control to the pilot.
- Limit flying angles.
- Mavlink 1.0x compatible protocol, Ground Stations for Win, Mac, Linux & Android
- Very accurate position hold, depending on GPS reception a hold within 15-30 cm is possible.
- Altitude override (ascend / descend) during position hold with controlled vertical speed.
- Full mission flight, speed, heading and ‘loiter time’ is settable.
- Return to home position, altitude is also recorded during home position set.
- POI (point of interest), autonomous circling around given coordinates
- Heading Free flight mode, Follow Me, Click & Go (needs exp. firmware)
- Radio loss detection and event triggering.
- Low battery detection and event triggering.
- 2 axis gimbal control with pitch angle override on transmitter.
- Gimbal working angle and response time is settable.
- Gimbal servos can use 50Hz (analog) to 400+ Hz (digital) settable.
- **Radio options:** Spektrum satellite (10 & 11bit), S-bus receiver, (C)PPM input, SUMD (Graupner) M-Link (Multiplex)
- **User Addon** Waypoint recording and playback using transmitter switch.
- **User Addon** External LED / Piezo signaling for status and events, MAVLink telemetry for Graupner HOTT radios

Learn more about the AutoQuad in our Wiki

AutoQuad firmware and most of the related software is published under the GNU GPLv3 software license. Please read the license carefully before you apply changes to the code.
More info on the AutoQuad licensing.

3D animation: use your mouse to turn the board 360deg. left & right

Getting Started

Getting Started

Welcome to the AutoQuad documentation project! You can use these pages to learn about and set up the system, progress to your first manual flight, and finally the first autonomous flight. This Getting Started section covers all the components you will need as well as the first assembly and configuration steps.

Components needed (the short version)

- **AutoQuad flight controller** available at flyduino.com and viacopter.eu
- **ESCs** flashed for 400Hz output or the ESC32.
- **FTDI / USB-Serial adapter** for uploading firmware and wired communication.
- **Frame:** tri – quad – hexa – octo or any other exotic is possible. Y3 is not supported at this time.
- **RC Gear**: A minimum of 6 channels is needed — see radio options.
- **A class 10 micro SD card** for calibration and on-board logging. See the recommended list.
- And last but not least: **Lots of time**.

There is NO reverse polarity protection on the AutoQuad
Reversing the voltage polarity will DESTROY YOUR BOARD INSTANTLY!

Take the following steps to get to the first flight.

1. Get all the additional components needed.
2. Assemble the headers on the AutoQuad.
3. Download firmware and AutoQuad software.
4. Install the Ground Control Station.
5. Flash firmware to the AutoQuad & basic setup.
6. Perform the static calibration.
7. Place the AutoQuad on your new frame.
8. Perform the dynamic calibration.
9. Perform the calculations.
10. Upload the calculations and perform final setup to the AutoQuad.
11. Run preflight tests and continue to your first flight.

This is not a plug-and-play project! You have to invest in precise calibration, understanding the science behind the AutoQuad, and give us (the team) time to enhance the tooling and even further improve firmware. The project is far from finished, but the team is flying a growing number of AutoQuads with success. If you invest that time, you will be rewarded with probably the best multicopter experience that exists at this moment, without braking the bank.

For example: AutoQuad 6.4 position hold in gusting wind:

Where to buy

Where to Buy AutoQuad FC and ESC32 Hardware

AutoQuad is not a commercial venture but an Open Source community project with closed source hardware.

The AutoQuad flight controller and AutoQuad ESC32 will be available for purchase at the following shops:

FLYDUINO MULTICOPTERSHOP	Germany
ViaCOPTER www.viacopter.eu	Denmark

What Else You Will Need

What Else You Will Need

FTDI / USB2Serial

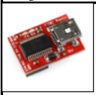
The AutoQuad flight controller does not contain a USB interface. For uploading the firmware an on-board serial interface is available that can be connected to a number of FDTI or USB2Serial adapters.

We recommend the combined AutoQuad/MultiWii USB UART adapter:

You can order one at Flyduino or Viacopter

The AutoQuad MCU (STM32F407 cortex M4) ports must not be provided with voltages above 3.3v. Although some ports are 5V tolerant, best is to use an FTDI or USB2Serial where the signals are on a 3.3V level.

Below is a list of known FTDI / USB2Serial adapters that are used by the development team.
You only need one of them – we recommend the cable.
Settings for the connection: 115.200, N81 and “flow control” activated
For flashing and setup you only need GND, Rx and Tx connected – flow control disabled.
Please have a look at our Wiki.

Flyduino FTDI (recommended or included)


Some FTDI adapters will keep the DTR signal low and prevent the AutoQuad from communicating with the ground control station. In that case disconnect the DTR pin. Please read the instructions in our Wiki

Micro SD card

The AutoQuad flight controller contains a microSD card slot.
The microSD card is vital for the AutoQuad, it is needed during the static and dynamic calibration sequence, it can store the calculated parameters (and thus a specific frame configuration), it can datalog all variables during flight and a future enhancement could be creating and storing waypoint information.

AutoQuad is very critical with the SD card performance, the firmware will offload every 10 seconds and based on the data an class 4 or 6 should be sufficient.
That is however **not** the case, the AQ team has tested a large number of class 4, 6 and 10 cards and only very specific class 10 cards are working.

Errors to expect on lower end class 10 cards;

- not initializing at all
- SDIO errors during logging in the telemetry data console
- sequence errors during calibration logging.

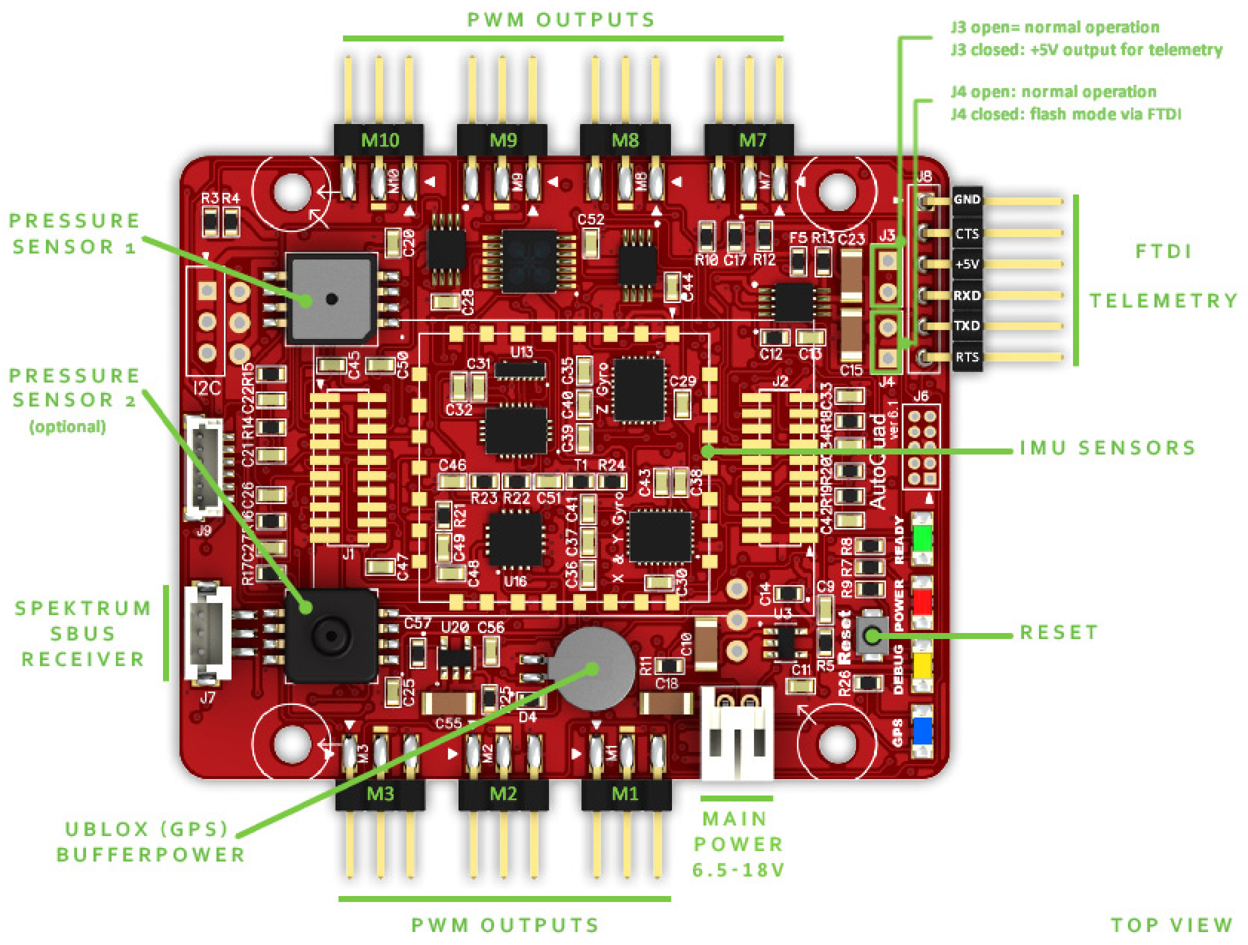
The latter is very important, a sequence error should **not** be in the calibration file, it will disturb the final calculations.

A (very) short list of known (successfully tested) **class 10 microSD** cards. Some users reported problems with cards above 8GB capacity. So atm. we recommend to only use microSD cards with a size **up to 8GB!**

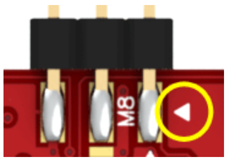
Name/Brand	Class	Information

Flight Controller Hardware

AutoQuad Flight Controller Hardware



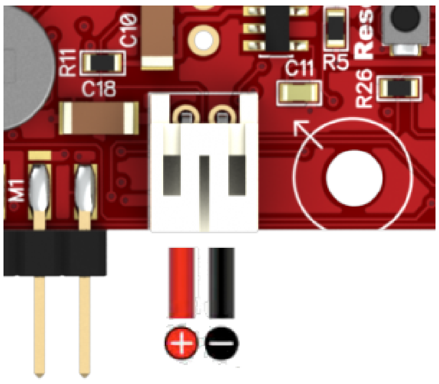
The AutoQuad flight controller will arrive without headers soldered. For the first static calibration headers are not necessary but it might be the moment to warm up your soldering iron and attach the headers to the AutoQuad flight controller. At a minimum, you will need the “PWM Output” connector headers, the “FTDI/Telemetry” header, and 2 headers for the jumpers J3 and J4 (they are next to the FTDI connector, click on above image for a better view). A short hardware assembly guide is available. But first please read through this section first! **Note:** Please be very careful when soldering the components! Please also do a short visual inspection of the hand-soldered parts Make sure the RECOMs (the big black boxes on the underside of the board) are fixated and soldered properly. Read the forum here .



The white arrow on the output pins marks the signal pin for the ESC/servo output or PPM input.

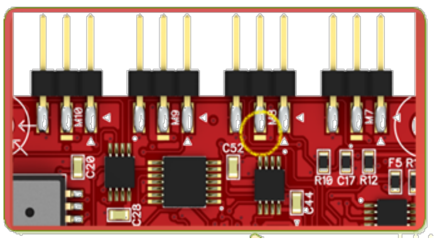
Power connector

This is the ONLY correct way to power the AQ. **Input voltage: 6.5v to 18v**



There is NO reverse polarity protection on the AutoQuad **Reversing the voltage polarity will DESTROY YOUR BOARD INSTANTLY!** If you are powering the FC on the bench through a power supply, make sure both have a common ground! We recommend using a lipo instead.ESC/Servo output and PPM input connectors.

The "M" connectors on the board can serve multiple purposes. Currently they can be configured as an ESC/Motor output, a servo output, or as PPM (RC signal) input. The middle pin of the "M" connectors is reserved for the typical 5V output. A servo requires that the middle pin is powered, as does an RC receiver, but BEC-based ESCs **do not!!** To prevent any conflicting power supplies across the board, the AutoQuad has a neat option: a solder bridge under every motor middle pin (see image below). If you solder the bridge across the pin, you will supply 5V to the middle pin. But be careful, if you attach a BEC based ESC, you introduce 2 power supplies that will "fight" each other, the BEC from the ESC and the 5V from the AutoQuad. Some general guidelines:



- Servos : solder the **5v contact point** across the pin — but only if you will be powering the servos directly from the AQ voltage regulators — **that means low-draw analog servos only** !

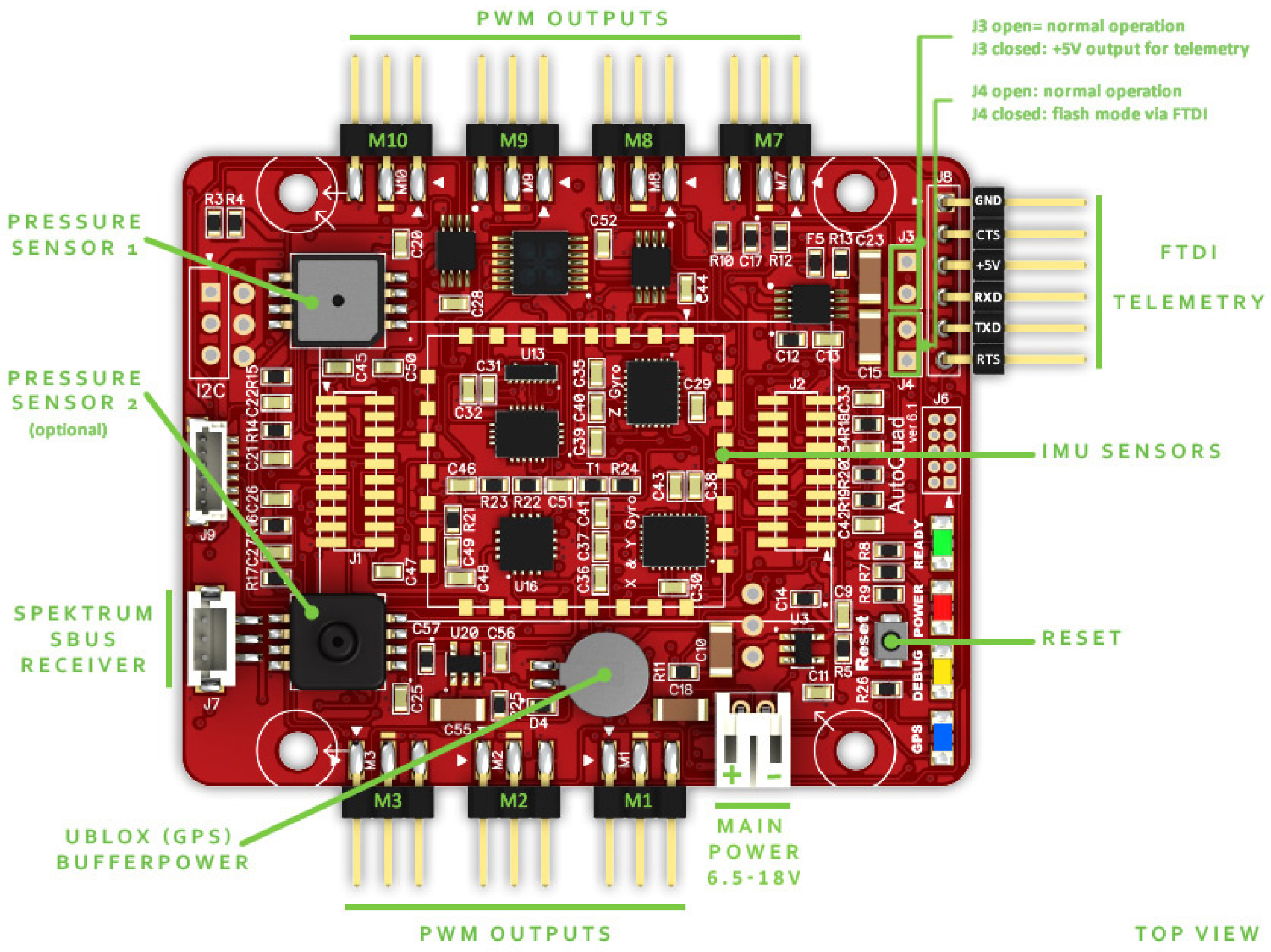
- PPM input on M14 from Rx: solder the 5v contact to conveniently provide the Rx with 5v power.
- Regular ESCs with BEC : do NOT solder the contact point.
- ESCs without BEC : optional — could be that power to the ESC MCU is needed, or not.

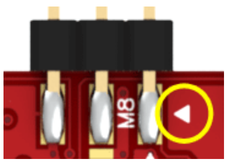
Other connectors and more information

For more hardware and connection details, LED indicator meanings, and a description of the external headers and their MCU connections you can check the AQ Hardware & Connections section. You can download a description and further details of Headers & Connectors as **PDF**.

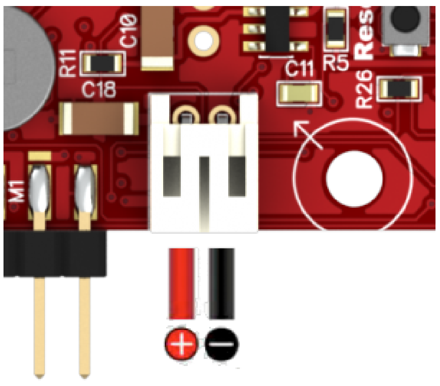
AutoQuad Hardware & Connections

Hardware and Connections



The white arrow on the output pins marks the signal pin for the ESC/servo output or PPM input.Always power your board externally when using the FTDI/USB connector.

Power connector

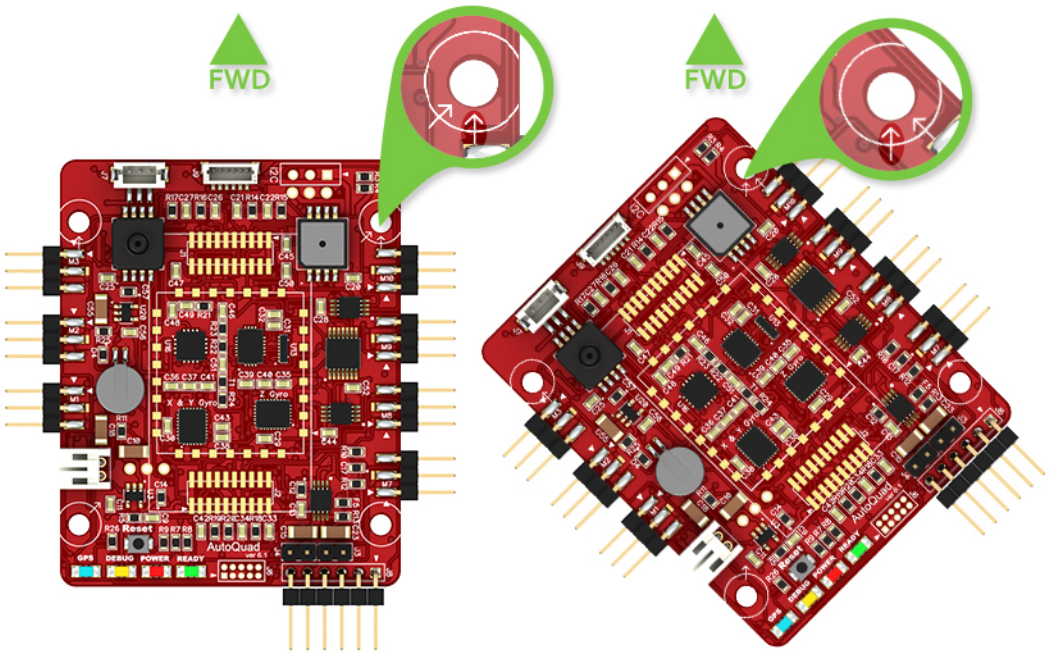


This is the ONLY correct way to power the AQ.

Input voltage: 6.5v to 18v

There is NO reverse polarity protection on the AutoQuad
Reversing the voltage polarity will DESTROY YOUR BOARD INSTANTLY!

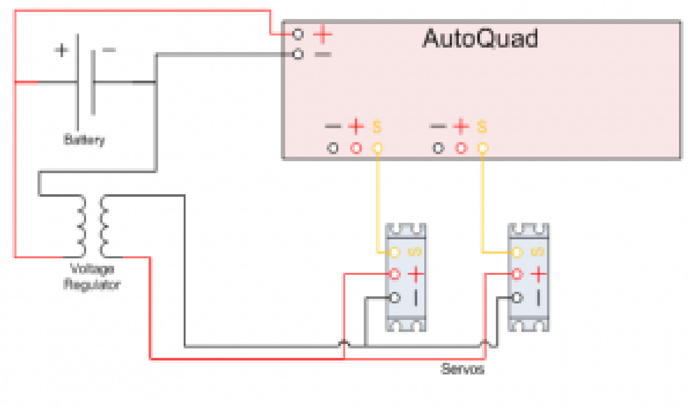
Flight Controller Orientation



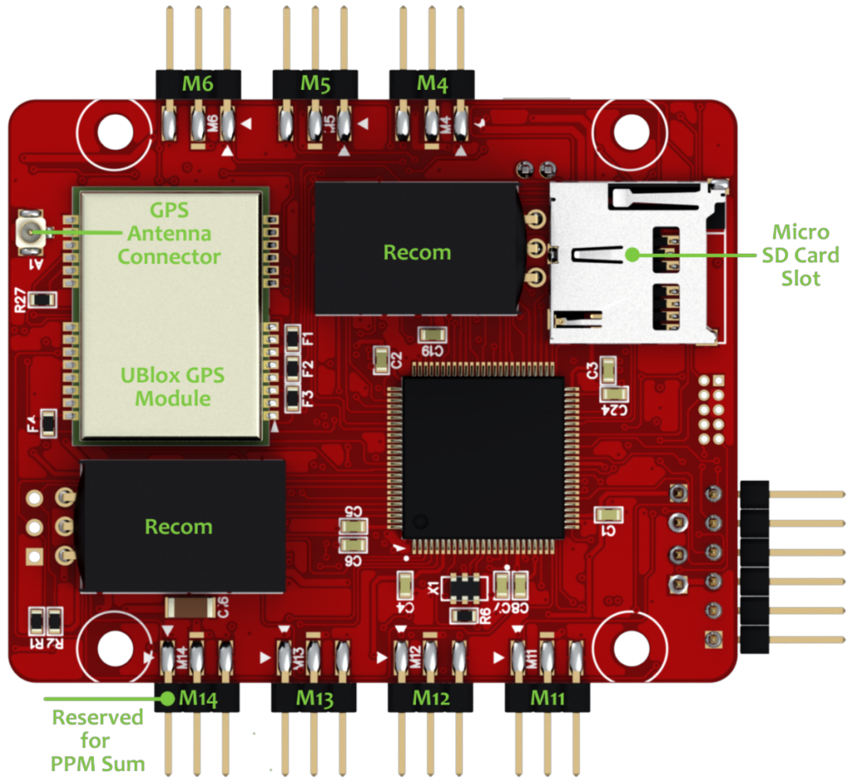
If mounting the board with the upper right corner facing forward (board is rotated counter clockwise), be sure to set the IMU Rotation parameter to “-45″ in the ground control software (Edit Parameter -> Special Settings -> Diverse). If the upper left corner is facing forward use “+45″ (clockwise rotation).

Connecting gimbal servos

The AutoQuad board can safely supply up to ~1.5 amps to your gimbal servo(s). This is adequate for small analog servos, but is **NOT recommended for larger (standard size) analog servos or any digital servos**. Since any servo may draw high amps when stalled, especially digital ones, it is recommended to power your servos with an external voltage regulator (UBEC). Here is a diagram of how to hook up servos using a separate power regulator (click for full-size version).

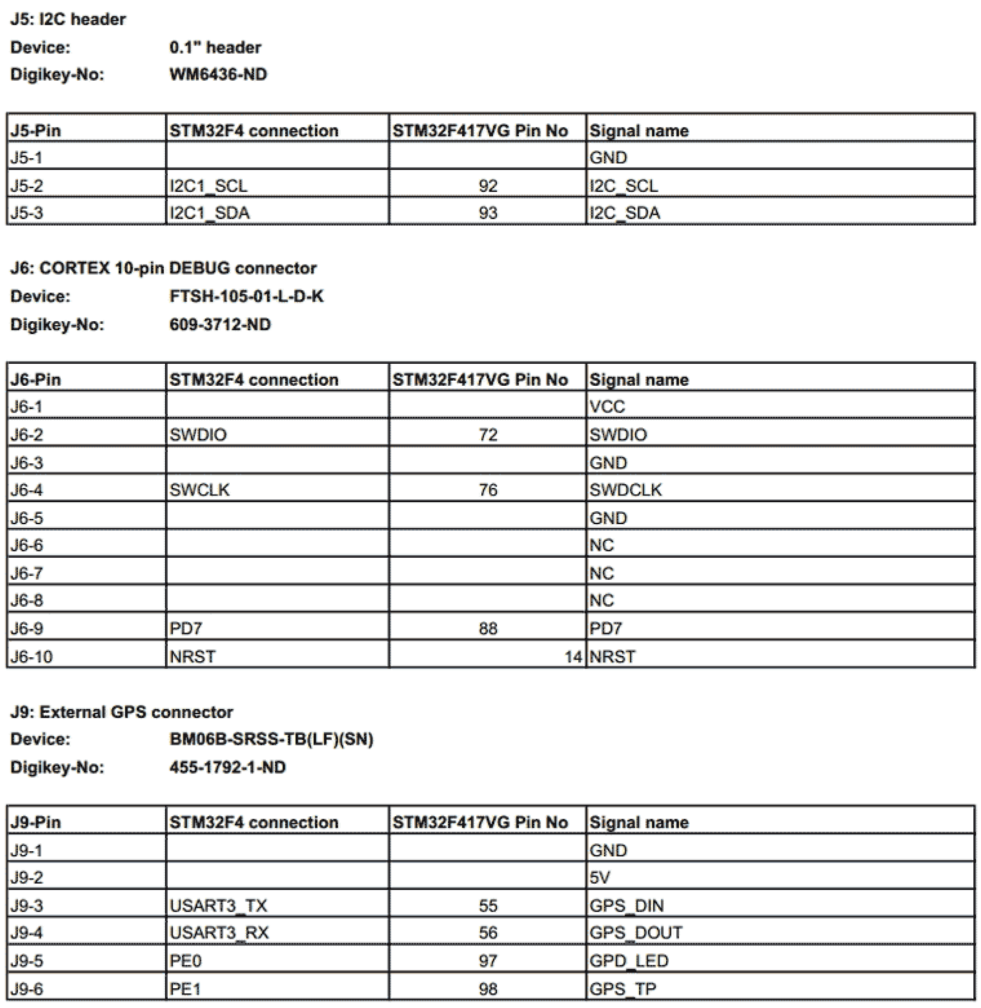


Underside components



The back with 2 DC-DC converters, Ublox GPS module, uSD card slot, and the heart of the system: STM32F407 MCU.

AutoQuad6 headers & pin descriptions



J5: I2C header

Device:	**0.1" header**
Digikey-No:	**WM6436-ND**

J5-Pin	STM32F4 connection	STM32F417VG Pin No	Signal name
J5-1			GND
J5-2	I2C1_SCL	92	I2C_SCL
J5-3	I2C1_SDA	93	I2C_SDA

J6: CORTEX 10-pin DEBUG connector

Device:	**FTSH-105-01-L-D-K**
Digikey-No:	**609-3712-ND**

J6-Pin	STM32F4 connection	STM32F417VG Pin No	Signal name
J6-1			VCC
J6-2	SWDIO	72	SWDIO
J6-3			GND
J6-4	SWCLK	76	SWDCLK
J6-5			GND
J6-6			NC
J6-7			NC
J6-8			NC
J6-9	PD7	88	PD7
J6-10	NRST	14	NRST

J9: External GPS connector

Device:	**BM06B-SRSS-TB(LF)(SN)**
Digikey-No:	**455-1792-1-ND**

J9-Pin	STM32F4 connection	STM32F417VG Pin No	Signal name
J9-1			GND
J9-2			5V
J9-3	USART3_TX	55	GPS_DIN
J9-4	USART3_RX	56	GPS_DOUT
J9-5	PE0	97	GPD_LED
J9-6	PE1	98	GPS_TP

You can also download this description and further details as a PDF.

Further reading in this section

- Flight Controller Status LEDs
- Radio connections
    - PPM notes
- GPS & Groundplane
- Barometers
- Telemetry – Xbee – Bluetooth – 3DR
- Voltage warnings and wiring advice
- DIMU upgrade

AutoQuad Software Setup

AutoQuad Software Setup

AutoQuad flight controller will arrive with no firmware or flashed with test firmware. To configure, upgrade, or flash the firmware you will need software. Most flight controllers will provide a Ground Control Station (GCS), and AutoQuad does as well. With the AutoQuad GCS you can flash firmware, configure settings, view real-time telemetry data or SD card logs, and plan missions. The AutoQuad GSC will also be used for the ESC32 setup.

AutoQuad firmware and most of the related software is published under the GNU GPLv3 software license. Please read the license carefully before you apply changes to the code.
Read more info on the AutoQuad licensing.

The GCS is available for Windows, Linux, and Android-based devices.

This section is divided into the following pages:

- Ground Control Station for PC (Win/Mac/Linux)
- Ground Control Station for Android
- AutoQuad Firmware

Ground Control Station for PC (Win/Mac/Linux)

Ground Control Station (GCS) for Windows and Linux

AutoQuad uses a custom version the excellent QGroundControl (QGC) GCS based on the MAVLink 1.0 protocol, both developed and maintained by the PIXHAWK MAV Student Team and Research Project at ETH Zurich , Switzerland.

You can download a specially tailored QGroundControl station for AutoQuad in our download section . The links below are provided as a reference to the base versions of QGC and ML — these will not work with AQ as-is.

QGROUNDCONTROL
GROUND CONTROL STATION FOR SMALL AIR - LAND - WATER AUTONOMOUS UNMANNED SYSTEMS Image not found

Download the QGC software and unzip it to a folder anywhere in your file system or desktop. QGC requires no installation process. You can run QGC by executing qgroundcontrol.exe (Windows) or qgroundcontrol (Linux).

It will greet you in the main screen with a number of default tabs. You can configure all tabs as you like — read the QGC documentation for all options. When you use the AutoQuad version of QGC, the AutoQuad main “widget” is available in the *‘Engineer* ‘ tab.

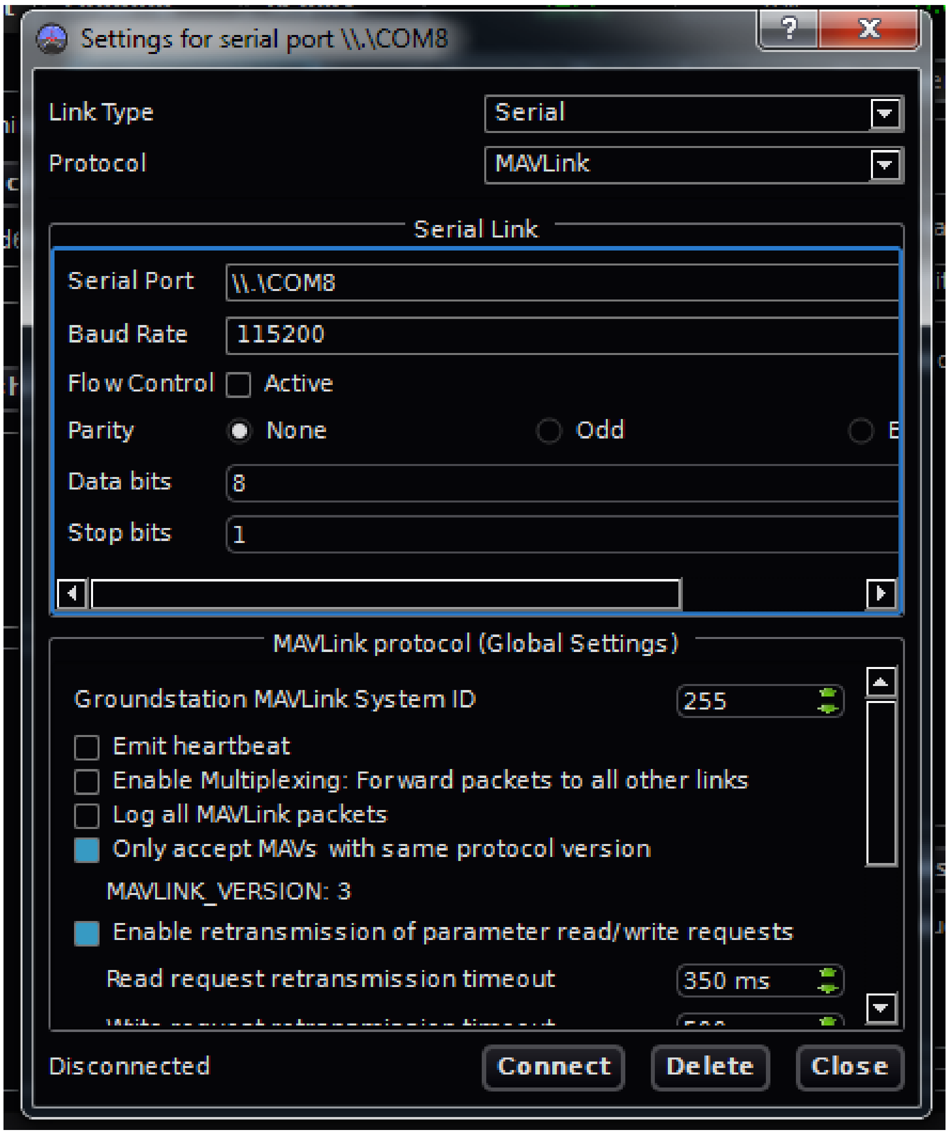
After installing the QGC and starting it for the first time, after selecting the “Engineer” tab, go to the top toolbar/menu, select ‘ *Main Widget* ‘ and then ‘ *AutoQuad* ‘ in order to bring up the AQ controls.

The AutoQuad widget can be used to flash the firmware, set all parameters, display log files, and control and set the ESC32.

Note: if you’re having problems to save/store values and parameters using the current firmware revision (r76), please read Ruffio’s Post in our forum.

Connecting to a flashed (ready) AutoQuad flight controller

In QGC, select *Communication* -> *Add Link* from the top menu bar. This will bring up a dialog similar to this:



Settings for serial port \\.\COM8

Link Type: Serial
Protocol: MAVLink

Serial Link
Serial Port: \\.\COM8
Baud Rate: 115200
Flow Control: ☐ Active
Parity: ● None ○ Odd ○ E
Data bits: 8
Stop bits: 1

MAVLink protocol (Global Settings)
Groundstation MAVLink System ID: 255
☐ Emit heartbeat
☐ Enable Multiplexing: Forward packets to all other links
☐ Log all MAVLink packets
☑ Only accept MAVs with same protocol version
MAVLINK_VERSION: 3
☑ Enable retransmission of parameter read/write requests
Read request retransmission timeout: 350 ms

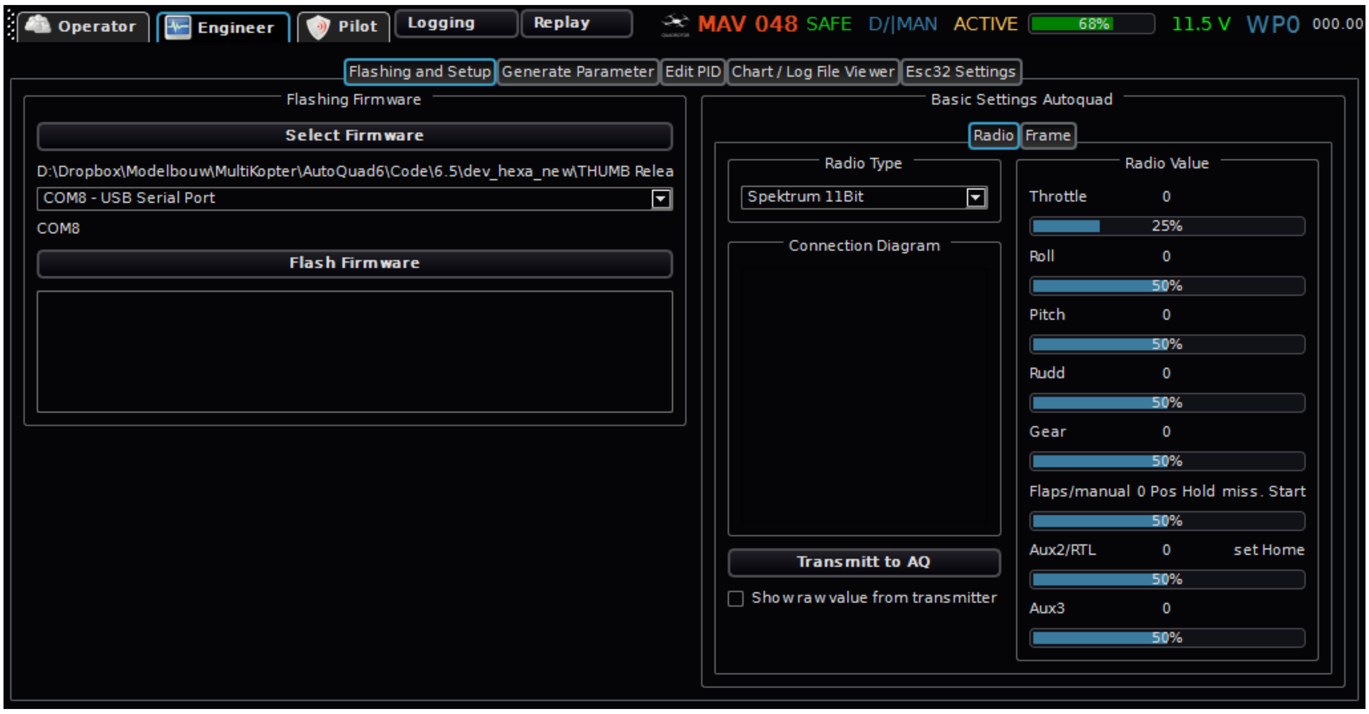
Disconnected

Here we have two options for Flow Control: If you have an FTDI interface that has the DTR pin (number 6 on the header) connected **select flowcontrol active** . If you have a Xbee, Bluetooth adapter of a FTDI interface that will pull the DTR down (and thus prevent communicating) **de-select flowcontrol active** like in the screenshot

1. Make sure the FTDI/USB adapter is connected to the computer and the AutoQuad. A serial port should be available once the adapter is inserted into the computer (COM8 in this example.) If the COM port does not show up, make sure you have the correct drivers installed for your adapter.
2. Select the COM port and the correct Baud Rate. By default that is 115200, unless you have previously changed it in the AQ parameters. Make sure the other settings match the screenshot — Parity: none; Data Bits: 8; Stop Bits: 1.
3. Click on *'Connect* '.
4. AutoQuad should now be connected and the QGC *Communication Console* windows will display a lot of data. The heads-up display (HUD) will display the orientation of the flight controller board. The AutoQuad widget will now be

populated with the PID settings (in the *Edit PID* tab).

AutoQuad Main Widget



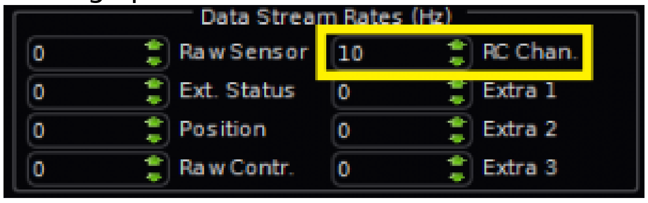
D:\Dropbox\Modelbouw\MultiKopter\AutoQuad6\Code\6.5\dev_hexa_new\THUMB Relea
COM8 - USB Serial Port
COM8

Radio Type: Spektrum 11Bit

Throttle	0	
25%		
Roll	0	
50%		
Pitch	0	
50%		
Rudd	0	
50%		
Gear	0	
50%		
Flaps/manual	0	Pos Hold miss. Start
50%		
Aux2/RTL	0	set Home
50%		
Aux3	0	
50%		

The first tab in the widget is used for flashing firmware, setting the radio type, monitoring the radio channels, as well as setting up your motor mixing table in the '*Frame*' tab (more on that later). See the Firmware section of the documentation for flashing procedure.

The default radio monitor output is updated at 1Hz, which may appear to "lag" behind your actual radio movements. If you want to display the radio output closer to real-time, you can increase that update rate to a higher value, like 10Hz for example. Look for this settings panel below the ' *Onboard Parameters* ' widget in QGC:

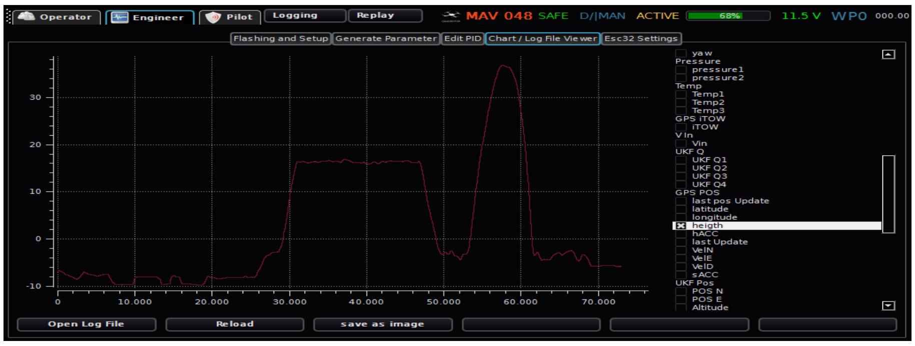

Data Stream Rates (Hz)

0	Raw Sensor	10	RC Chan.
0	Ext. Status	0	Extra 1
0	Position	0	Extra 2
0	Raw Contr.	0	Extra 3

The *Generate Parameter* tab is used to generate the calibration files. This is explained in detail in the Calibration Section.

The third tab, *Edit PID*, is for display and setting of all AutoQuads parameters like PID settings, battery voltages, etc. Most of these are explained in the Tuning section of this documentation and in the section about Gimbal Settings.

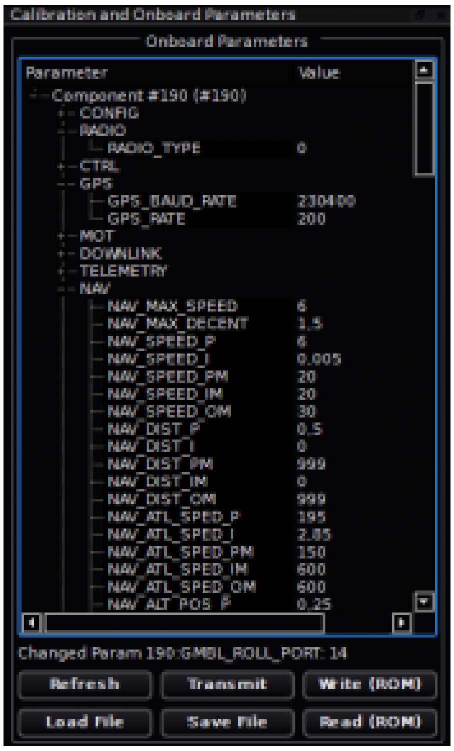
The *Chart/Log File Viewer* tab is used to display AutoQuads log files that can be recorded on the SD card. You can display and compare different values. More information about using the log viewer can be found in the Log File Analysis section.



The last tab is used or configuring, flashing, and telemetry display of the ESC32. More details are to be found in the ESC32 section.

Setting parameters not in main screens

AutoQuad uses a lot more parameters then present in the main screen widget. To view, edit, and transmit those parameters you can use the widget titled '*Calibration and Onboard Parameters*'.


Calibration and Onboard Parameters

Onboard Parameters

```
Parameter                     Value
- Component #190 (#190)
  + CONFIG
  - RADIO
      RADIO_TYPE              0
  + CTRL
  - GPS
      GPS_BAUD_RATE           230400
      GPS_RATE                200
  + MOT
  + DOWNLINK
  + TELEMETRY
  - NAV
      NAV_MAX_SPEED           6
      NAV_MAX_DECENT          1.5
      NAV_SPEED_P             6
      NAV_SPEED_I             0.005
      NAV_SPEED_PM            20
      NAV_SPEED_IM            20
      NAV_SPEED_OM            30
      NAV_DIST_P              0.5
      NAV_DIST_I              0
      NAV_DIST_PM             999
      NAV_DIST_IM             0
      NAV_DIST_OM             999
      NAV_ATL_SPED_P          195
      NAV_ATL_SPED_I          2.85
      NAV_ATL_SPED_PM         150
      NAV_ATL_SPED_IM         600
      NAV_ATL_SPED_OM         600
      NAV_ALT_POS_P           0.25
```

Changed Param 190:GMBL_ROLL_PORT: 14

In this widget you can open the parameter tree, browse to a needed parameter, and double-click on the value to change it.

- Use the '*Transmit*' button to save changes to the flight controller.
- '*Write (ROM)*' to permanently store those settings on the flight controller (otherwise

they will only keep until the next time the AQ is rebooted).

- ‘*Refresh*‘ to check the settings currently active on the AQ.
- ‘*Read (ROM)*‘ to load the settings saved in ROM into the flight controller’s active memory (this does NOT refresh them on the screen, you have to hit the ‘*Refresh*‘ button to do that).
- ‘*Save File*‘ will save the currently displayed settings to a file of your choice.
- ‘*Load File*‘ currently does not do anything when connected to AQ.

How to use Live Telemetry display to check your sensors

[Link to embedded object]

Ground Control Station for Android

Ground Control Station for Android

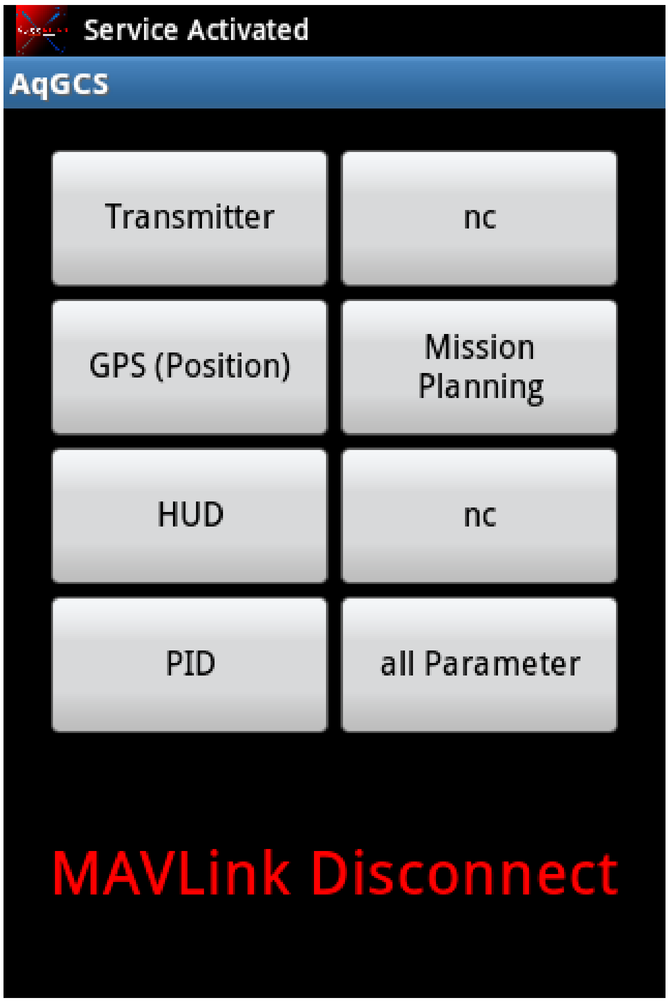
AutoQuad has an Android-based ground station. Originally based on the , it is tailored for AutoQuad with some nice enhancements. You can download the latest version in the AQ site downloads section.

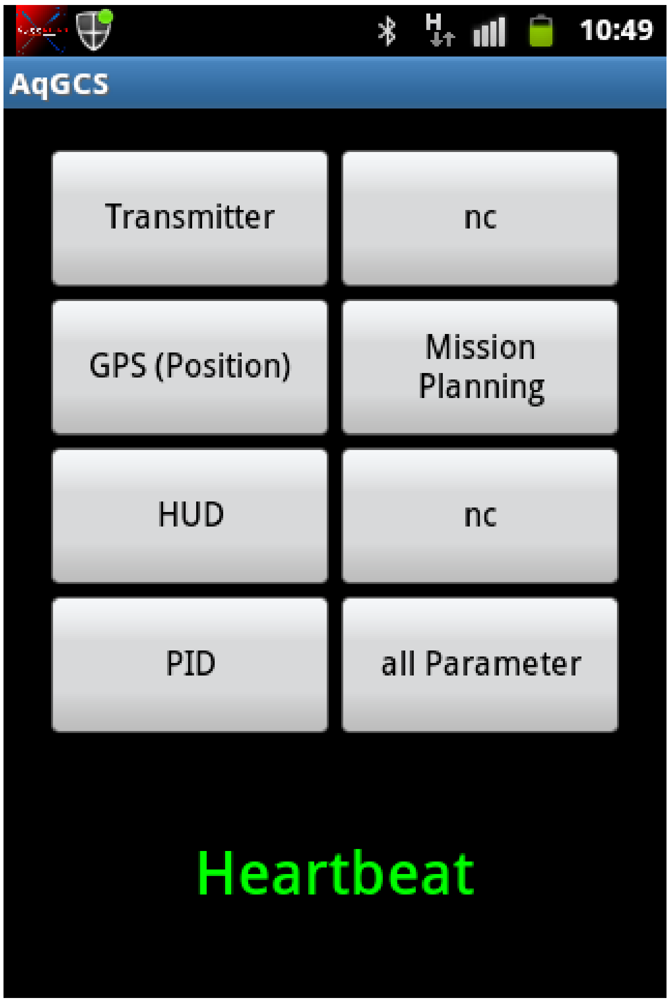
The current version supports:

- Using Android Bluetooth.
- Android 2.2 and later.
- HUD (Head Up Display) with battery voltage, altitude, GPS info.
- PID settings – display and configure.
- Access to all parameters.
- GPS location of AutoQuad.
- Full screen mission planner with waypoint editor.
- Text-to-speech — all AutoQuad messages can be spoken!
- Radio output screen (diagnostics).

The prerequisites are simple: an Android device with Bluetooth, and an AutoQuad flight controller with Bluetooth adapter connected to the FTDI port. Start the app on your Android device, scan for the Bluetooth adapter attached to the AutoQuad flight controller and connect. If a heartbeat is received, the app will display the connection.

Note : The following screenshots are based on an older version of the Android GCS. New screenshots are forthcoming, but the basic functions remain the same.

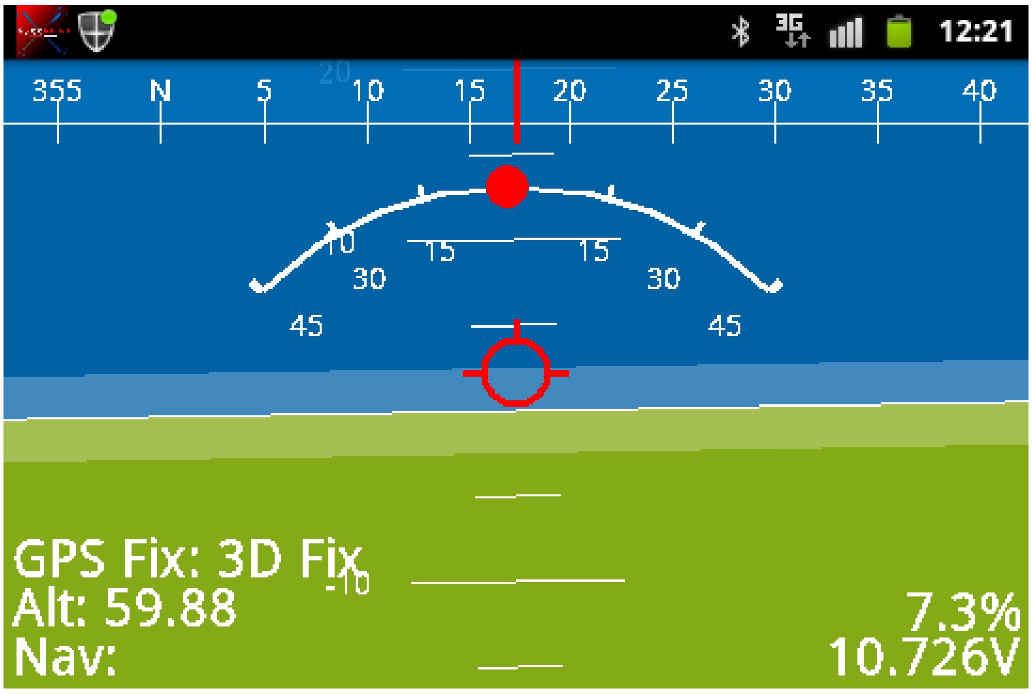




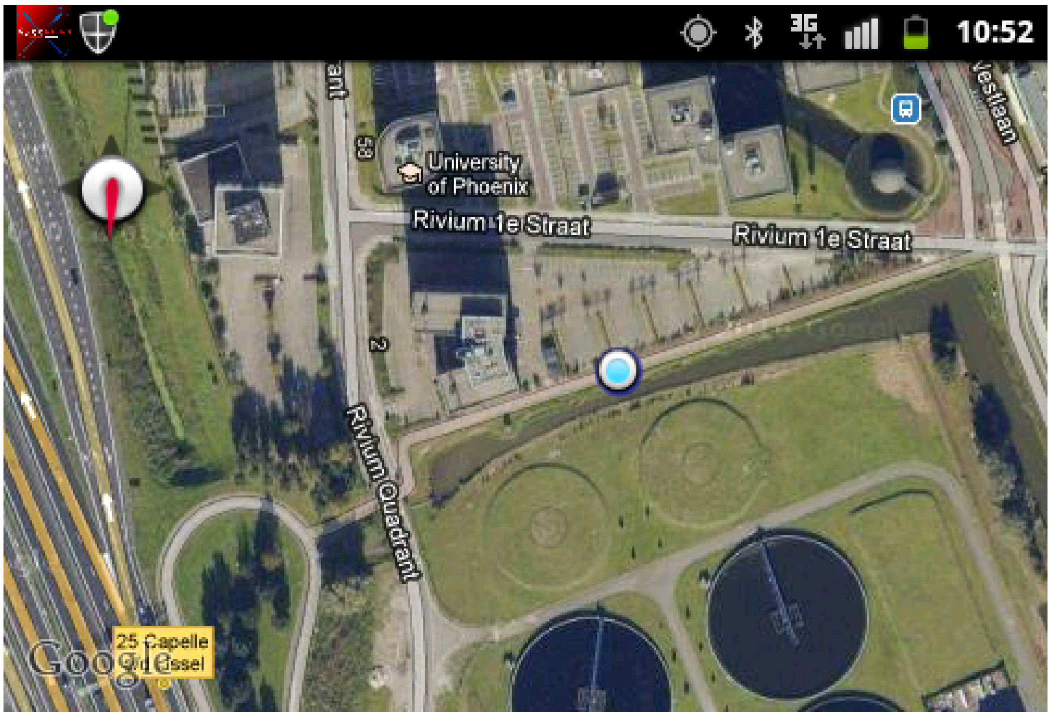
Main screen

Mavlink heartbeat received

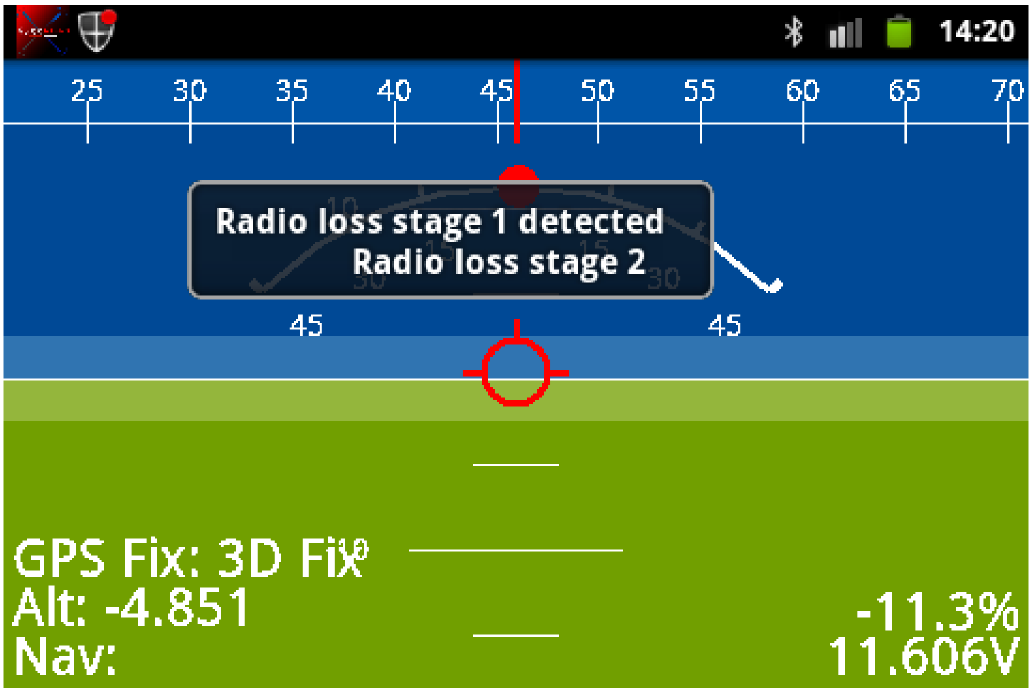
Accurate attitude, altitude, and battery voltage display:



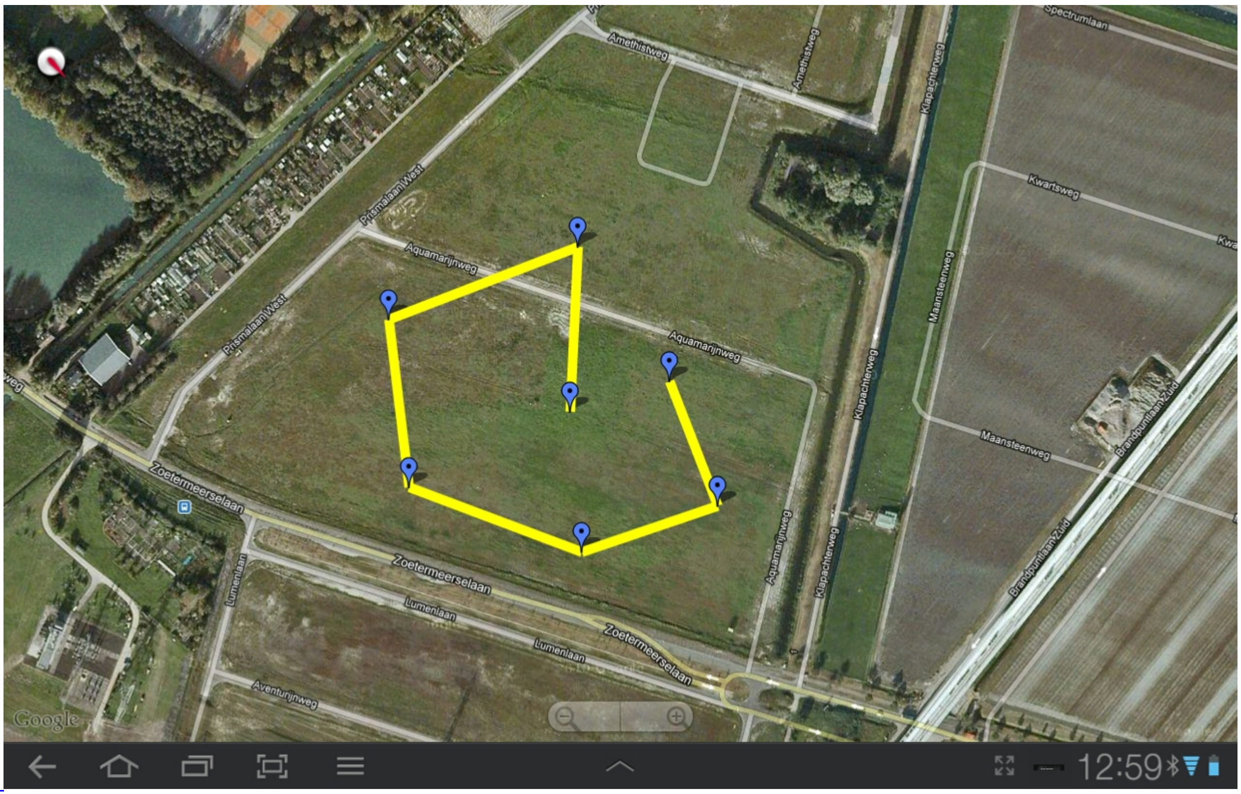
GPS location of AQ:



Text notification of important event:

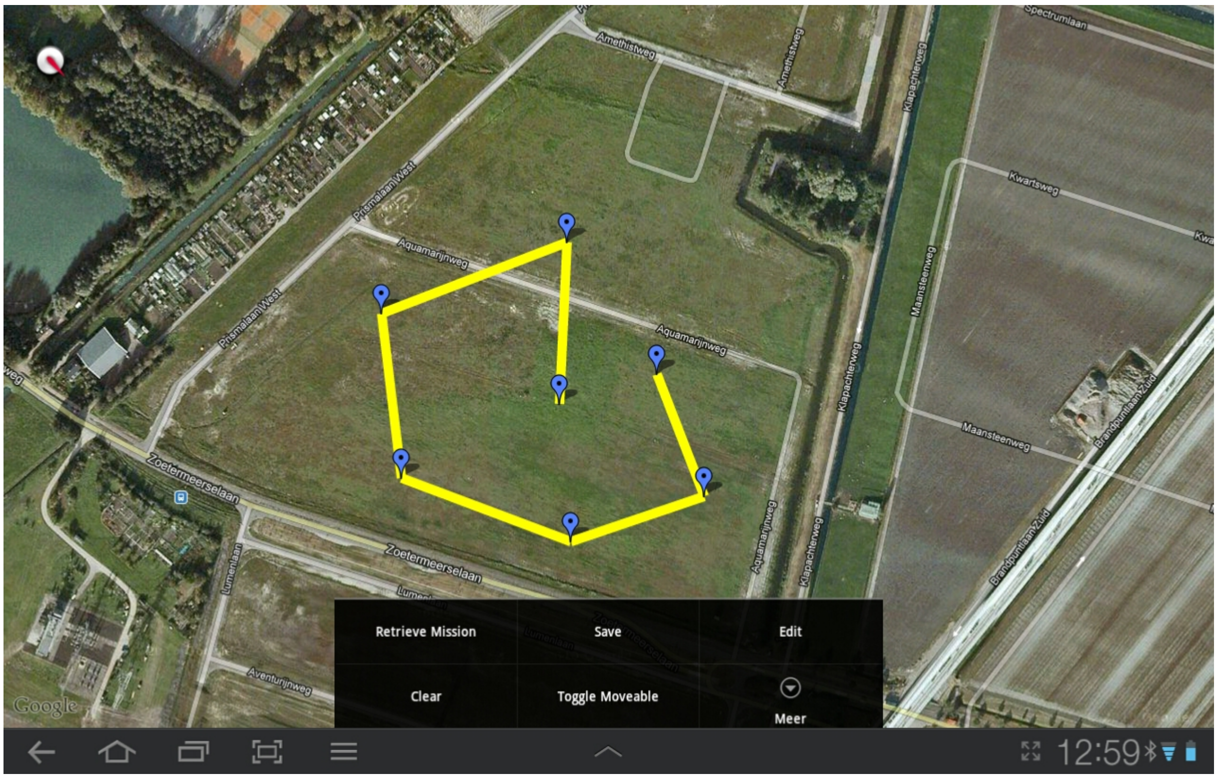


Waypoint editor :

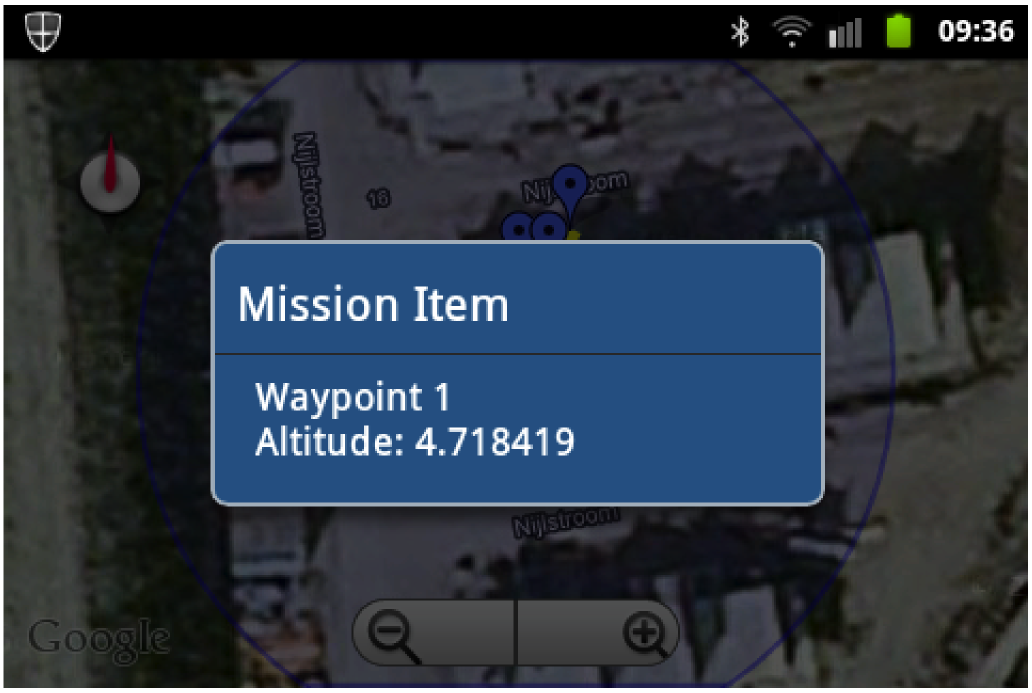


Tap and place a waypoint on the map screen.

Waypoint editor:

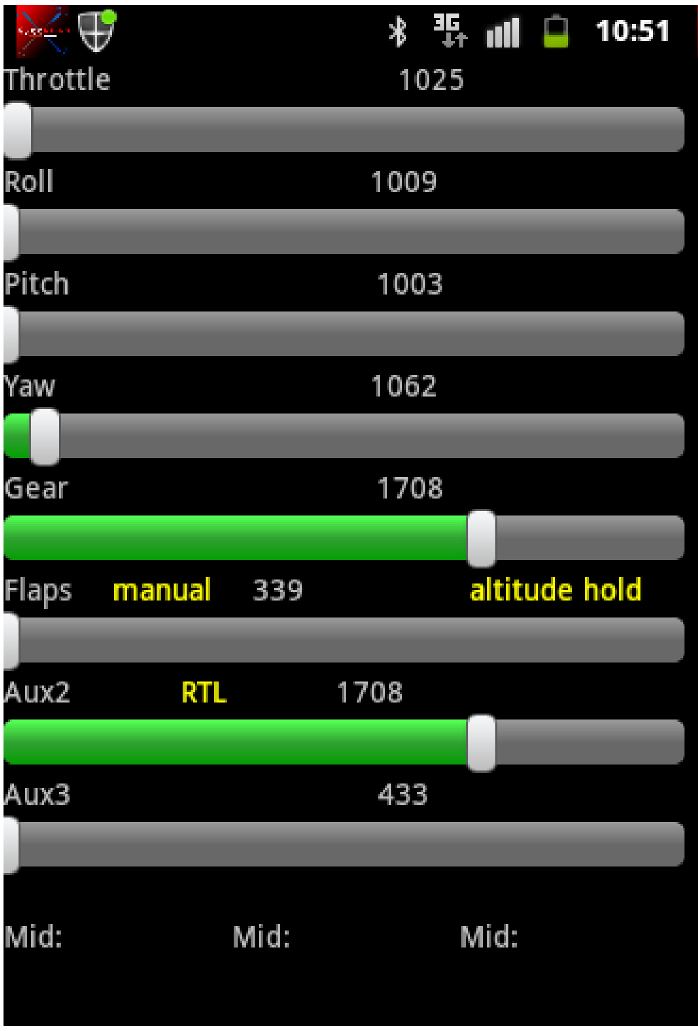


Press the menu button and Save Mission to upload the mission to AutoQuad.



Mission Item

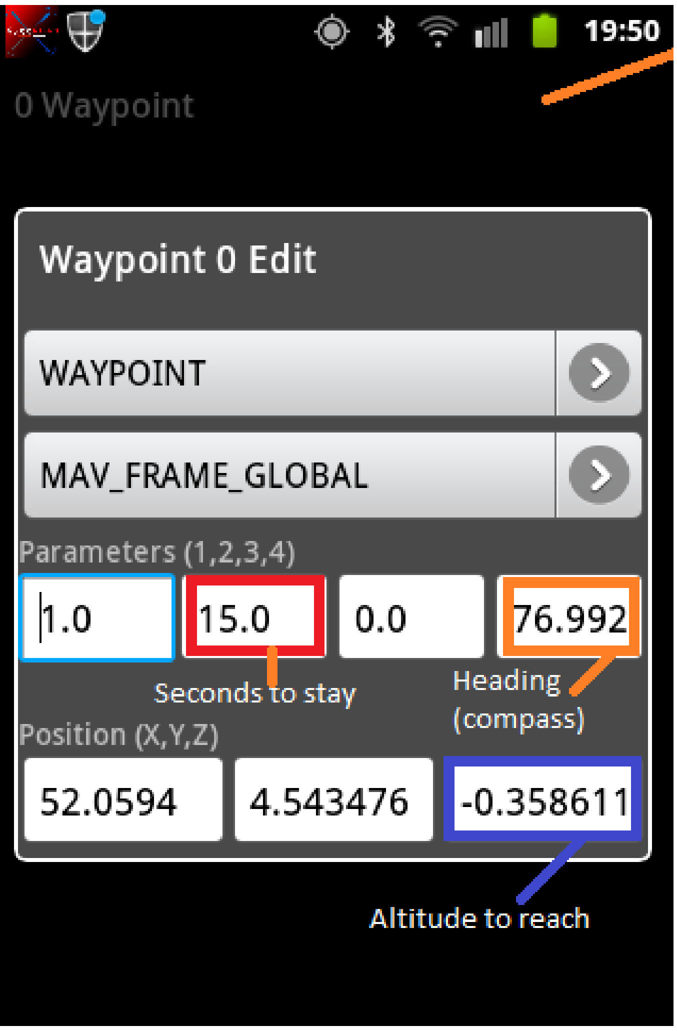
Waypoint 1
Altitude: 4.718419



10:51

Throttle	1025
Roll	1009
Pitch	1003
Yaw	1062
Gear	1708
Flaps manual 339 altitude hold	
Aux2 RTL 1708	
Aux3	433

Mid: Mid: Mid:



19:50

0 Waypoint

Waypoint 0 Edit

WAYPOINT

MAV_FRAME_GLOBAL

Parameters (1,2,3,4)

1.0 | 15.0 | 0.0 | 76.992

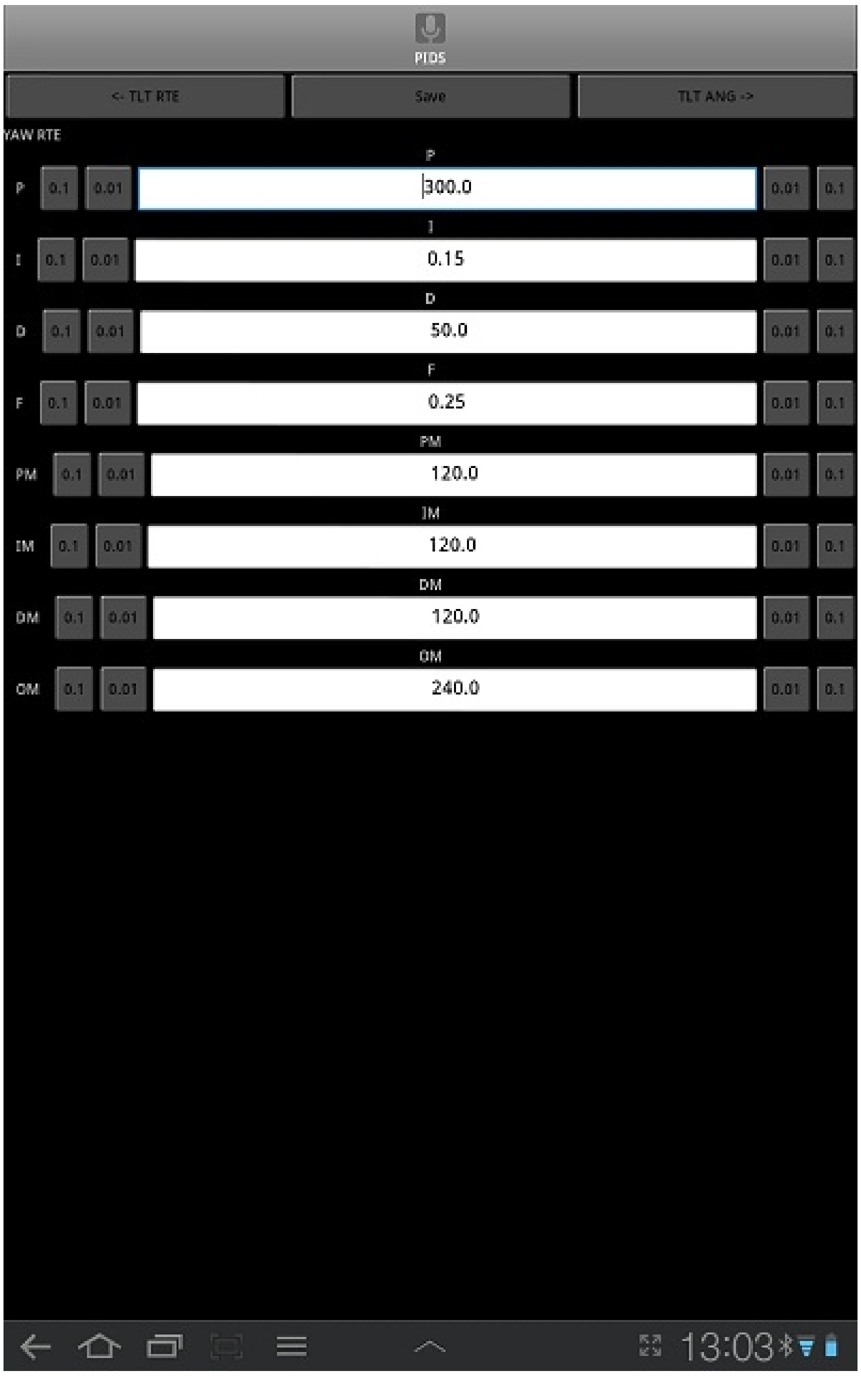
Position (X,Y,Z)

52.0594 | 4.543476 | -0.358611

Transmitter channel display *Waypoint editor*

Selecting a waypoint in the list and a short tap will display the editor. Here you can fine-tune the waypoint with required altitude, time to stay and move to the next waypoint and the type of waypoint.

PID editor



YAW RTE

P	300.0
I	0.15
D	50.0
F	0.25
PM	120.0
IM	120.0
DM	120.0
OM	240.0

Every available PID related value can be tuned here .
Please be careful, most values !Do Not! need to be changed.

Short intro video of the AQ Android GCS:

[Link to embedded object]

Configuring AutoQuad Flight Controller

Configuring AutoQuad Flight Controller

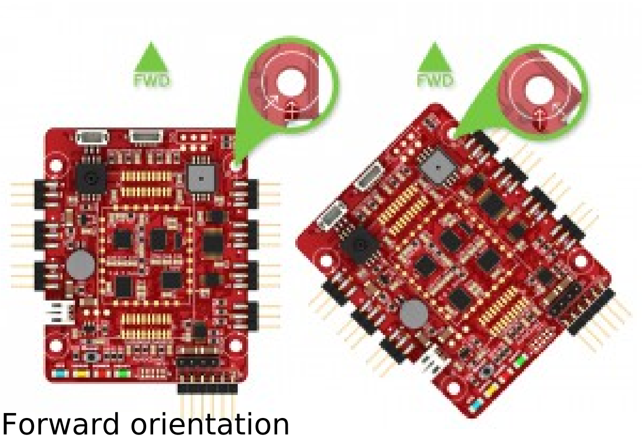
Basic setup

An AutoQuad ground control station is required to configure and update your board. While many settings can be adjusted using the app for Android, the full-featured QGroundControl for AQ is recommended for initial setup. Please see our downloads section.

After your firmware is uploaded, almost all settings can be configured from the QGC AutoQuad main widget. Switch to Engineer view, and if the map is still displayed, use the top *Main Widget* menu and select *AutoQuad* .

AutoQuad uses a large number of configurable parameters. Not all parameters are directly visible in the AutoQuad widget.
All available parameters can be viewed and configured using the “All Parameters” or “Onboard Parameters” widgets. These are by default visible in the “Config” screen. Just open the tree, look for the parameter you need, edit, and transmit to AutoQuad.

Flight direction



Forward orientation

The first image will show the default flight direction (IMU_ROT 0), but the parameter IMU_ROT allows for any angle in degrees. The second picture will need an IMU_ROT -45 for example.

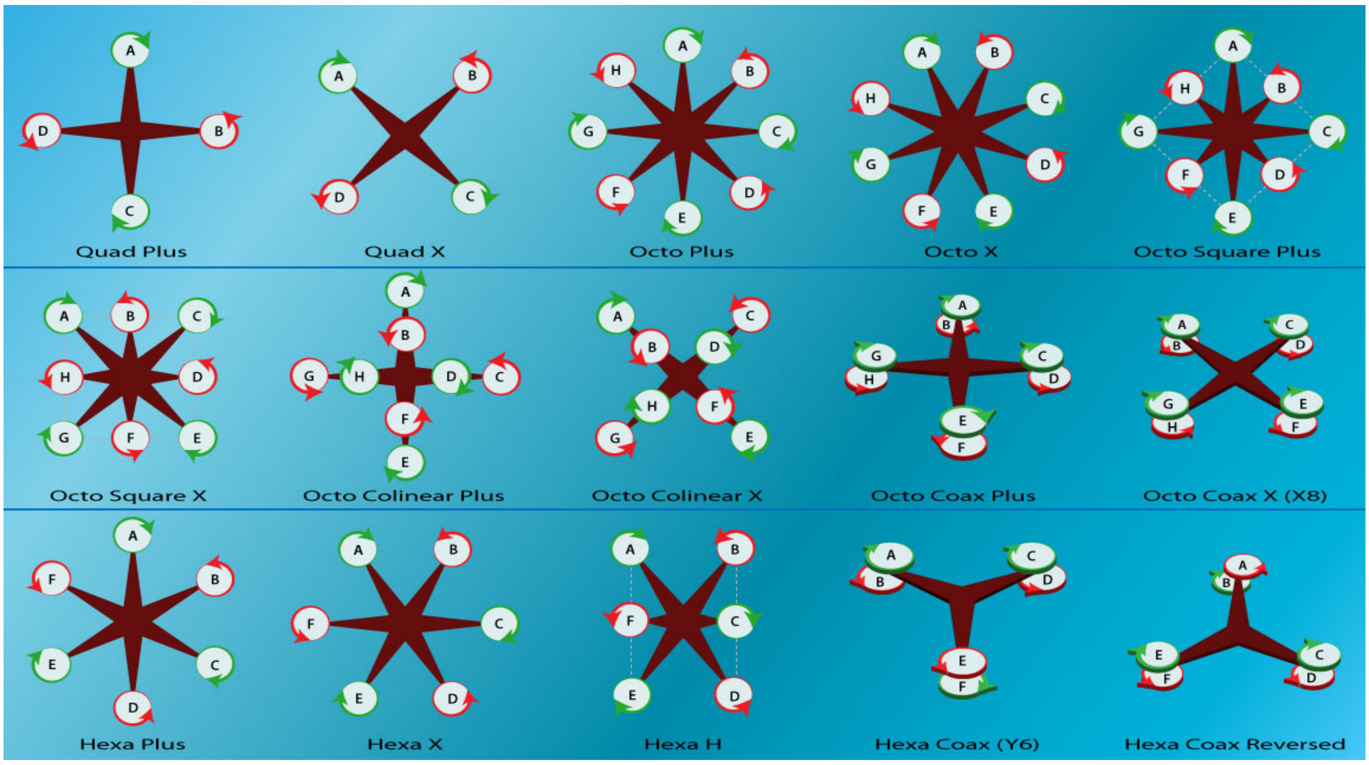
Please continue to the following sub-sections for more specific setup details.

- Motor Mixing Table Setup

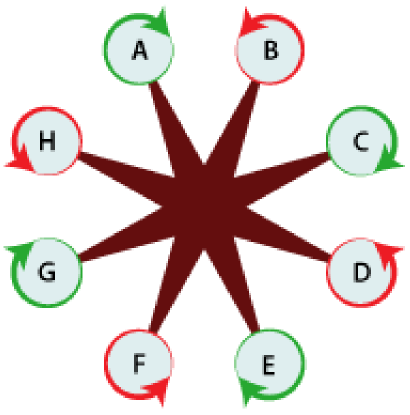
Motor Mixing Table Setup

Motor Mixing Table Setup

QGroundControl for AQ (v1.2.0 and up) comes with a number of pre-defined mixing tables for various frame types. But, there are no limits on how to configure the motors on an AutoQuad. AQ will control any motor output you want — you just have to mix it on the right channel, with the correct direction and amount of output per axis. The pre-defined mixes should serve as-is in most cases, but you always have the option to customize the outputs for your particular application.



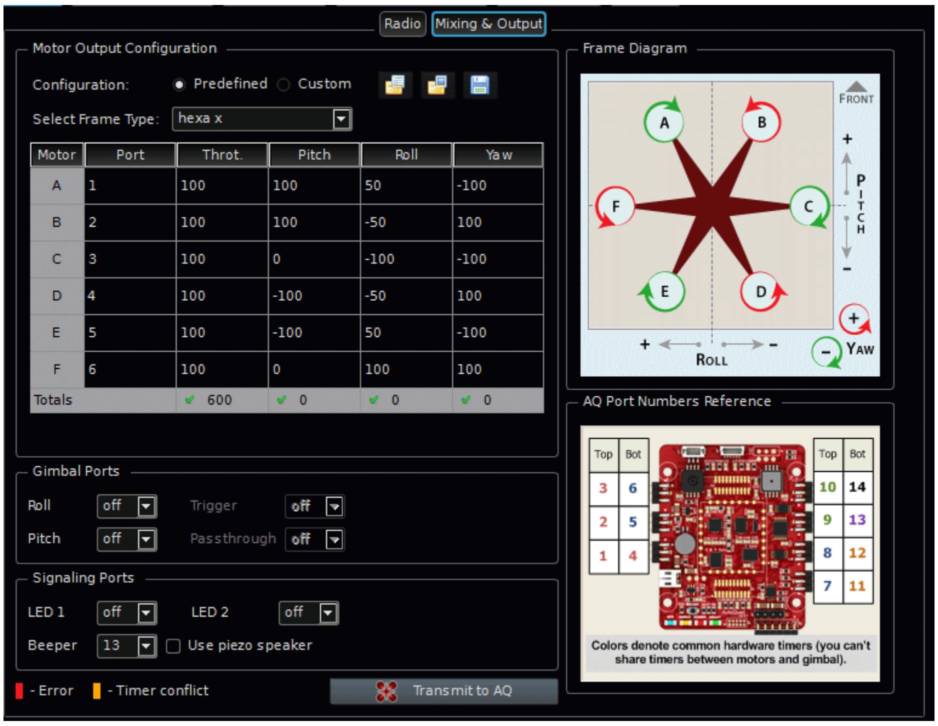
Choosing which ports to use



Whether you choose a pre-defined or a custom mix, you also get to decide which ports you want your motors plugged into. This is a departure from "typical" multicopter flight controllers which pre-define the ports used for various frame setups. Usually you will want to choose the ports closest to your motors, to reduce cabling and possible interference, but you're free to choose any ports you want. That is why in AQ frame diagrams you will not see port numbers on the motors, but just letters instead to show the intended order of the motors. The port numbering is up to you.

Note: If you are planning to control gimbal servos/motors from the AQ, you must be more selective in your port choices. Please refer to more information on [PWM frequencies](). The QGC UI (v1.2.0+) will also validate that you do not have conflicting port selections.

Standard mixing setup



Motor Output Configuration

Configuration: [✓ Predefined] Custom

Select Frame Type: hexa x

Motor	Port	Throt.	Pitch	Roll	Yaw
A	1	100	100	50	-100
B	2	100	100	-50	100
C	3	100	0	-100	-100
D	4	100	-100	-50	100
E	5	100	-100	50	-100
F	6	100	0	100	100
Totals		✓ 600	✓ 0	✓ 0	✓ 0

Gimbal Ports

Roll	off	Trigger	off
Pitch	off	Passthrough	off

Signaling Ports

LED 1	off	LED 2	off
Beeper	13	☐ Use piezo speaker	

🟥 - Error 🟧 - Timer conflict

AQ Port Numbers Reference

Top	Bot
3	6
2	5
1	4

Top	Bot
10	14
9	13
8	12
7	11

Colors denote common hardware timers (you can't share timers between motors and gimbal).

Motor output is configured on the *Mixing & Output* tab of QGroundControl for AQ. To use one of the pre-defined frame types, simply make sure the *Predefined* radio button is selected and choose a frame type from the *Frame Type* options box below that. You can leave the port numbers as they are, or customize them as you see fit. Use the pictures for reference. Although you can customize the numbers in each of the other columns, you probably don't want to until you have a solid understanding of the mixing theory, and a good reason to adjust the outputs (more on this below).

If you want to define your own mix for a new type of frame, simply choose the Custom option, and then select the number of motors you need. The rest of the setup is essentially the same. You don't get any frame image in the diagram, but you can load your own using the button (ideally your image should be, or scale to, 200x200px, and have a transparent background).

Use the and buttons to save and load configurations to/from your local file system. Use the big *Transmit to AQ* button at the bottom so save your changes to AutoQuad (when connected). If you want your changes saved to non-volatile memory on the AQ (to survive a restart), be sure to answer “Yes” to the resulting prompt.

Now simply connect your ESC signal inputs to the ports you set up for each motor, and be sure to set proper rotation direction for the motor/prop, as defined in the Yaw column and corresponding diagram.

Please note: When you connect to the AQ for the first time after starting QGC, it will load whatever configuration is currently on-board. If you have made changes to the mixing table (or anything else), you will loose them unless they are saved to a file. After the initial connection, the settings are only re-loaded when you hit one of the parameter *Reload* buttons elsewhere in the QGC UI.

Also: The provided frame diagrams will NOT update themselves to your configuration… for example if you reverse the yaw in your setup, or choose to follow a different motor order, the diagram will then be wrong. Better to use a custom config in these cases to avoid confusion.

Motor mixing theory

The following sections will attempt to explain how to customize the mixing setup and why you might want to. This may be a bit confusing at first, so look through the examples here and the mixing tables provided with QGC. The arrows and +/- signs in the frame reference diagram are explained here.

What do those Throttle/Pitch/Roll/Yaw numbers in the table mean? They correspond to the percentage of output given to that motor for each of the movement planes/axes. The +/- part determines the direction of movement that motor influences (left/right/CW/CCW… more on that later). The absolute number, 0-100, corresponds to the percentage of the total given command for that particular control direction, which is decided by the internal AQ calculations.

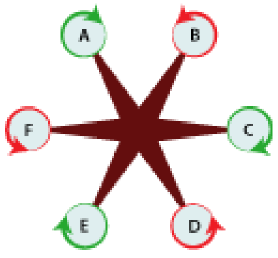
So for example, say the AQ thinks it needs to give X amount of power to the pitch axis in order to move in a certain direction (or maintain a level attitude). It will then try to distribute that X amount to each motor, as defined in the mixing table for the pitch axis. If a motor is configured with a “100″ for the pitch value, then 100% of the desired power will be applied to that motor to speed it up. If zero is configured, then no pitch control would be applied at all. (Ignore for now the mathematical conundrum that some motors have negative values… that is explained in more detail later.)

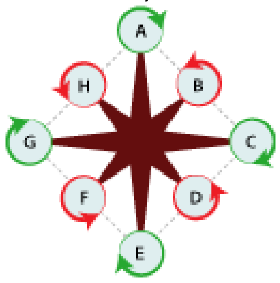
It is important to understand that the AQ has other controls to avoid saturation of the motors, mostly via PID settings and some internal checks. So configuring a motor for “100%” doesn’t mean that 100% of all available power is sent to that motor. With correct PID settings, the AQ takes care of proper overall power distribution (within physical limits, of course).

You might also ask why would we need more than 100% power per direction of movement? To get there quicker, of course! The more total power applied to the movement direction, the quicker the desired attitude is achieved. More power also means better ability to fight wind and turbulence, or to recover from difficult situations. Of course in some cases you may not need or want that power… read on.

Balancing your movements

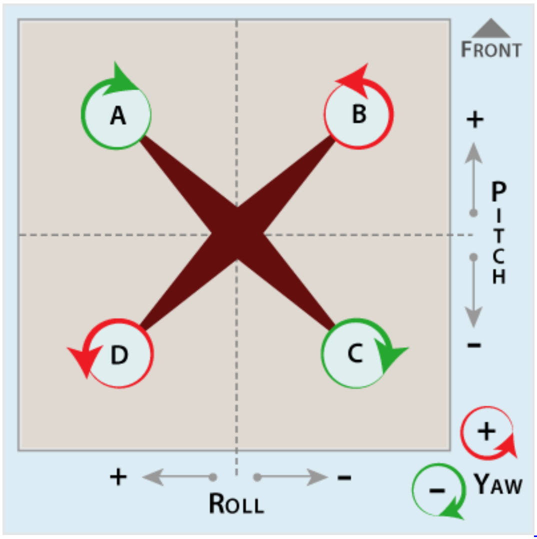
Which brings us to WHY you might want to change the mixing table from just having +/-100 on each control. This varies depending on which control direction we're talking about:

- **Throttle** – There is probably no reason to reduce this from 100% unless you have a very overpowered setup, or have a frame where some motors need to provide less lift than others. At any rate, you want to make sure these are evenly distributed across all motors, so the multicopter doesn't lean violently when throttle is applied. Note that the Throttle channel only takes positive values, since there's no such thing as "negative" throttle (until we develop collective pitch control, perhaps 😉).
- **Pitch and Roll** – 

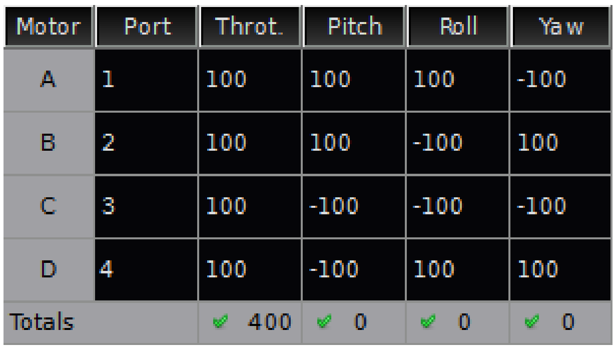
These are the most commonly adjusted values. The reason is simple (sort of!)… the AQ has only one set of PID controls for both the roll and pitch axes. As mentioned before, PID settings are important for overall power distribution. Since we only have one set of these important controls, we need to ensure that the overall power available to the pitch and roll axes is equalized. For example in a Hexa X setup, you have 3 motors per side affecting roll movement, but only 2 motors affecting pitch movement. So, we can use the mixing table to limit the output to some motors in order to help make the roll and pitch rates more even, which in turn will help the PID settings do their job properly (actual example follows).
- **Yaw** – 

Again, there isn't often much reason to change yaw output amounts. But again the point here is to equalize the amount of power/torque in each control direction (clockwise and counter-clockwise). If for example your CW props are bigger/steeper than your CCW props (for some strange reason), you would need to reduce their output, or your multi will yaw unevenly (and will struggle to maintain heading). Or, consider the frame at right. Since the outer motors have a larger influence on yaw than the inner motors, you would, theoretically, need to figure out how much less power they need than the inner motors.

Controlling movement direction



The positive vs. negative signs simply control direction of movement. For roll and pitch, imagine the multicopter divided into 4 quadrants, each having a + or – for both pitch and roll axes. For yaw, it's simply + for CCW, and – for CW rotation. Refer to the setup diagram on the left to determine which sign corresponds to which direction.



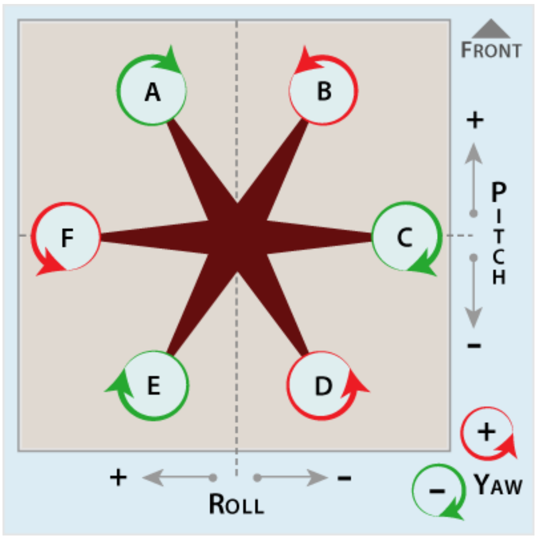
Motor	Port	Throt.	Pitch	Roll	Yaw
A	1	100	100	100	-100
B	2	100	100	-100	100
C	3	100	-100	-100	-100
D	4	100	-100	100	100
Totals		✓ 400	✓ 0	✓ 0	✓ 0

On the right is the the simple mixing table for this quad. In the diagram you can see that motors A and B lie in the "+" pitch direction, so in the mix table their numbers are positive. On the other hand, the C and D motors lie in the "-" pitch direction, so their numbers are negative. For Roll, A and D lie in the positive direction, while B and C are in the negative. Since A and C rotate CW, they have negative yaw, while B and D have positive numbers. Simple, right?

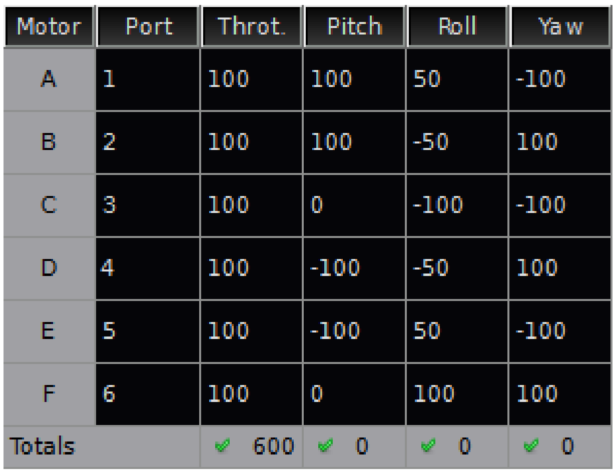
One more **important** thing about **pitch, roll,** and **yaw**, direction. You want the totals of those columns to **add up to zero**. Otherwise you will most likely have a serious imbalance problem. The QGC UI keeps a running total of the columns for you, and warns you if you they are not correct. When the AQ boots it will also issue a warning message if it detects an imbalance. However, neither will prevent you from actually flying with an imbalanced setup if you really want to!

Putting it all together (practical examples)

Hexa X



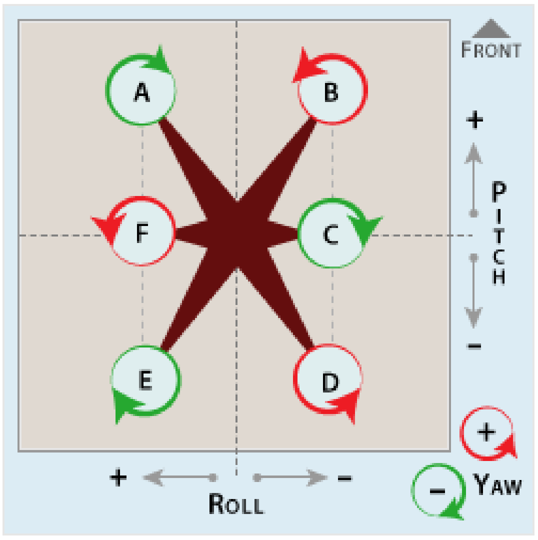
OK, let's take a closer look at that Hexa X mix as an example. We have 2 motors per side on the pitch axis, and 3 per side on the roll axis. If we set them all to 100%, the multicopter will roll faster than it can pitch. This might work OK in some situations, but it's probably not ideal.

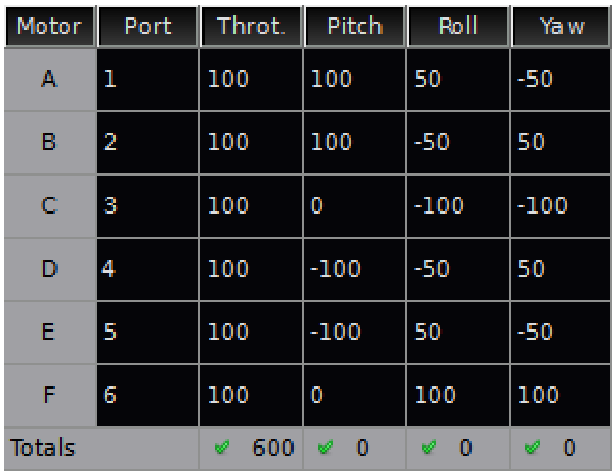


Motor	Port	Throt.	Pitch	Roll	Yaw
A	1	100	100	50	-100
B	2	100	100	-50	100
C	3	100	0	-100	-100
D	4	100	-100	-50	100
E	5	100	-100	50	-100
F	6	100	0	100	100
Totals		✓ 600	✓ 0	✓ 0	✓ 0

To balance the axes movement rates, we simply reduce the output of some of the roll motors. There are many ways to achieve this, but the simplest is to reduce the roll output to A, B, D, and E by 50%. Now we have 200% total output in each direction on roll as well as pitch axes.

Hexa H





Motor	Port	Throt.	Pitch	Roll	Yaw
A	1	100	100	50	-50
B	2	100	100	-50	50
C	3	100	0	-100	-100
D	4	100	-100	-50	50
E	5	100	-100	50	-50
F	6	100	0	100	100
Totals		✓ 600	✓ 0	✓ 0	✓ 0

Here's another interesting example where we need to adjust the yaw balance as well. You can see this example is essentially the same as the Hexa X, above, but the F and C motors are much closer to the center of the frame than the other 4. If you give all the motors 100% yaw influence, whenever you yaw, the hexa will want to roll to one direction or the other. For example if you yaw to the left, CCW, B, D, and F motors will speed up, while the other slow down. Now you have 2 motors on long arms trying to roll the hexa to the left, and only one motor on a short arm to counteract that movement. Obviously, not so good. The answer, of course, is to reduce the yaw output of the outer motors so they don't throw the hexa off balance every time you yaw.

Further reading

Motor mixing is not always a simple subject and much discussion has surrounded it. Some people have found it easier or more intuitive to tune an AQ's performance by using the mixing table settings. Generally it has been found that unless you have one of the conditions described above, it is better to use as much output per motor as possible. There are exceptions to this rule, for example some very light or overpowered multicopters might need to lower some outputs to prevent over-control. One interesting set of findings has been published by team member Angel (afernan) in this document, following a lengthy diagnostic process described in this forum thread. As always, use your best judgement, and experiment carefully!

Radio connections

Radio Connections

The AutoQuad has one dedicated interface available for connecting a receiver with a serial connection like the Spektrum satellites and Futaba S-Bus. PPM uses a PWM connector (M14 by default). More about radio options

The white arrow on the PWM outputs marks the signal pin for the ESC’s or PPM input. Always connect the GND cable from ESC or PPM receiver to the PWM connector GND pin.

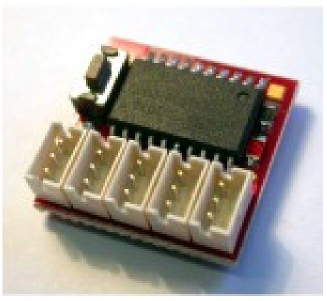
Note: If you’re facing drop outs on the throttle channel (sudden motor glitches), power your receiver and converter via external BEC instead of powering it via the FC or ESC.

Warning: Always power your transmitter first before you power your receiver!

Every radio type change needs to be transmitted to the AutoQuad, saved to ROM, and AutoQuad needs to be reset before the changes will take effect.

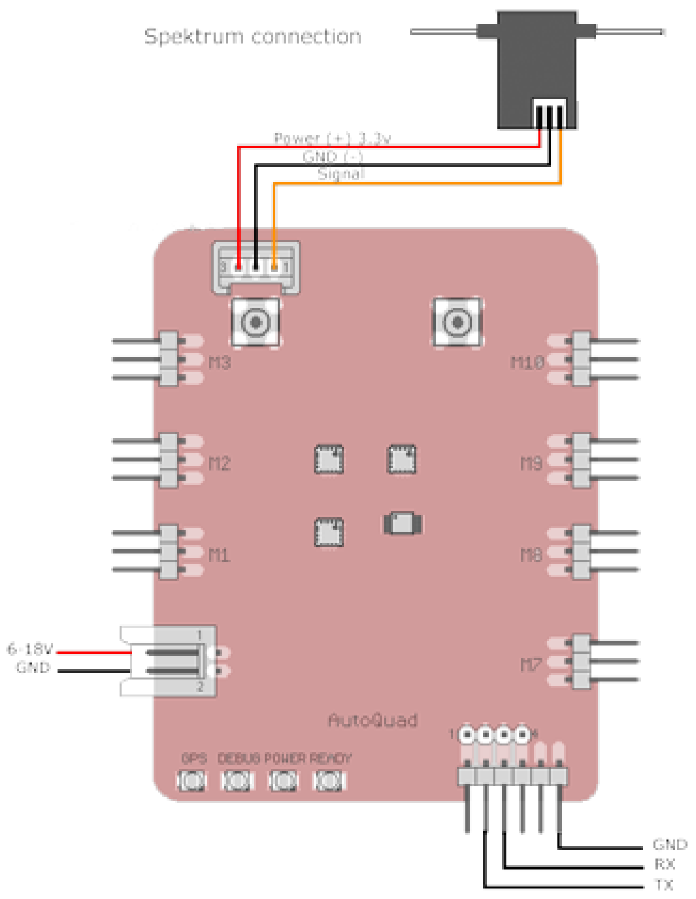
Spektrum satellite

One Spektrum (or compatible) satellite can be directly attached to the AutoQuad. If more range or diversity is needed, a satellite diversity board can be purchased to attach more satellites. At the time of writing, the binding is not yet possible from within the firmware so a bind must be made with a full Spektrum receiver and the satellite attached. If you use a Spektrum diversity board, binding can be done with the switch on the diversity board.



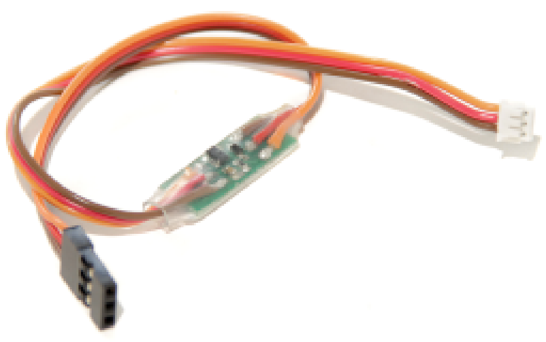
Spektrum diversity board

The standard Spektrum cable is to be used, but to be complete, a diagram of the connection is shown.

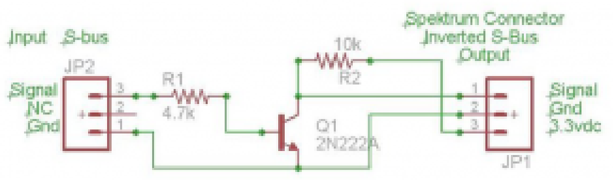


Futaba S-Bus

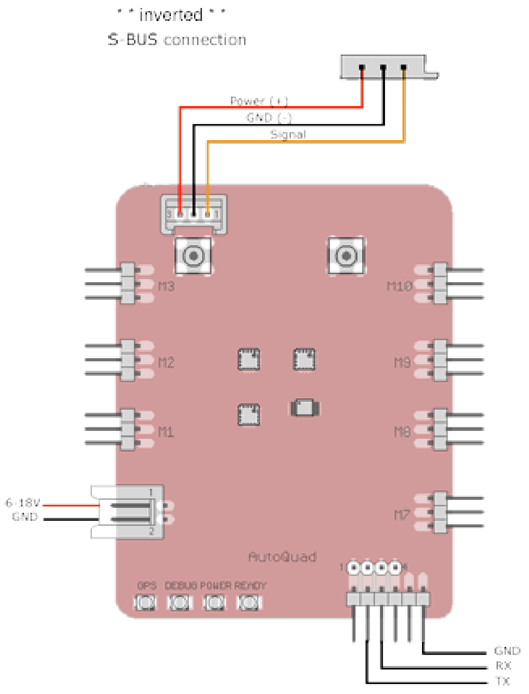
One S-bus receiver (or compatible) can be directly attached to the AutoQuad. As with most flightcontroller, **you need to invert the signal from the S-bus** . There are a number of [s-bus inverters]() for sale on various shops ($15).



Or build your own:



A diagram of the connection is shown.



It is possible that the s-bus receiver must be powered with a higher voltage than 3.3. In that case the power line must not be used from AutoQuad receiver connector but from a 5V line on the board. You can use any 5V output from the motor/pwm connectors.

S-Bus Receivers general Warning!

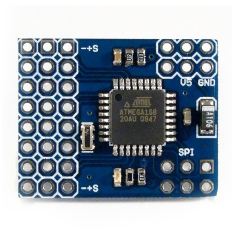
Make sure you always have your Transmitter powered first. On some FrSky and other Receivers send Zeros at startup, when the Transmitter wasn´t powered first.

Futaba SBE-1 S-Bus Converter and non-Futaba SBus receivers — Warning!

There have been 2 confirmed cases of the Futaba SBE-1 S-Bus adapter not working correctly with AQ. Investigations are still ongoing at this time, but this converter is not currently recommended. It is not clear if other converters would be affected as well, or if non-Futaba S-Bus receivers are fully compatible. Details and updates can be found at this thread in the forums .

PPM

PPM support is included in AQ experimental firmware v6.6 r22 and up. If your receiver supports PPM output, or can be converted to provide it, then that output can be used directly. In other cases, a PPM converter/encoder can be used between any standard Rx and AQ. Here is an example of an inexpensive converter known to work well with AQ.



Extra PPM info on the dedicated page .

We're currently also testing the **ASA Delta-8 multiprotocol receiver** for Hitec, Futaba and FrSky radios.

It supports CPPM output (motor output #14 on the AQ board) and is compatible to different protocols:
Hitec A-FHSS, Futaba S-FHSS, Futaba FHSS and FrSky V8 & D8.
It can be purchased starting at ~25$ at various locations. You can follow the discussion by developers and resellers here.

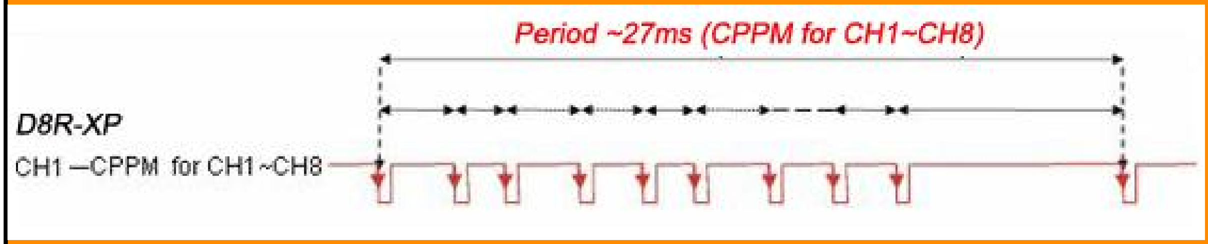
FrSky Rx failsafe

FrSky receivers (status summer 2012) need a special sequence for disabling the built-in FrSky failsafe. If that is enabled (by default), the AutoQuad controller will NEVER detect a failsafe and trigger a failsafe event.
For now the following procedure seems to disable the failsafe. Please test in the GroundControlStation if the AutoQuad detects a lost signal when the radio is shut down.

1. **Rebind** the receiver with the transmitter. (Power off the receiver, poweroff the transmitter, Power on the transmitter with bind button pushed and then power on the receiver with bind button pushed). For details see your FrSky documentation.
2. Power off the transmitter, power off the receiver.
3. Now ONLY power on the receiver (red light will blink), push the receiver bind button ONCE.
4. Power off the receiver. Failsafe is now disabled.

FrSky CPPM

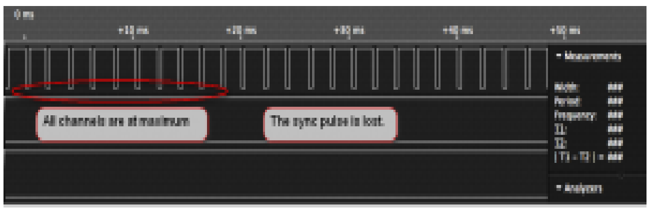
UPDATE: In late 2012, FrSky released an updated firmware version for D8R receivers which extends the PPM pulse length to 27ms. This solves the problem described below. Anyone using a compatible FrSky Rx with AQ is encouraged to update their Rx firmware ASAP.

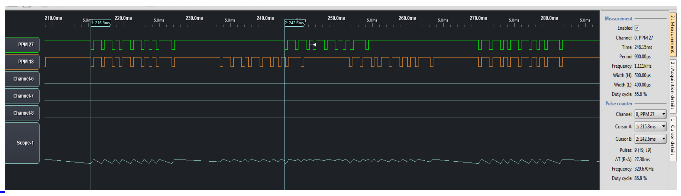


Warning about FrSky CPPM (original 18ms version) : A FrSky PPM frame is 18ms long. A standard channel moves from 1 – 2ms. When using 8 channels to the max it will come to 8 x 2 = 16ms, the (needed) PPM sync puls is then only 2ms. That can either be considered as a valid channel (very wrong) or at least it will not recognize it as a sync puls and discard it.

This obviously can create a dangerous situation that AutoQuad (or any other flight controller) can't solve. Either only use 7 channels with (FrSky) PPM, or limit the switch output channels to 60-70%. AutoQuad will still be able to use that output but the maximum PPM frame will be less occupied.

In normal circumstances this will probably never cause issues, but beware that FrSky is aware of this error but has yet not released a solution.
More on this topic on [this forum](#)





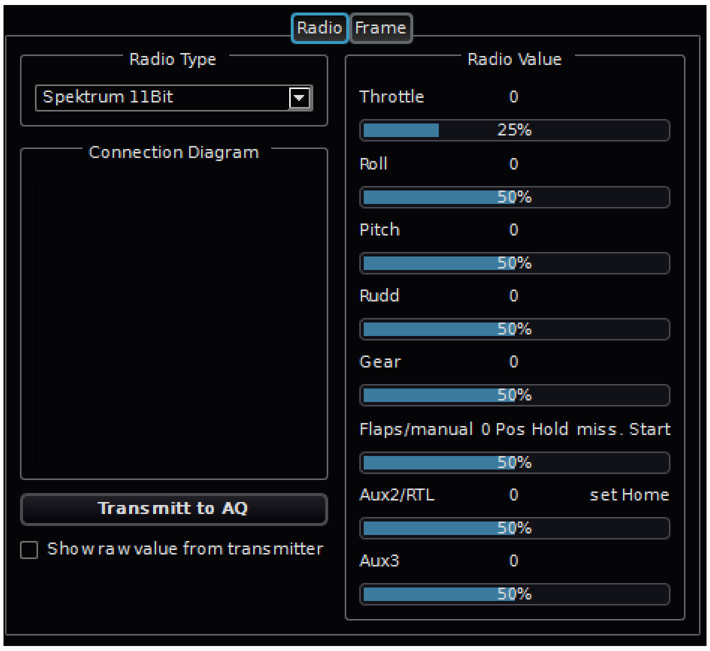
We're currently testing the SBus output of FrSky receivers with good results. So this might be a better option for users who are having trouble with PPM. There will be some code changes needed to make it work properly.

Radio options

The current code of AutoQuad will enable the use of Spektrum satellite(s), S-bus receivers & PPM support. PPM support is considered experimental in firmware 6.6 r22 and above. With Spektrum you can attach one (1) satellite on the main board but if you need extra long range or diversity you can use a spektrum diversity board that will enable the use up to 4 satellites.

Read more **about radio connections and special details.**

Setting the receiver type will be done in the Ground control station:
For that you need to connect the AutoQuad + FTDI to the computer.



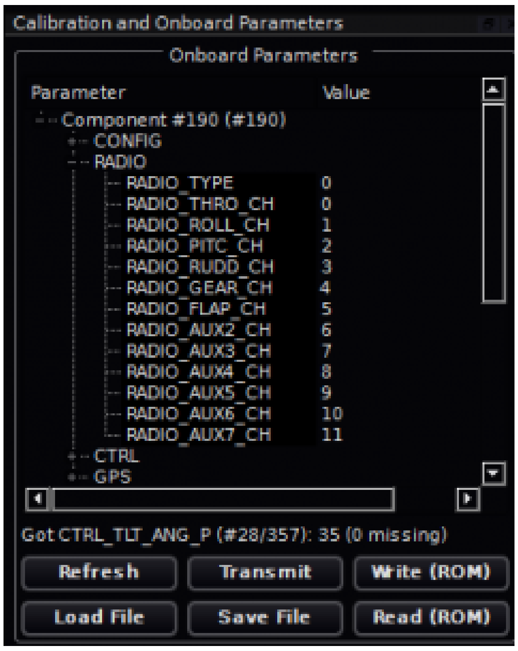
Select the correct interface and click ‘*transmit to AQ*‘. **Best to reset the AutoQuad flightcontroller after setting a new radio interface**.

You can select several Spektrum DSM2/DSMX compatible radios, Futaba S-bus, (C)PPM and M-Link (Multiplex).

- Spektrum 11 bit (JR 9303, 9503, DXM, Spektrum DX7i, DX7s and later)
- Spektrum 10 bit (Spektrum DX7, DX6)
- Futaba S-bus (needs signal inverter)
- (C)PPM, FrSky, Hitec and others via multiprotocol receivers (uses the motor channel #14)
- SUMD for Graupner HOTT radios (needs firmware r128 or higher and signal inverter. details in this thread)
- M-Link (Multiplex)

Radio channel mapping

New in firmware 6.6r20 is radio channel mapping. With this option you can set the appropriate radio channels to AutoQuad radio map. This option will for example allow to have AutoQuad's flap channel to be on channel 9. It is available in the all parameter screen in QGCS under the radio section.



Parameter	Value
Component #190 (#190)	
CONFIG	
RADIO	
RADIO_TYPE	0
RADIO_THRO_CH	0
RADIO_ROLL_CH	1
RADIO_PITC_CH	2
RADIO_RUDD_CH	3
RADIO_GEAR_CH	4
RADIO_FLAP_CH	5
RADIO_AUX2_CH	6
RADIO_AUX3_CH	7
RADIO_AUX4_CH	8
RADIO_AUX5_CH	9
RADIO_AUX6_CH	10
RADIO_AUX7_CH	11
CTRL	
GPS	

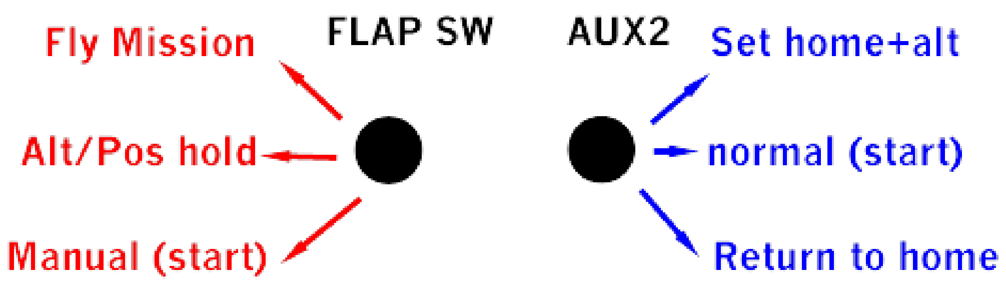
Got CTRL_TLT_ANG_P (#28/357): 35 (0 missing)

Just click on the number next to the channel name and enter the radio channel you want. As always use the *transmit* and *write (rom)* button for transmitting and storing the setting into AutoQuad's flash. Reset the AutoQuad after writing to Rom.

Flight radio switches

Flight radio switches

AutoQuad has several flight modes. For selecting them we use two channels: the **flap** (aux1) channel and **aux 2**.



Start flying with the **Flap** switch in manual (down) and **Aux 2** in normal (center) position.
Aux 2 is normally always in center.

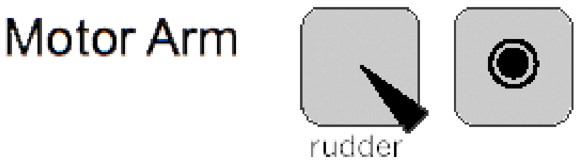
Action	**Flap**	**Aux 2**
Start flying	down	center
Set home position & altitude	down or center	1 sec up, then center
Return to home	down or center	down
Alt or Position hold	center	center
Fly waypoint mission	up	center
Abort RTL or Mission	down	center

Arming / disarming

Just like the majority of available flight controllers, Autoquad is not armed when applying power.
Arming / disarming is only possible when the switches are set for manual/normal mode!
(FLAP/AUX1=low/manual, AUX2=mid/normal)

For **arming** the motor, hold throttle down and yaw (rudder) right for 3 seconds.
For **dis-arming** the motor, hold throttle down and yaw (rudder) left for 3 seconds.

mode 2 layout



Part 2 - Calibration & Calculation
AutoQuad Calibrations

Calibration is the Key!

Introduction

One of the most important reasons the AQ flies so well is that it has the ability to know what orientation it is in, what speed it is traveling at, and what position it has in 3D space. The only way it can know this precisely, is if the various sensors are calibrated very well.

For the AutoQuad we perform two types of calibration routines: Static and Dynamic.

Why the calibration?

Every sensor is different, even from the same batch. Imagine what you can accomplish, if you take all those specific variables into consideration in your flight algorithms? That’s the AutoQuad approach.

“Dry” calibration run

It’s a good idea to just perform a log run before you put your FC into the freezer. That way you can find out from the start whether the sensors act fine or if there are conditions that might disturb the static calibration later.
Just let the AQ6 FC log for a while (1-2 hours) to see if there are any magnetic fields, power lines, electrical devices that might spoil the act later. You will also have a chance to see if any vibrations happened during the run or if the SD card isn’t logging properly.
So you’ll find defective or strange acting units at an early stage.

Static calibration

In the static procedure, the Inertial Measurement Unit (IMU) is temperature-compensated by calculating the offset and bias values over a temperature range. We basically first freeze the board, and and let it heat up slowly, logging the changes in the sensors while heating up. During logging, the board must be placed absolutely still, hence the term “static calibration.”

Dynamic calibration

In the dynamic part, the entire magnetic profile of the craft is taken into account by performing a “calibration dance” with your fully-assembled and powered craft while logging the changes in the sensors.

Calculation

The data from the static and dynamic procedures is then brought back for analysis using the ground station software. For this part you need a fast computer with multiple CPUs/cores (i7 or higher is recommended). Depending on how much CPU power you have, and the quality of your log files, it can take 24 hours or more to complete the calculations.

Since the dynamic calibration takes into account the magnetic profile of the complete multi-rotor craft, it needs to be fully assembled, with all peripherals powered up and running, when performing the dynamic part. This also means that if you change the craft or move the AQ board to another frame, you will have to redo the dynamic calibration.

Warning!

If done properly, the procedure will yield a very precise IMU and that is really the key to \the precise performance of AQ. **But be warned:** You need patience, a steady hand and a good computer to complete the calibration properly. If you do it hastily or incorrectly, the result might be disappointing, or only “adequate” instead of “great.”

So please make sure you read the calibration documentation and watch the videos carefully before you proceed. **Autoquad is highly advanced and it is NOT an “out the box” controller**, but if you have patience and like to tinker, there is ample reward in the end.

We offer **no guarantees** besides that the Hardware is working as it should. The final result is up to you — your skills and patience! We will do our best to help you on the forum, but know that experienced guys have tried and failed a few times before getting it right!

When to recalibrate

- **Static & Dynamic Recalibration:**
    1. When an AutoQuad board is repaired.
    2. Any change of the IMU sensors and/or the MCU.
    3. After a serious crash. The high G impact might have influenced the proof masses of the sensors in some way.
- **Dynamic recalibration:**
    1. Change of frame.
    2. Change in layout, like rewiring power cables.
    3. Change in ESCs or their position.
    4. Installation of large iron masses in the proximity of the flight controller.

Calibration Steps Outline

A note about AutoQuad leveling

AutoQuad needs no leveling step as the calibration procedure (static AND dynamic) leaves all sensors orthogonally aligned to each other and to the bottom of the acc sensor. As long as the acc chip sits flat on the board and the board is mounted square with the frame, there should be very little deviation from level. If there is a small amount, it can usually be trimmed out with a few clicks on your radio.

Static Data Gathering

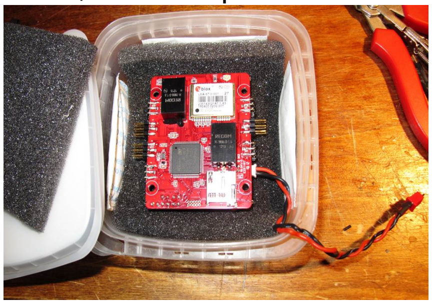
Static Data Gathering

Introduction

The purpose of this procedure is to gather data about how the sensor values change with temperature. Basically it's about freezing the board and letting it thaw slowly under its own heat while kept absolutely still (static) and logging the changes in the individual sensors. This data is then used to calculate calibration values for a particular board.

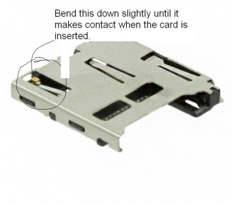
Logging for calibration is a 'standalone' action, you do not need to have a FDTI-USB connection active. Only the uSD card is needed. The uSD card will need to be read using an SD card reader on a desktop or laptop computer.

Prerequisites

- A micro SD card from the confirmed cards list . It is a good idea to scan the card for errors before using it. On Windows, insert the card into a reader, R-click on the drive, then Properties -> Tools -> Check for errors.
- 
  An airtight and insulated container, just big enough to hold the AutoQuad board. Plastic food containers work well, if you stuff them with some dry insulation material (foam works well).
    - If you live in a humid environment, you may also need a Ziplock bag just big enough to hold the AutoQuad board. Silica desiccant packages can also help, as can satchels of rice in porous containers (like cheesecloth or stocking).
    - It can be useful to see the LED lights on the AQ board during the logging process, so a clear container with a clear view of the lights is recommended. This way you can ensure the board is really logging after you plug it in.
    - Do NOT use anti-static silver/metallic bags as these have been determined to cause problems with magnetometer readings.
- 9-18V power source (3S lipo is perfect).
- A long power wire for the AQ board.
- A freezer that can reach at least -15 Celsius.
- A location where the board can sit undisturbed from vibrations. You need to be picky about this — even people walking by on the same floor or cars going by can create noise in the accelerometers and screw up your result. A soft pillow or

mattress can be good insulation against vibrational noise. Buildings themselves vibrate less the lower you are. So the cellar might be a perfect room for the thawing.

- Pick a place where there are no running appliances, electrical devices or A/C power lines nearby that can disturb the magnetometers while logging. Constant magnetic influences like iron or steel objects are not a problem but fields that change over time should be avoided. So don't use your bathroom floor with electrical heating, for instance.


Bend this down slightly until it makes contact when the card is inserted.

Note: some users reported difficulties with their SD card not logging correctly. Some boards of the early beta batch need a little "mod" by bending a latch down slightly. Click on the image at right.

Procedure overview

1. Connect the power wire to AQ board (don't connect to battery). Optionally, insert the uSD card now (this will make it easier to remove the board from the freezer w/out introducing moisture, but on the other hand it may not be very good for your card).
2. Place the board inside an open Ziploc bag (optional for humid environments).
3. Put the board/bag into the insulated container with the lid open and put it into freezer (as deep or as far back as you can get — it's colder there).
4. Freeze it down to -15 to -25 degrees (this takes at least one hour, depending on your freezer).
5. After freezing, quickly close the Ziploc bag (if using one) and container while still inside the freezer. If you have not inserted the uSD card before freezing in step 1, do so before sealing up the board in the bag/container (do it quickly, inside the freezer, and don't breathe on it). Just the power lead should stick out so you can plug it into the battery. Make it as airtight as possible.
6. Place the container in a undisturbed area and plug in the battery (be sure to observe correct polarity!). This way the board will slowly heat the compartment by the heat emitted from the board itself. If you can see into your container without disturbing it, make sure the red LED is on steadily, and the green LED is blinking rapidly (this means the board is on and logging).
7. In approximately 60 minutes the board will be heated to around +50C or 120F. For the first run, it is better to wait longer. After that you can check the log file and see how long you need to wait for a full thaw. Depending on the size of the box you might need up to 90 minutes.
8. When the 60-90 minutes have passed, disconnect battery, taking care not to disturb the board.

For a good result you need to do this 3 times in total with the board in different orientations to expose the acceleration sensors and magnetometers to different angles of the Earth's gravity and magnetic fields. The orientations do not have to be exact but a good sequence is:

- First session: Orient the board so a corner is pointing into the earth.
- Second session: Orient the board so the opposite corner is pointing into the earth
- Third session : Orient the board level.

The goal is to capture as much temperature range as possible, aiming to get at least 10 degrees Celsius above and below the expected flying temperature range. An ideal range is something like -25 to 60 degrees, but that can be hard to achieve, and not needed in most cases.

After your first freezing/thawing session, it is a good idea to check the generated log file (and make a backup of it). This will give you a good idea of your logging environment (for example if something is disturbing the board), show your temperature range, your thawing time, and so on. All this can be useful to know before doing the next sessions. If nothing else, at least make sure you are actually getting a log generated at all.

You need to check the files for errors, noise, and condensation glitches. Refer to the ” analyze data ” section for details. It is essential to check all your logs before proceeding to the calculations (in a later step). It is also a good idea to check the logs before mounting the board in your frame and proceeding with the dynamic data gathering. It is much easier to re-freeze the board now than after it is all mounted up!

Video showing the static calibration procedure

Errors to watch for

The most common problem during the static data gathering is condensation inside the box as it’s thawing. This can form tiny drops of water on the board, which can make small short circuits on the analog sensors and make your data unusable. Refer to the analyze calibration data section for details on log file analysis.

The best way to prevent condensation, is to place the board inside a small Ziplock bag before putting it in the container. This will minimize the amount of air around the board that can condensate. Freeze with Ziplock bag and container open, and close them inside the freezer just before bringing them out. You can also add desiccants like small silica satchels or a little bag of rice made from a piece of nylon stocking.

Now if you have gathered sufficiently good static data, and have analyzed it to your satisfaction, proceed to the Dynamic calibration steps....

Dynamic Data Gathering

Dynamic Data Gathering

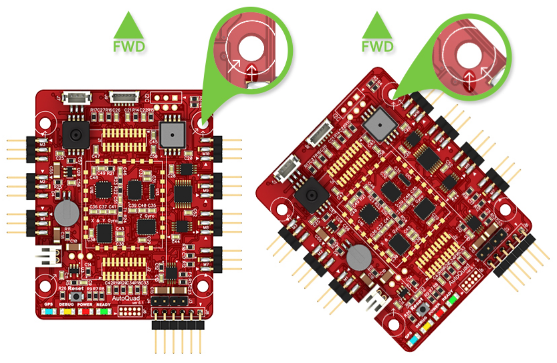
Now it’s getting difficult (also to explain) but read carefully and watch the videos…

In this process we are aiming to expose the magnetometers to magnetic north in all orientations. We do that by a series of coordinated moves, affectionately known as the dance “Calibso”.

The Calibso must be performed with the board installed in the airframe in a completely flight ready condition except for propellors. That means that all peripherals near the flight controller (cameras, telemetry, etc) are mounted, powered up, and active (broadcasting, recording, etc). This is important as all soft and hard iron effects in the frame must be taken into consideration.

When done correctly the complete magnetic profile of the entire frame is recorded. This is used later in a offline calculation that creates a complete profile for the IMU.

When mounting the board, pay attention to the direction arrows printed near the mounting holes. One of the sets needs to be pointing towards the front of your frame. For the Calibso it doesn’t matter which board orientation you use. You don’t need to configure the firmware before you perform the dance.



Introducing the Calibso

The “Calibso” is a specific sequence of rotations (hence the ‘dance’). Consider that the frame has 6 “faces” like a cube:

- Top face
- Bottom face
- Right face
- Left face
- Front face
- Rear (back) face

Start with any face pointing up (opposite of gravity's pull) and then slowly rotate 90 degrees to an adjacent face and back up. Repeat until all 4 adjacant faces have been covered. (Front, back, right, left).

Then rotate craft slowly to bring a new face up and repeat the 4 rotations to the adjacent faces.

While this is going on, slowly rotate your body on your own yaw axis. Repeat until all 6 faces of the craft have been covered. Each sequence will take around 1 minute to perform.

Complete Calibso dance sequence

IMPORTANT**:** Go outside, away from sources of electromagnetic noise coming from power lines, household appliances and electronic devices. There should be no large steel, iron or other ferris objects near you while you do this. Remove iron, steel or other metallic or electronic items from yourself (belt buckles, pocketknife, cellphone, watch, jewelry, etc). You can take our word for it — it matters!

You can also get step-by-step instructions here. Download a PDF with illustrated instructions here**.**

1. Place your craft on a level surface, and power it up.
2. You can wait for GPS 3D fix, but its not needed. However it will make sure that your log files are time stamped.
3. When board is booted (Green LED blinks 2 times per second) and you have a GPS fix (blue LED on solid), insert the card (green LED blinks 5 times a second when logging) and let the craft sit still for at least 10-15 seconds.
4. Pick up your craft GENTLY and begin the dance we call "Calibso" (If you don't know what Calibso is, go back and watch the videos with the dancing lessons again).
5. Start with any face pointing up (opposite of gravity's pull) and then slowly rotate 90 degrees to an adjacent face and back up. Repeat until all 4 adjacent faces have been covered. (Front, back, right, left). While this is going on, slowly spin your body in place (yaw) like a slow waltz (but without swaying!).
6. Then rotate craft slowly to bring a new face up and repeat the 4 rotations to the adjacent faces, still turning slowly on your own yaw axis.
7. Repeat until all 6 faces of the craft have been covered and return your craft gently to a level surface and let it log for at least another 10-15 seconds.
8. Unplug power and remove the uSD card.

The whole sequence should take at least 5 minutes – time yourself and if it's shorter than 5 mins, you are going too fast.

Hints:

- *You only need one good dynamic file to complete the calculations. But we recommend that you do a few more while you are at it anyway. This will give you more data to choose from in the calibration calculations.*
- *Remember that practice & patience makes it perfect and don't be*

discouraged if you don’t succeed the first time. Like all good courtships, learning to Calibso takes time..;-)

- *Always move your craft gently around at all times while dancing. Remember that it is not jitterbug or jazz ballet, and your “partner” does not like to be thrown around like a cheap date* 😉

The micro SD card will now contain a number of log files that can be used for the calculations. You should now have at least 3 large log files containing the static data (around 200Mb) and 1 or more smaller log files containing the just performed dynamic data (around 20Mb). You could rename them to S_sequencenumber and D_sequencenumber, but that is not mandatory.

Just like the static files, you should check the files in the log viewer and possibly also clip away the noise in the beginning and end of your files. More on this in the “analyze data” section.

Dynamic Calibration Movie, part 1

Dynamic Calibration Movie, part 2 (complete sequence)

Now proceed to verify your recordings.

AutoQuad Calculations

AutoQuad Calculations

So, now that all calibrations are recorded, it’s time to calculate the values from them. Reserve some time for it, it’s going to take some. A lot of screen captures are included in this guide. In the following guides we will perform the calculations on the calibration data.

Prerequisites

- **A 64-Bit Intel-based computer, the faster, with multiple cores, the better. i7 is recommended.**
- **An installed AutoQuad** ground control station**.**
- **Time! The calculations take everything they can from the computer but may still take 24h or more to complete.**
- **The log files from the static and dynamic calibrations.**

Calibration Software

The calibration software consists of two programs, Cal and Sim3. They are included in the QGroundControl AQ widget with a simple interface to choose the files and execute the calculations.

How it works

The software effectively tries to converge on a solution by adding random noise to the data set testing randomly selected time steps in a random order. There is actually no perfect solution possible, but instead an almost infinite number of good solutions for which it tries to get close to. Each run will be influenced by the random nature and approach the best solutions from a different “angle”.

If let run long enough, both programs should usually converge to an acceptable solution. However, by default the run time for cal has been set rather low for expediency sake. You can compensate for this by running individual step multiple times, using the prior solution as a starting point for the next run to improve the result.

How to determine a good result (brief version)

The main quality of the solution is expressed as Median Absolute Error – MAE. The lower this number gets, the better, but there are other stuff to look out for, so read on!

MAE is not an absolute value that can be compared from board to board. MAE is purely a tool to see if the calibrations are proceeding as expected. What you are looking for is not a specific value, but instead you are looking for a nice steady drop of MAE as the calculations proceed.

For the first 3 “cal” steps, the program will finish on its own and produce a result. The run times can be between a few minutes and up to 15-20 minutes depending on how many iterations (loops) the cal program runs before deciding on an acceptable solution. This can vary from run to run even with the same data.

For the later Sim3 runs, there is a bit more to look out for, and the runtimes are considerably longer (many hours) for each run. The Sim3 simulation will run until you stop it, so you need to look out for a few things in the beginning of each run and make adjustments if needed and keep an eye on the MAE as it runs. When the MAE is not dropping significantly on each loop it is time to move on to the next step.

There are four sim3 steps and total sim3 runtimes can be as high as 24 hours before an acceptable result is produced. But it can also happen much faster. Ultimately the quality of the final solution depends highly on the quality of the logfiles you created and how much CPU time you can throw at it.

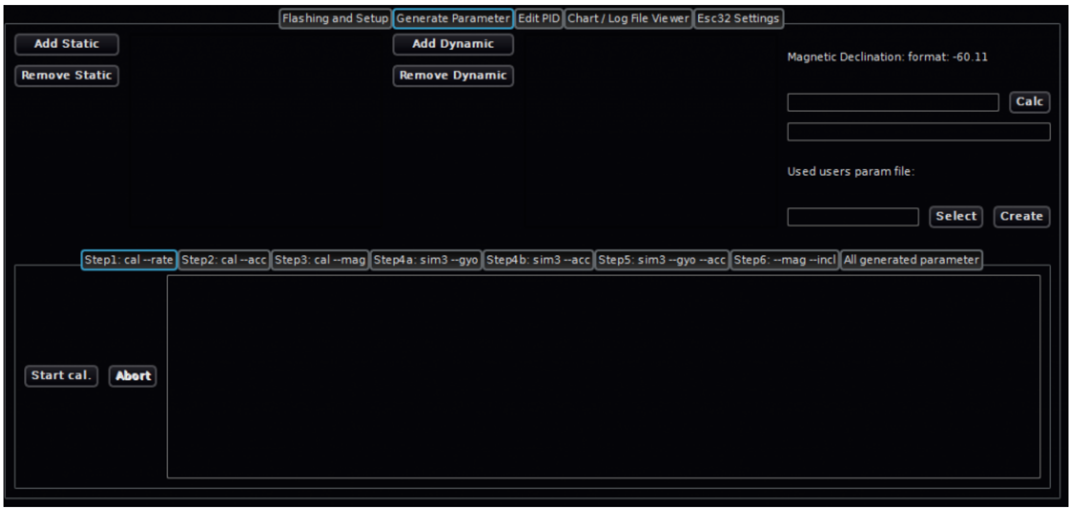
The sim3 program will suck all the CPU power you can throw at it and still take a long time to come up with a solution. 4 processor cores or more is the recommended minumum, but you may get away with less if you have enough time 😉

Step 1: Prepare & Run the calculations
Step 2: Upload the calcuations to the AutoQuad flightcontroller.

Preparing To Run The Calculations

Preparing To Run The Calculations

1: Load the log files.



Open the Ground Control Station, make sure the AutoQuad main widget is on the screen, and select *'Generate Parameter.'*

Now select '*Add Static* ' and add your static log file(s) from the freezer calibration, then '*Add Dynamic* ' for the dynamic calibration log files.

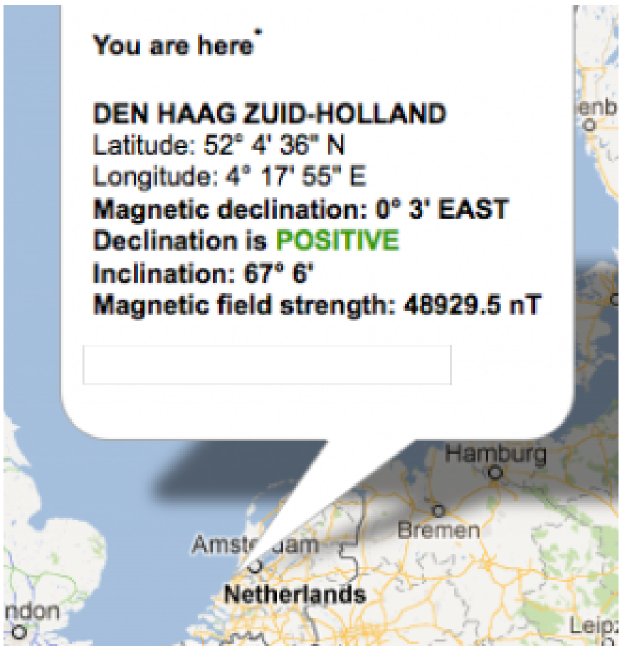
2: Retrieve magnetic inclination for your location

The next step will require an estimate of your magnetic inclination for your geographic location. Inclination is also called magnetic dip. Is it the amount of degrees a compass needle will point either upwards or downwards into the earth if the compass is turned to a vertical orientation.

There is a page of documentation dedicated to magnetic declination and inclination. Please make a step to that page for an explanation. IT IS IMPORTANT YOU GET THIS RIGHT!

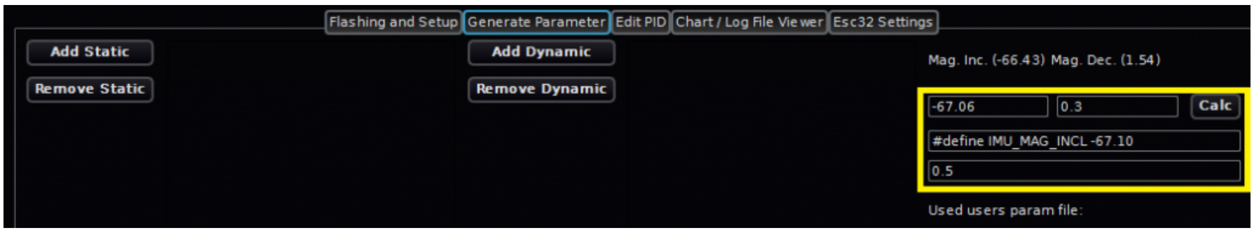
IMPORTANT NOTE: For mathematical reasons, AutoQuad has the inclination reversed. If you fly in the northern hemisphere, then inclination will be negative, and for southern hemisphere, inclination will be positive — this is the opposite of what the online tool shows. You should always make sure that the define produced has the polarity reversed (add or remove the "-" in front of the number in the define).

The declination and inclination for your location can be retrieved from the site


You are here

DEN HAAG ZUID-HOLLAND
Latitude: 52° 4' 36" N
Longitude: 4° 17' 55" E
Magnetic declination: 0° 3' EAST
Declination is POSITIVE
Inclination: 67° 6'
Magnetic field strength: 48929.5 nT

Note that this tool shows inclination as degrees/minutes. AutoQuad operates with decimal degrees.



Mag. Inc. (-66.43) Mag. Dec. (1.54)

-67.06 | 0.3

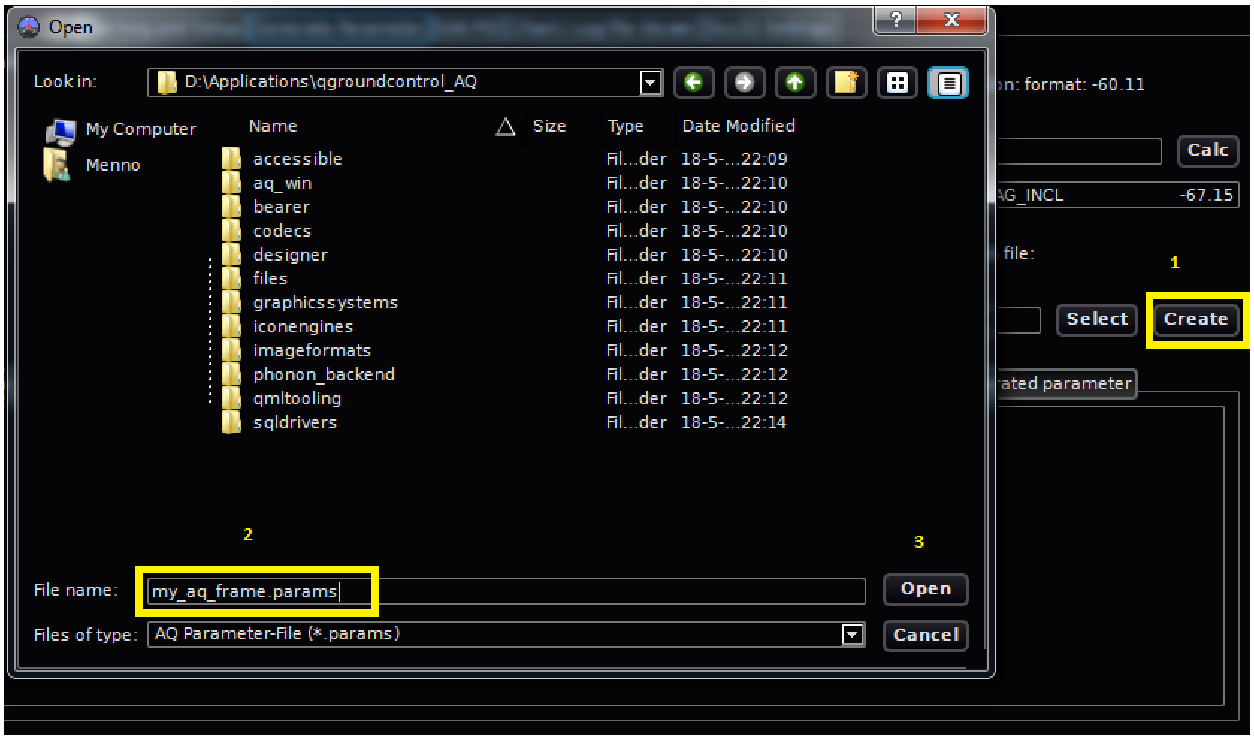
#define IMU_MAG_INCL -67.10

0.5

Used users param file:

In this example the inclination is 67.6, enter that in the GSC with a leading 0 (zero) when the minutes are just one digit. So 67.06. As explained on the magnetic inclination/declination page, AutoQuad need to use the reversed variant, so in this example the final number will be ***-67.06***. Just enter the retrieved inclination in the first field and the magnetic declination in the second field and press *'calc.'*
The widget will convert the inclination to decimal degrees and provide a #define statement for use in the param file later on.

3: Create your parameters file.

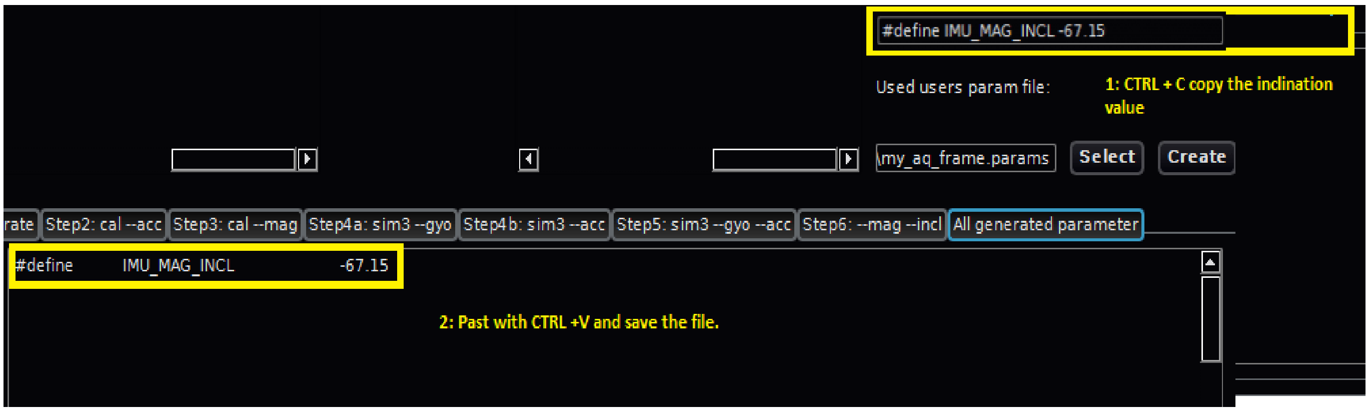


Name	Size	Type	Date Modified
accessible		Fil...der	18-5-...22:09
aq_win		Fil...der	18-5-...22:10
bearer		Fil...der	18-5-...22:10
codecs		Fil...der	18-5-...22:10
designer		Fil...der	18-5-...22:10
files		Fil...der	18-5-...22:11
graphicssystems		Fil...der	18-5-...22:11
iconengines		Fil...der	18-5-...22:11
imageformats		Fil...der	18-5-...22:12
phonon_backend		Fil...der	18-5-...22:12
qmltooling		Fil...der	18-5-...22:12
sqldrivers		Fil...der	18-5-...22:14

File name: my_aq_frame.params
Files of type: AQ Parameter-File (*.params)

All calculations should be saved to a file. You can create a new param file by clicking on the '*Create*' button. Select a appropriate name for it and save it by selecting ' *open.*'

Switch to the '*All generated parameter*' tab. This window contains the contents of your param file, which should currently be empty. Now select and copy (CTRL + C) the '#define IMU_MAG_INCL xx.xxxx' you just created in the previous step, and paste it (CTRL + V) into the blank part of this window. The result should look something like this:

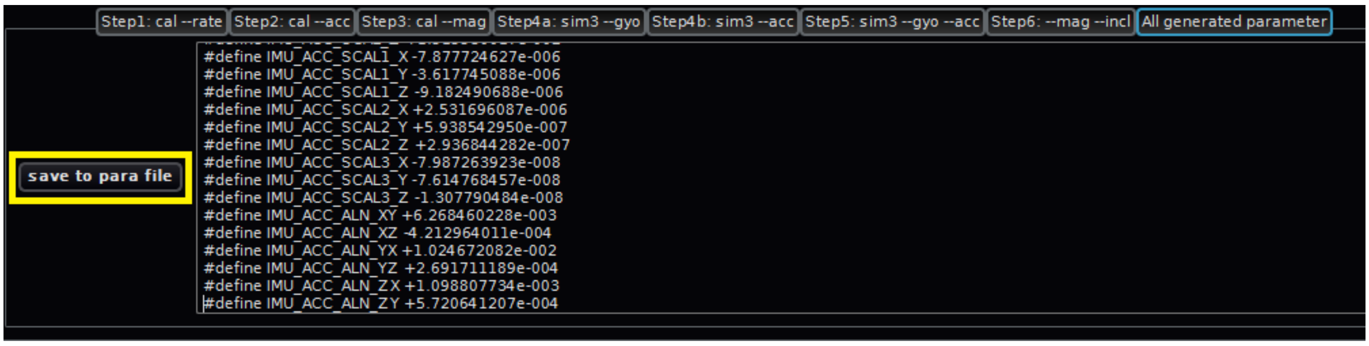


```
#define IMU_MAG_INCL -67.15
```

Used users param file: my_aq_frame.params

```
#define        IMU_MAG_INCL              -67.15
```

Save your new parameter file by clicking the 'save to para file' button to the left of the window.



```
#define IMU_ACC_SCAL1_X -7.877724627e-006
#define IMU_ACC_SCAL1_Y -3.617745088e-006
#define IMU_ACC_SCAL1_Z -9.182490688e-006
#define IMU_ACC_SCAL2_X +2.531696087e-006
#define IMU_ACC_SCAL2_Y +5.938542950e-007
#define IMU_ACC_SCAL2_Z +2.936844282e-007
#define IMU_ACC_SCAL3_X -7.987263923e-008
#define IMU_ACC_SCAL3_Y -7.614768457e-008
#define IMU_ACC_SCAL3_Z -1.307790484e-008
#define IMU_ACC_ALN_XY +6.268460228e-003
#define IMU_ACC_ALN_XZ -4.212964011e-004
#define IMU_ACC_ALN_YX +1.024672082e-002
#define IMU_ACC_ALN_YZ +2.691711189e-004
#define IMU_ACC_ALN_ZX +1.098807734e-003
#define IMU_ACC_ALN_ZY +5.720641207e-004
```

You are now ready for the actual calculations. Go to the next page to proceed.

Run the calculations - Steps 1-3

Running the calculations – Steps 1 through 3

Note: For every calculation step: copy the output (select all + CTRL + C) and paste (CTRL + V) in the '*All generated parameter*' window.
Every time you paste data to the '*All generated parameter*' screen, please click on the '*save to para file*' button. The previously created param file will then be updated. Almost all calculation steps use the data created by the previous steps.

Tip: to compare results of several runs of the same step, which is a good idea, use a plain text editor (eg. Notepad) to keep several versions. Simply paste the results into the editor and use the best ones in the generated parameters file before proceeding to the next step.

Tip 2: When pasting values to your param file, you can insert comments by preceding them with two slashes on the line. This can be useful for keeping track of your calculation process, when requesting assistance, and for historical purposes. For example, here are some notes included in a param file from step 4b:

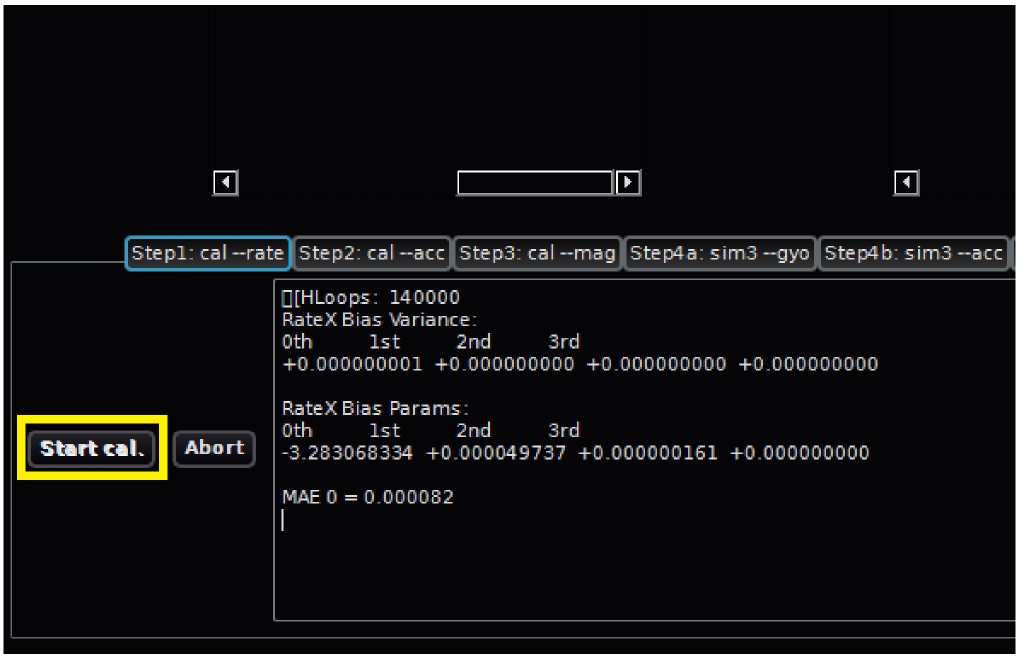
```
// sim3 –acc

// 0.029386 0.008872 0.029981 0.000000 0.000000  = 0.068239

// Loops:  1,394
#define IMU_ACC_BIAS_X        -1.647290262642    //  0.000000000729 -0.000000003981
…
```

If you run into problems with any of the following steps, scroll down to the troubleshooting section, below.

Step 1: Calculation of the gyro (rate) bias.



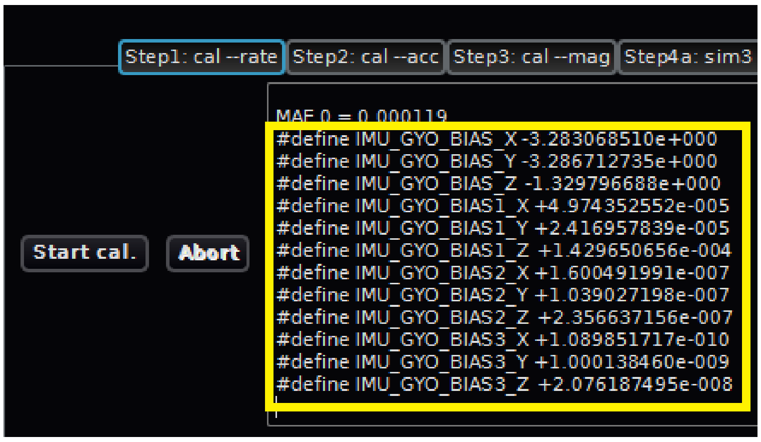
```
[HLoops: 140000
RateX Bias Variance:
0th        1st        2nd        3rd
+0.000000001 +0.000000000 +0.000000000 +0.000000000

RateX Bias Params:
0th        1st        2nd        3rd
-3.283068334 +0.000049737 +0.000000161 +0.000000000

MAE 0 = 0.000082
```

This step uses only the static log files as input.

Select the ‘*Step1: cal –rate*‘ tab and hit ‘*Start cal.*‘ button. The output screen will now show the gyro bias calculations. It will run for a couple of minutes.



```
MAE 0 = 0.000119
#define IMU_GYO_BIAS_X -3.283068510e+000
#define IMU_GYO_BIAS_Y -3.286712735e+000
#define IMU_GYO_BIAS_Z -1.329796688e+000
#define IMU_GYO_BIAS1_X +4.974352552e-005
#define IMU_GYO_BIAS1_Y +2.416957839e-005
#define IMU_GYO_BIAS1_Z +1.429650656e-004
#define IMU_GYO_BIAS2_X +1.600491991e-007
#define IMU_GYO_BIAS2_Y +1.039027198e-007
#define IMU_GYO_BIAS2_Z +2.356637156e-007
#define IMU_GYO_BIAS3_X +1.089851717e-010
#define IMU_GYO_BIAS3_Y +1.000138460e-009
#define IMU_GYO_BIAS3_Z +2.076187495e-008
```

Once finished the output will look similar to this. This output is needed for the param file you created. The MAE (Median Absolute Error) is an expression of the quality of the solution calculated. The aim is to get the lowest possible MAE, so it can be a good idea to re-run the cal steps a few times and pick the set of defines that has the lowest MAE.

Note that the output will show one MAE value for each static log file used (only one was used for the above screenshots). Pick a run that has the lowest average MAE across the files. Often the MAE changes will be very minor from run to run (+/- Tools -> Check for errors).

Probably the best thing to do if you are having trouble with your calculations or doubts about your data, is to post your issue and your log graphs on the forums. Here is an example post with all the relevant graphs provided.

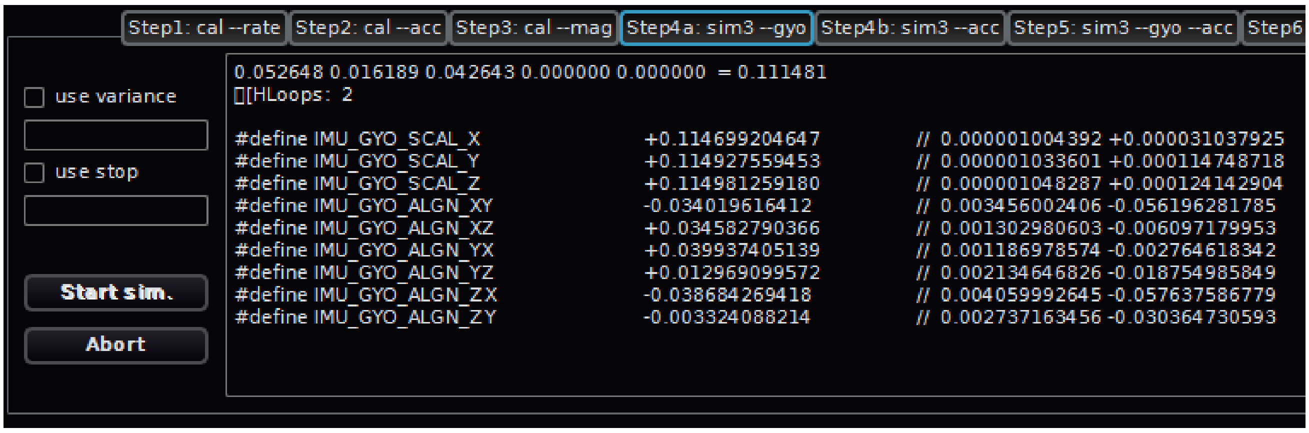
Instructional video :: Calculations step 1-3 (preliminary German version)

[Link to embedded object]

Run the calculations - Steps 4-6

Running the calculations – Steps 4 through 6

Steps 4a, 4b, and 5: gyro and accelerometer estimation simulation



```
0.052648 0.016189 0.042643 0.000000 0.000000  = 0.111481
[HLoops: 2

#define IMU_GYO_SCAL_X           +0.114699204647       // 0.000001004392 +0.000031037925
#define IMU_GYO_SCAL_Y           +0.114927559453       // 0.000001033601 +0.000114748718
#define IMU_GYO_SCAL_Z           +0.114981259180       // 0.000001048287 +0.000124142904
#define IMU_GYO_ALGN_XY          -0.034019616412       // 0.003456002406 -0.056196281785
#define IMU_GYO_ALGN_XZ          +0.034582790366       // 0.001302980603 -0.006097179953
#define IMU_GYO_ALGN_YX          +0.039937405139       // 0.001186978574 -0.002764618342
#define IMU_GYO_ALGN_YZ          +0.012969099572       // 0.002134646826 -0.018754985849
#define IMU_GYO_ALGN_ZX          -0.038684269418       // 0.004059992645 -0.057637586779
#define IMU_GYO_ALGN_ZY          -0.003324088214       // 0.002737163456 -0.030364730593
```

These 3 steps are very similar to each other. 4a and 4b calculate the sensors individually, and step 5 further refines this data by leveraging the calculations against each other. These calculations are **very** CPU intensive. The simulation will never end.

In the top left corner, the number of calculation loops is displayed. The MAE is the last number on the top line, after the '=' sign. You can abort the simulation with the '*Abort*' button when the MAE is not dropping significantly anymore on each loop. If you're patient, or have good data and/or good luck, wait 10-50 loops with no MAE drop. 500-2000 loops for each step are not uncommon, possibly more on Step 5. Each step can take several hours to complete. On the other hand, with a good set of data, the MAE may stop dropping much quicker, perhaps after only 50 – 200 loops, so that is OK also, as long as the numbers look good.

You can stop any of these steps at any time, save the generated parameters to your params file, and then start the step again. The calculations will resume where they left off.

Checking the calculations

You should be looking for a slow drop of the MAE (last number on the top line in the calculation window). If the MAE drops very fast or starts to rise or jump up and down, this may indicate a problem with your data, or some of it. It may help to simply start the step again (the simulation algorithm involves a certain amount of "luck" to produce a good solution).

Another thing to watch are the numbers being generated (the 1st column of numbers in the output). Small changes on each run are expected, but if you start seeing big swings, for example from -0.00005 to +0.00005 in one loop, this may indicate a problem. Keep an eye on these swings and if they get worse, abort the run and try again.

If you look at the results produced, you will see 3 columns of numbers. The first column is the calibration values produced so far. The 2nd column is the total change that was made in this run, and the last set of numbers on each line is the magnitude of change to each calculated value that was made in the last loop.

Let the simulation run until the MAE is not dropping significantly and the magnitude on each loop is low.

After each step, don't forget to add the generated data to the '*All generated parameter'* window and save it.

If you have problems...

If the results aren't looking correct, there are a couple of things to try. The simplest is to just try running the step again. As mentioned, the calculations depend on a certain amount of luck so come up with a good result.

You may also need to change the variance parameter (to the left of the calculations window, above the '*Start sim*' and '*abort*' buttons. Default variance for Sim3 is 1e-6 (0.000001) so try 1e-7 or lower. Check the '*use variance*' box and enter a number in the field below, then hit '*Start sim*.'

In general if you have to lower variance, it points to problems with your data. Sometimes it can be good to run a few dozen/hundred loops at a low variance to converge the data and save the results into your params file. Then run the step again at a higher (or default) variance. The more you lower variance the longer it will take to arrive at a good solution.

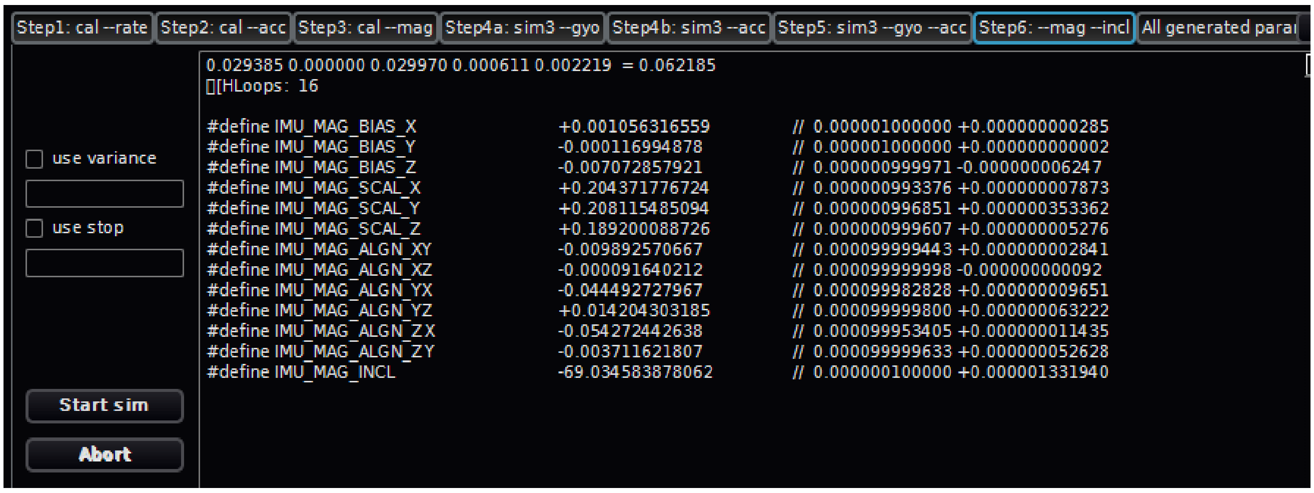
If you continue to experience problems running any of these 3 steps, then it may be time to re-examine your log files or create new ones. The dynamic log in particular (from your dance) may take several tries to perfect, and it is harder to spot anomalies in that data.

If you continue having trouble with your calculations or doubts about your data, post your issue and your log graphs on the forums. Here is an example post with all the relevant graphs provided.

Before Step 6...

Before proceeding to step 6, you can go back to Step 3 and see if you can get your MAE even lower for those calculations. Use the same procedure as outlined for Step 3 until the MAE number stops falling. Keep replacing the IMU_MAG values in your generated params file with the new ones. Once the MAE stop dropping in Step 3, move on to Step 6. You're almost there!

Step 6: magnetic inclination estimation simulation



```
0.029385 0.000000 0.029970 0.000611 0.002219  = 0.062185
[][HLoops: 16

#define IMU_MAG_BIAS_X          +0.001056316559     // 0.000001000000 +0.000000000285
#define IMU_MAG_BIAS_Y          -0.000116994878     // 0.000001000000 +0.000000000002
#define IMU_MAG_BIAS_Z          -0.007072857921     // 0.000000999971 -0.000000006247
#define IMU_MAG_SCAL_X          +0.204371776724     // 0.000000993376 +0.000000007873
#define IMU_MAG_SCAL_Y          +0.208115485094     // 0.000000996851 +0.000000353362
#define IMU_MAG_SCAL_Z          +0.189200088726     // 0.000000999607 +0.000000005276
#define IMU_MAG_ALGN_XY         -0.009892570667     // 0.000099999443 +0.000000002841
#define IMU_MAG_ALGN_XZ         -0.000091640212     // 0.000099999998 -0.000000000092
#define IMU_MAG_ALGN_YX         -0.044492727967     // 0.000099982828 +0.000000009651
#define IMU_MAG_ALGN_YZ         +0.014204303185     // 0.000099999800 +0.000000063222
#define IMU_MAG_ALGN_ZX         -0.054272442638     // 0.000099953405 +0.000000011435
#define IMU_MAG_ALGN_ZY         -0.003711621807     // 0.000099999633 +0.000000052628
#define IMU_MAG_INCL            -69.034583878062    // 0.000000100000 +0.000001331940
```

This last step finalizes your magnetometer calibrations. If the mags are estimated separately, the software can optimize the process and perform a lot of iterations in a short period of time.

The software will also estimate the magnetic inclination. Don't worry if this comes up a few degrees off from what is estimated in the online inclination tool. Its most likely due to local geomagnetic variances, like iron deposits in the Earth´s crust.

Like the previous 3 steps, the simulation never ends. Abort it when the MAE stops dropping (MAE is the last number in the top line of output, after the '=' sign). It may not drop for 100 loops and then go down a little more. Give it some time. 30.000-40.000 loops are not uncommon (they run quickly!). Again, you can abort the run, save the data to your param file, and re-start the step again — it will, essentially, continue where it left off.

When you're happy with the result, copy/paste the defines into your params file and save it. A complete param file specific for your AQ board is now compiled. Congratulations! This would be a good time to make a backup of it.

This file will now be used to feed the firmware with the needed information. Next step, uploading the parameter file to AutoQuad

Instructional video :: Calculations step 4-6 (preliminary German version)

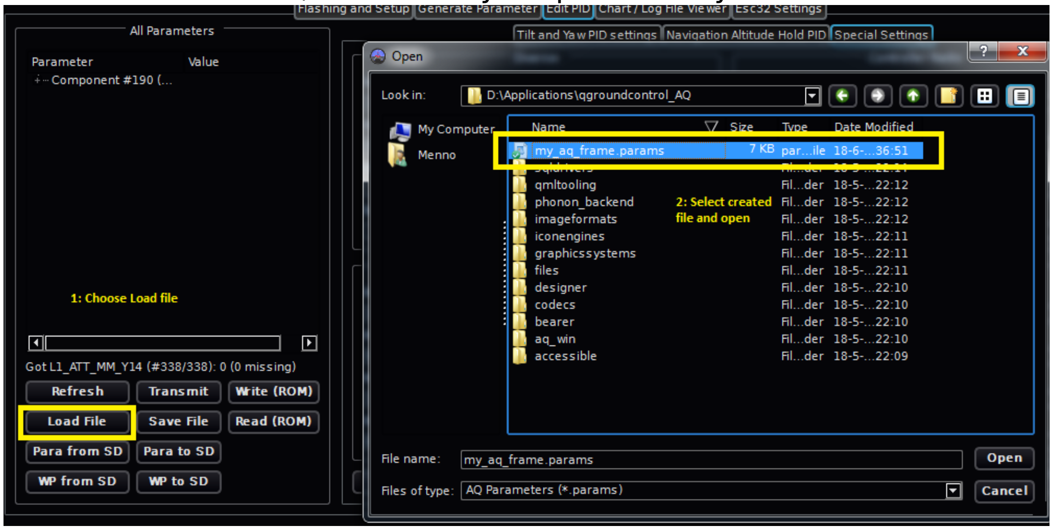
[Link to embedded object]

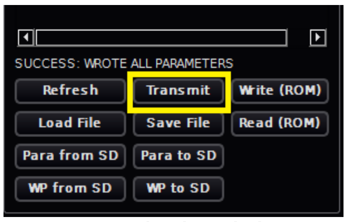
Upload the calculated parameters

Upload the calculated parameters

Now that the params file is complete, you can now upload it to the AutoQuad flight controller. For that you need a running flight controller with a serial connection to the Ground Control station. Forgot how ?

Connect to the AQ, open the ‘*Edit PID*‘ tab, then click on the ‘*Load file*‘ button in the ‘*All Parameters’* sidebar, and select your param file you created with the calculations:

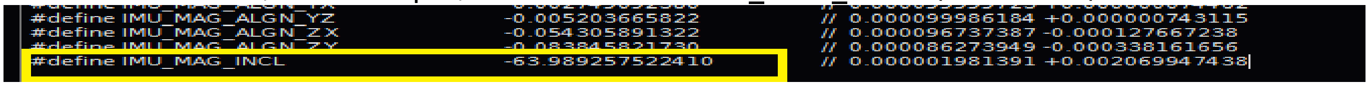




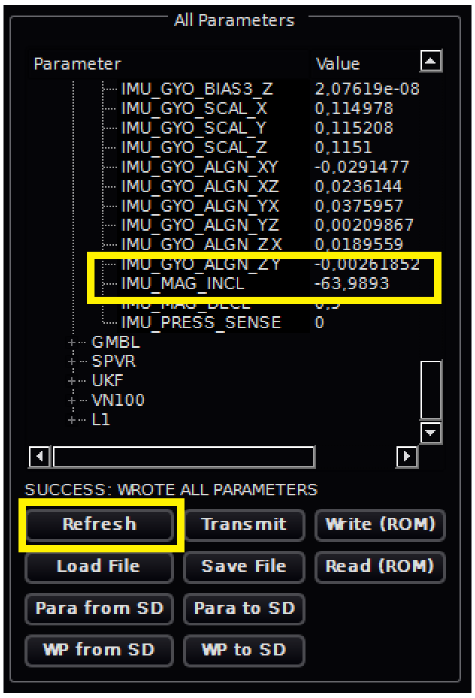
Once loaded, click on ‘*Transmit*.’ A successful transmission status message will be displayed beneath the parameters list.

Random check if upload was successful

To verify if the AutoQuad is actually using your created data, go back to the generation screen and check, for example, the calculated IMU_MAG_INCL. (-63.98…..)



```
#define IMU_MAG_ALGN_YZ          -0.005203665822        // 0.000099986184 +0.000000743115
#define IMU_MAG_ALGN_ZX          -0.054305891322        // 0.000096737387 -0.000127667238
#define IMU_MAG_ALGN_ZY          -0.083845821730        // 0.000086273949 -0.000338161656
#define IMU_MAG_INCL             -63.989257522410       // 0.000001981391 +0.002069947438
```



```
All Parameters

Parameter                Value
    IMU_GYO_BIAS3_Z      2,07619e-08
    IMU_GYO_SCAL_X       0,114978
    IMU_GYO_SCAL_Y       0,115208
    IMU_GYO_SCAL_Z       0,1151
    IMU_GYO_ALGN_XY      -0,0291477
    IMU_GYO_ALGN_XZ      0,0236144
    IMU_GYO_ALGN_YX      0,0375957
    IMU_GYO_ALGN_YZ      0,00209867
    IMU_GYO_ALGN_ZX      0,0189559
    IMU_GYO_ALGN_ZY      -0,00261852
    IMU_MAG_INCL         -63,9893
    IMU_MAG_DECL         [illegible]
    IMU_PRESS_SENSE      0
+ GMBL
+ SPVR
+ UKF
+ VN100
+ L1

SUCCESS: WROTE ALL PARAMETERS
```

Return to the PID screen and press '*refresh*' in the '*All Paremeters*' window. Fold out the Parameter tree, expand the IMU tree and search for the IMU_MAG_INCL. Note that it indeed shows the same value as from your params file. Don't worry if the number of decimal places isn't exactly the same.

After you have transmitted the parameters to the FC make sure you **write it to the ROM** so it will be stored permanently.

Congratulations! Your AutoQuad is now using the calculated calibrated values!

Now that you are done with all that, you may want to read Calibration FAQ & Additional Documentation and especially How To Determine A Good Calibration Result.

Resetting parameter values to default ("factory reset")

You can achieve a factory reset in 2 different ways:

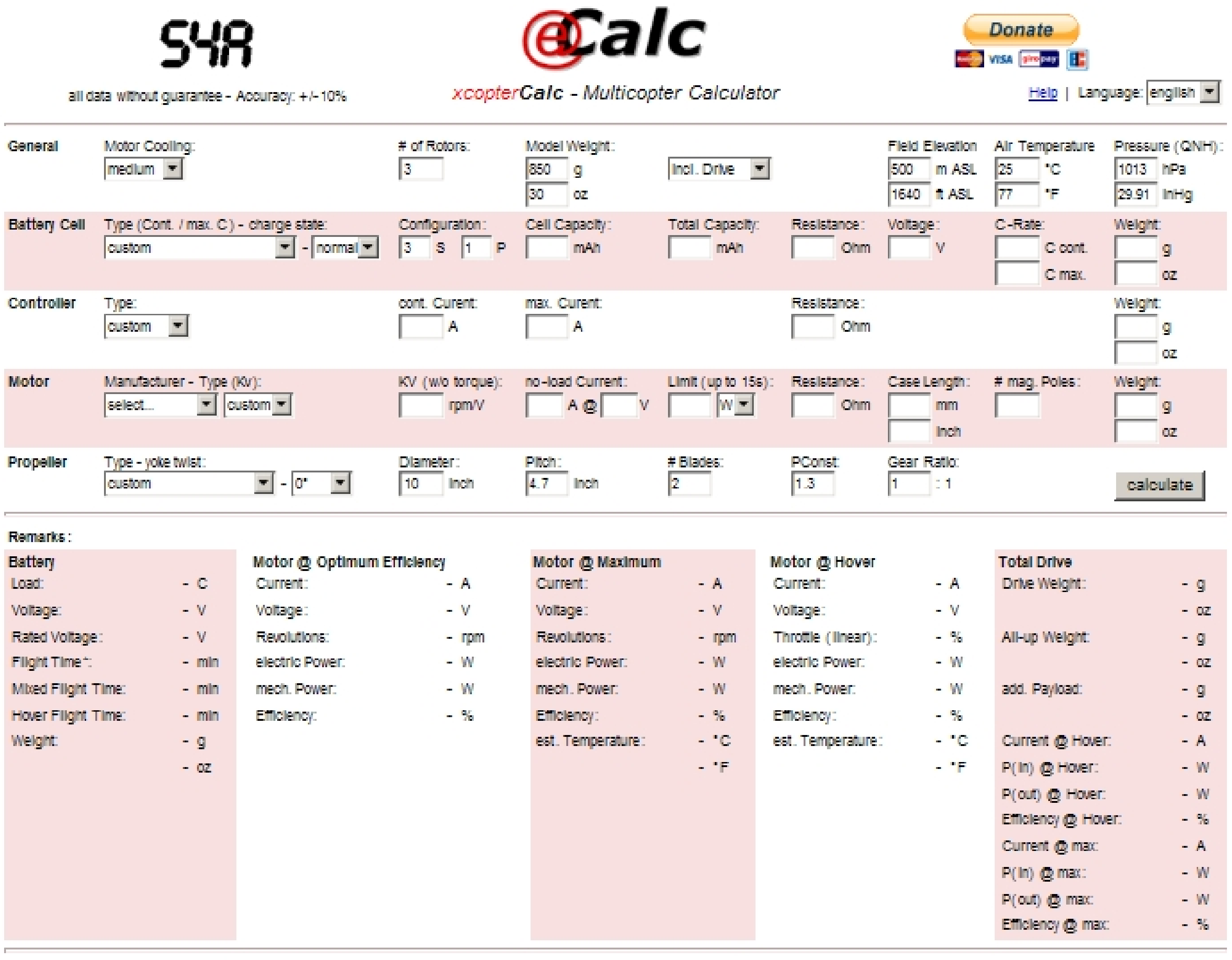
A. Change your CONFIG_VERSION to 0 in the *Edit PID -> All Parameters* widget of QGC, transmit, and write the changes. After a reboot your values are reset to default.
B. Create a params.txt file on your SD card only containing the text line:
   #define CONFIG_VERSION 0
   Insert the card into your AQ board and reboot.

Part 3 - Tuning & Troubleshooting
Tuning and troubleshooting

Tuning and troubleshooting

On the following pages we will concentrate on tuning the AutoQuad flight controller to your needs or a guide for basic troubleshooting.

Hint: a very good tool for motor/prop/lipo calculations is eCalc



all data without guarantee - Accuracy: +/-10%

xcopterCalc - Multicopter Calculator

Help | Language: english

General	Motor Cooling: medium	# of Rotors: 3	Model Weight: 850 g / 30 oz	Incl. Drive	Field Elevation: 500 m ASL / 1640 ft ASL	Air Temperature: 25 °C / 77 °F	Pressure (QNH): 1013 hPa / 29.91 inHg	
Battery Cell	Type (Cont. / max. C) - charge state: custom - normal	Configuration: 3 S 1 P	Cell Capacity: mAh	Total Capacity: mAh	Resistance: Ohm	Voltage: V	C-Rate: C cont. / C max.	Weight: g / oz
Controller	Type: custom	cont. Current: A	max. Current: A		Resistance: Ohm			Weight: g / oz
Motor	Manufacturer - Type (Kv): select... custom	KV (w/o torque): rpm/V	no-load Current: A @ V	Limit (up to 15s): W	Resistance: Ohm	Case Length: mm / inch	# mag. Poles:	Weight: g / oz
Propeller	Type - yoke twist: custom - 0°	Diameter: 10 inch	Pitch: 4.7 inch	# Blades: 2	PConst: 1.3	Gear Ratio: 1 : 1		calculate

Remarks:

Battery		**Motor @ Optimum Efficiency**		**Motor @ Maximum**		**Motor @ Hover**		**Total Drive**	
Load:	- C	Current:	- A	Current:	- A	Current:	- A	Drive Weight:	- g
Voltage:	- V	Voltage:	- V	Voltage:	- V	Voltage:	- V		- oz
Rated Voltage:	- V	Revolutions:	- rpm	Revolutions:	- rpm	Throttle (linear):	- %	All-up Weight:	- g
Flight Time*:	- min	electric Power:	- W	electric Power:	- W	electric Power:	- W		- oz
Mixed Flight Time:	- min	mech. Power:	- W	mech. Power:	- W	mech. Power:	- W	add. Payload:	- g
Hover Flight Time:	- min	Efficiency:	- %	Efficiency:	- %	Efficiency:	- %		- oz
Weight:	- g			est. Temperature:	- °C	est. Temperature:	- °C	Current @ Hover:	- A
	- oz				- °F		- °F	P(in) @ Hover:	- W
								P(out) @ Hover:	- W
								Efficiency @ Hover:	- %
								Current @ max:	- A
								P(in) @ max:	- W
								P(out) @ max:	- W
								Efficiency @ max:	- %

WARNING! There have been reports of certain PID values over saturating the motor outputs possibly causing a crash if not adjusted properly. Read our forum thread here**.**

The tuning pages are divided into sections:

- PID Tuning theory : to tune the different PID controllers in AutoQuad it is vital that the basic principles of a PID controller are understood. This page will try to explain a PID controller.

- Attitude controller tuning : tuning the flight controller attitude behavior like the rate and stabilization.
- Navigation controller tuning: tuning the navigation controller behavior like altitude, navigation, position hold.
- Troubleshooting : a guide to parameters and recommendations. It can and will grow with experience from the community
- Logfile Analysis : control each and any behaviour of your AutoQuad and your flights

What are all these numbers for?

What's all this number stuff?

Collecting the calibration numbers can be quite intimidating at first. This section tries to give you an idea what they are used for, in the hope it will make the collection process less bothersome.

The programs Cal and Sim3 create 81 values from the static and dynamic logs that you have gathered.

There are 3 sensor groups — the gyros, the accelerometers, and the magnetometers — and each have an X, Y and Z axis. So there are 9 data points affecting everything the calculations do.

The values generatet correct for the manufacturing differences of each sensor, how accurately they are mounted, temperature effects, and how they are affected by external influences like wires, ESCs, motors, etc.

Each sensor axis has a Bias value and a Scale value for each axis. These values vary with temperature, so the programs Cal and Sim3 try to find 3 numbers that best fit a 3rd order polynomial equation that represents the corrections needed over the operating temperature range. There are temperature corrections for Bias for all three axis' on all three sensors. There are also corrections for Scale for the accs and mags. Since the gyros would require some sort of calibrated rate of change, the Scale is taken from the manufacturer's data sheet.

So, if you've been calculating all this, we're up to 63 different values. 24 each for accel and 24 for the mags. Only 15 for the gyros since we leave out the scale temperature calibration and only apply the bias temperature calibration.

The `define` names clearly explain what each value is used for. If you look closer at the source code, you can see how they are skillfully applied.

Currently, more accurate gyro bias values are generated on AQ startup, so those values are not used at this time, but the temperature corrections are being applied.

There are also 18 values created that correct for the slight misalignment of the sensors. That brings it up to 81, and with Magnetic Inclination and Declination we get a rousing 83 values that will make the AQ fly better than its competitors… if we generate good numbers!

The proprietary programs Cal and Sim3 are used to take all the data provided, sometimes leveraging one against the other to come up with a unique set of numbers for each board's installation.

The idea is simple, the more accurate the data, the better the machine will fly.

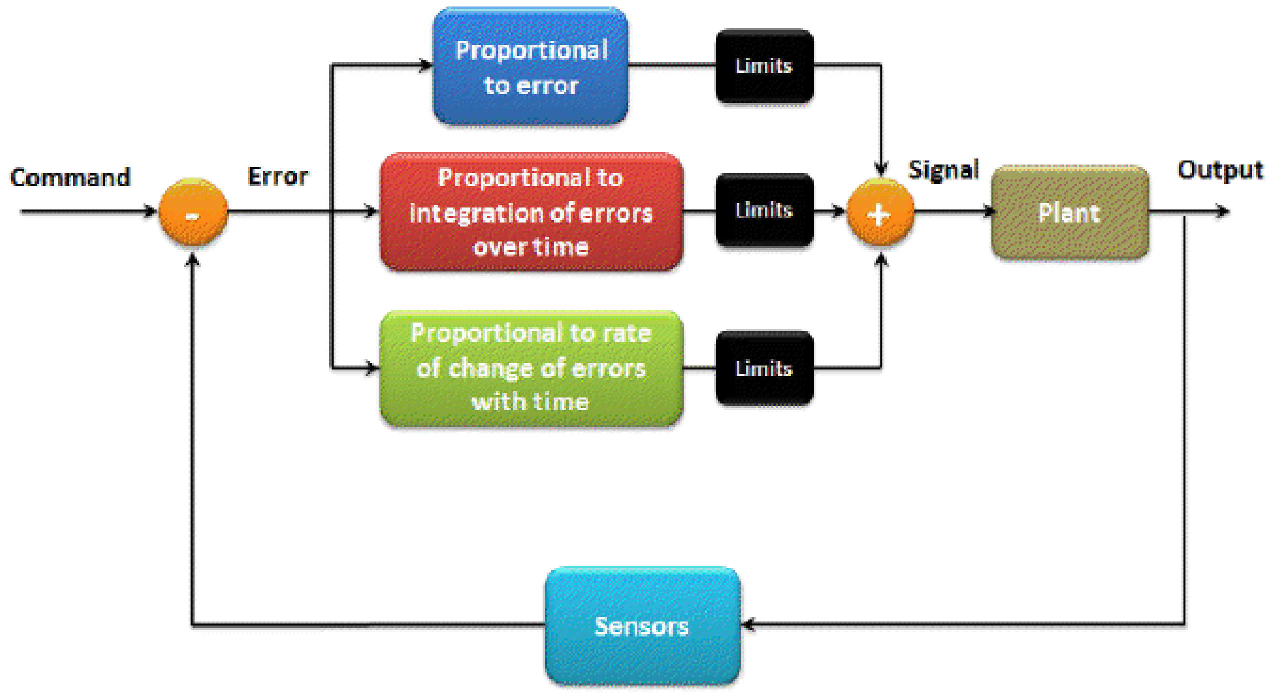
PID Tuning theory

PID Tuning (theory)

I’m pretty sure you heard the term “PID” somewhere! Whether in R/C planes and multi rotors forums, in automated factories, in motor control, climate control, or even in during your coursework; PID is the most famous controller around. Let’s start with a little bit of theory and then dive in.

What PID stands for?

It stands for Proportional, Integral, and Derivative controller. It’s a mathematical description of the way you think. PID helps you automatically achieve your goal, exactly the same way you used to do it manually. This diagram shows a general structure for a PID controller.


Proportional to error
Limits
Command
Error
Proportional to integration of errors over time
Limits
Signal
Plant
Output
Proportional to rate of change of errors with time
Limits
Sensors

So, what do you need to form a PID controller?

You need the following six basic elements:

- **Error**: It is the difference between your command (e.g., speed up your car, tilt your copter to the right, increase the room temperature) and the output of your car, AutoQuad, air conditioner, etc. You measure this output using sensors. The AutoQuad has variety of advanced sensors to measure its position, speed, and acceleration. (GPS, GYRO, ACC, MAG, and pressure sensors)
- **Proportional** term P: It is a constant directly related to the amount of error. If you have large error, this term gives you a large output. And if you have a small error, it’ll give you a small output, that simple! The P term affects the speed to reach your target.

- **Integral** term I: It is a constant related to the integration (summation) of errors over time. If your error is increasing, this term gives you a large output. However, if the error is decreasing, the I term gives you a small output. Thus it's used to fine tune your results, i.e., when you almost reach your goal, the P term cannot serve you anymore (actually it works against you here!), the I term is the one you can count on to drive you error signal to zero.
- **Derivative term D**: It is a constant related to the rate of change (derivative) of errors with time. What does this mean? It means that if your error signal changes rapidly, i.e., you have a highly dynamic system like a multi copter, the D term will give you higher output to catch up with the changes. On the other hand, if your error changes slowly, like in the room temperature example, the D term won't find anything fast enough to amplify. Thus it'll look for your noise signal (which usually has a high frequency) and amplify it to make your life miserable! The D term is a very dangerous controller if it's not tuned perfectly!
- **Limits**: You need to limit the output of each of the previous controllers! Your motors might accept only a voltage between 4 and 12 volts for example. You don't want to stop the motor or destroy it just because the PID controller suggested that! Use your brain.
- Finally, you need your system of course! Unless you're satisfied with simulations.

So, how do you tune your PID parameters to the optimal response?

Most often tuning is an art more than a science. Observe the system and use your intuitive guess and logical reasoning. Here are seven golden rules for general PID tuning:

1. After nulling all the parameters, increase the P term so that the output reaches the target in the shortest possible time.
2. If your output starts oscillating, it means you have too much P. Lower your P term until the oscillation disappears. You'll end up slightly higher or lower than your target. Don't worry; we'll fix that in the next step.
3. Now, increase I term slightly until your error goes away. Note that usual I values are very small (in the order of one thousandth for example) and they're dependent on the update rate of your PID loop. The I term is very useful when you have outside error signals affecting your system (e.g. wind in a multi copter). It drives your error to zero whenever possible.
4. If you feel your output is oscillating and it was not before you adjusted your I term, lower I slightly.
5. For many slow dynamic systems, your job is almost done! You just have to jump to the last step.
   When dealing with highly dynamic systems, however, you need to adjust the D term. If you feel your output "lagging" behind the error variations and trying hard but failing to catch them, increase this term slightly.
6. If your system starts to oscillate with high frequency and small transitions, you probably have too much D term which is amplifying your noise. Decrease D appropriately. If your system,however, has too much noise, it's better to keep this parameter to zero.
7. Last but not least, watch your limits! If you were changing the previous parameters without any noticeable change in the output, remember that limits cut down your output signal. Increase them probably, but be careful not to burn or saturate your

system.

Attitude tuning

Attitude tuning

WARNING! There have been reports of certain PID values over saturating the motor outputs possibly causing a crash if not adjusted properly. Read our forum thread here**.**

Let’s dive into the special set of PIDs available in AutoQuad. Since AQ is relatively a complicated system, it uses multiple PID loops to control your vehicle. Some of these loops are separate, and some are nested inside each other. There are basically two separate sets of PIDs, one dedicated to the attitude and the other to the navigation.

Be aware that default values are suitable for most “normal” frames; you’ll probably just need a little bit of fine tuning. If your frame, however, has different structure, weight distribution, or unusual features -even small ones- that will require appropriate PID tuning. In any case, when adjusting any PID parameter, start with small increments/decrements, a 20%-30% of the current value.

This page will cover the Attitude PID controller

This set of PIDs control how your AutoQuad behaves on each of yaw, pitch and roll axes. It consists of two nested PID loops: the rate PID and the angle PID.

The first (inner) loop controls the rate (speed) of change of the gyro signal. Its goal is to keep this rate of change at 0 radian/sec. When the rate of change (which is a derivative) is zero, this means the change is constant. The rate PID here aims to keep any gyro signal in a constant state, i.e., the gyro is not tilting, which gives your AQ its incredibly stable behavior! *Note* that this doesn’t mean the gyro output should be zero. It can be anything, but it should be constant with time. This gives your AQ the ability to balance itself at any given angle.

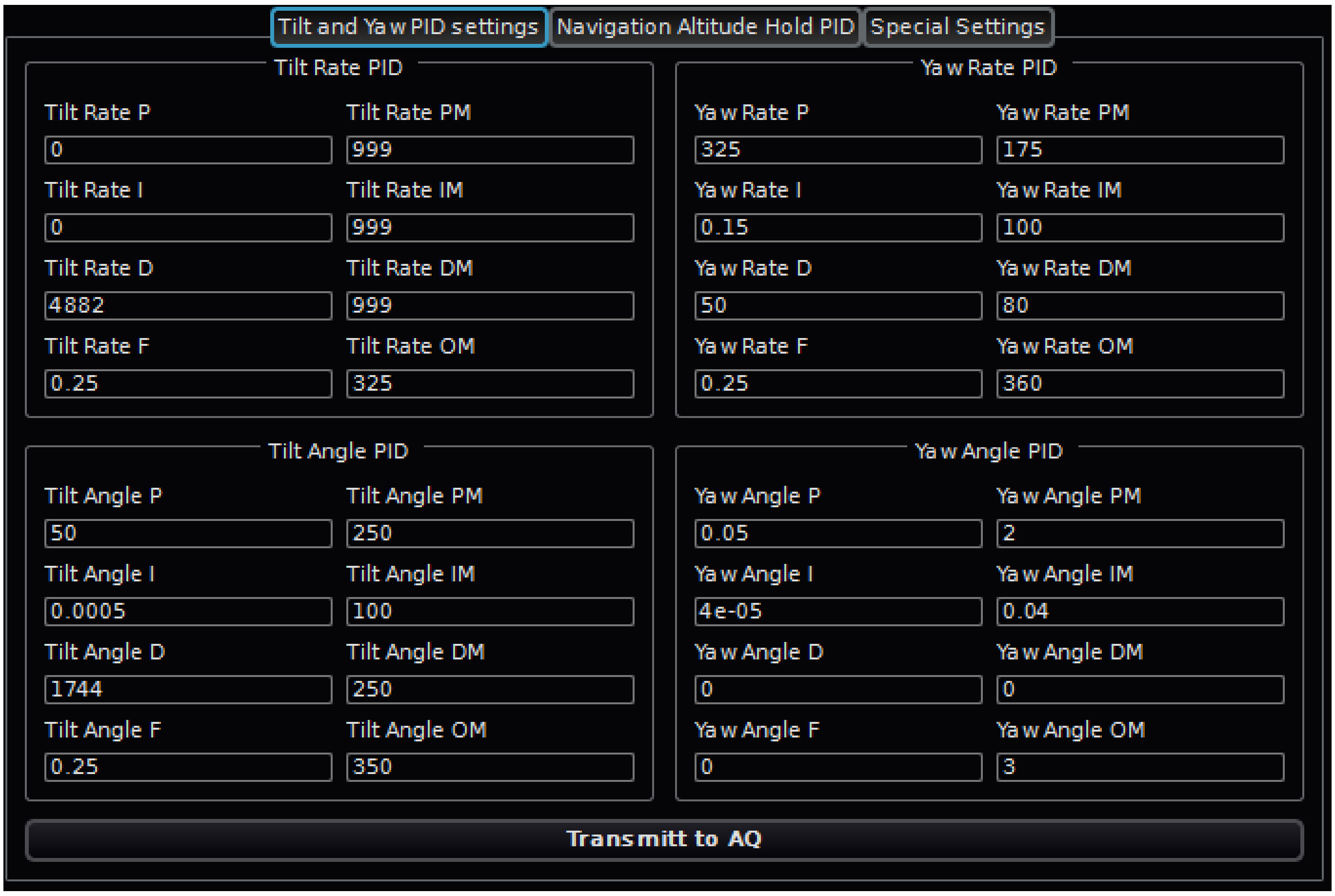
The tilt rate PID controls the AQ’s pitch and roll stability, while the yaw rate PID controls the AQ’s yaw stability. Since gyro is highly sensitive to small movements and the multi copter is, by definition, a highly vibrating system, the tilt rate D value is chosen quiet large, shown in the next figure, in order to compensate for those vibrations.

The rate angle controller only operates with a D term because the controller set point (command) is set at zero and the system changes are quiet fast.

To respond quickly enough to these changes, the rate PID loop is run at 400 Hz. Gyro signals are used directly without any mathematical filtering or correction which enables the system to achieve a quick dynamic behavior (meaning that AQ responds very quickly and actively to the R/C or autopilot commands). To guarantee not to amplify the noise signals with this high D value, the ARM Cortex-M4 MCU applies a tremendous oversampling (262.4 KHz) to the gyro signals.

The yaw rate PID is a typical PID controller which is responsible of your AQ’s yaw orientation. Since vibrations don’t affect the yaw orientation that much, a relatively small value of D is chosen.

Appropriate P and I values are chosen to achieve accurate and quick yaw lock.



Tilt Rate PID

Tilt Rate P	0
Tilt Rate PM	999
Tilt Rate I	0
Tilt Rate IM	999
Tilt Rate D	4882
Tilt Rate DM	999
Tilt Rate F	0.25
Tilt Rate OM	325

Yaw Rate PID

Yaw Rate P	325
Yaw Rate PM	175
Yaw Rate I	0.15
Yaw Rate IM	100
Yaw Rate D	50
Yaw Rate DM	80
Yaw Rate F	0.25
Yaw Rate OM	360

Tilt Angle PID

Tilt Angle P	50
Tilt Angle PM	250
Tilt Angle I	0.0005
Tilt Angle IM	100
Tilt Angle D	1744
Tilt Angle DM	250
Tilt Angle F	0.25
Tilt Angle OM	350

Yaw Angle PID

Yaw Angle P	0.05
Yaw Angle PM	2
Yaw Angle I	4e-05
Yaw Angle IM	0.04
Yaw Angle D	0
Yaw Angle DM	0
Yaw Angle F	0
Yaw Angle OM	3

AutoQuad PID config screen in QGroundControl

Rate limits

The attitude rate PID generates required PWM width (in microseconds) to control the motors in order to achieve stability. PM, IM and DM limits the output of each PID term so that it doesn’t burn the motors or saturate the system. The OM parameter defines the maximum output from the rate controller, thus no matter how much you set the PID limits, the output won’t exceed OM value.

F term

The F term is used to smooth out the D term. Since we have relatively large D value, it’s not practical to apply all that output to the motors in a small time step (1/400Hz = 2.5 milliseconds). The F term regulates how much of the D term is applied at each time step. In the above example, a value of 0.25 means only 25% of the D term is applied to the motors at each time step.

Attitude Angle PID

The second (outer) loop in the attitude control regulates the angle (in degrees) at which the AQ rotates on each of yaw (yaw angle PID), pitch and roll axes (tilt angle PID). This loops is run at 200Hz.
Once again, a relatively large D value is used for tilt angle control while no D term at all is used in the yaw angle. The set point (angle) is either commanded directly from the user's R/C input or from the navigation PID controllers (autopilot) or a combination of both depending on the operating mode at the time.
Again, the output of this stage is R/C PWM pulse lengths measured in microseconds. The output of this PID is mathematically added to the output of the inner rate PID and both are sent to the motors and added with the current throttle setting at each time step.

(Please add default settings here)

So to think of it simply, the outer PID is constantly fighting to maintain a given angular set point and the inner PID is fighting for a zero change in rotation rate (but not necessarily a zero rotation.) Sometimes they are pushing or pulling together and sometime they are fighting each other. The result is a very fine control over the controller output which leads to a very stable platform when paired with high performance ESC's.

Maximum tilt and roll.

AutoQuad is designed for stable flight and not acrobatic flight. The angular movements are limited to provide maximum stability.
The default behavior will limit the angle to around 37 degrees.
The max tilt (pitch) and roll angle is controlled by the these two parameters:

CTRL_FACT_PITC	0.05f	// user pitch multiplier
CTRL_FACT_ROLL	0.05f	// user roll multiplier

They are multiplied by the elevator and roll channel data from the radio. So if you're radio produces -750 to 750 output, your max tilt range will be +-37.5 degrees.

While there is room for more tilt, be careful as you get to much higher values. This theory of operation breaks down as you approach the 90 deg singularities. For any accuracy over 45 deg on more than one axis, you will need to ditch the PID controller and go with something quaternion based

[Link to embedded object]

Navigation tuning

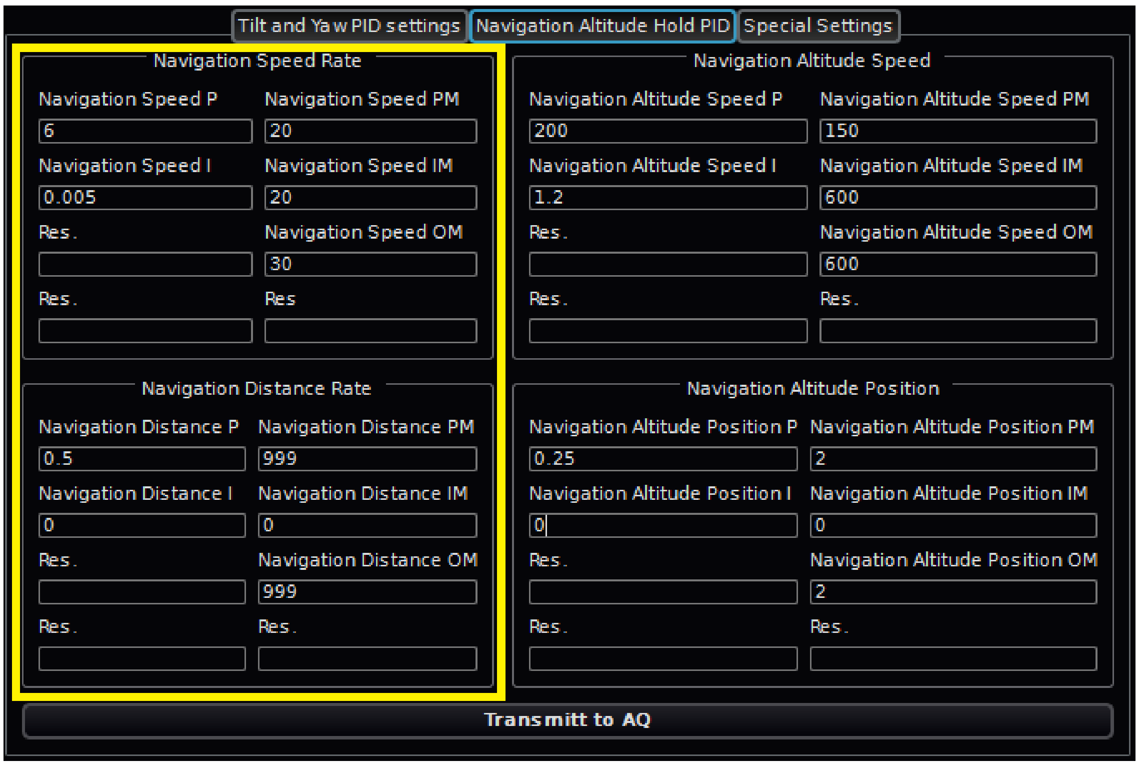
Navigation PIDs

In the navigation PIDs, we control the vertical movement (altitude) and horizontal movement (location)of your AutoQuad. Both PIDs work independent from each other and they're run at 200Hz.

Horizontal Navigation PIDs

There are two horizontal navigation PIDs connected to each other. One controls AQ's horizontal speed (navigation speed rate) and the other controls its distance to target (navigation distance rate). You won't find any D terms here because we deal with relatively slow changing signals, thus these are actually PI controllers.

WARNING! There have been reports of certain PID values over saturating the motor outputs possibly causing a crash if not adjusted properly. Read our forum thread here**.**



Navigation Speed Rate

Navigation Speed P	Navigation Speed PM
6	20
Navigation Speed I	Navigation Speed IM
0.005	20
Res.	Navigation Speed OM
	30
Res.	Res

Navigation Distance Rate

Navigation Distance P	Navigation Distance PM
0.5	999
Navigation Distance I	Navigation Distance IM
0	0
Res.	Navigation Distance OM
	999
Res.	Res.

Navigation Altitude Speed

Navigation Altitude Speed P	Navigation Altitude Speed PM
200	150
Navigation Altitude Speed I	Navigation Altitude Speed IM
1.2	600
Res.	Navigation Altitude Speed OM
	600
Res.	Res.

Navigation Altitude Position

Navigation Altitude Position P	Navigation Altitude Position PM
0.25	2
Navigation Altitude Position I	Navigation Altitude Position IM
0	0
Res.	Navigation Altitude Position OM
	2
Res.	Res.

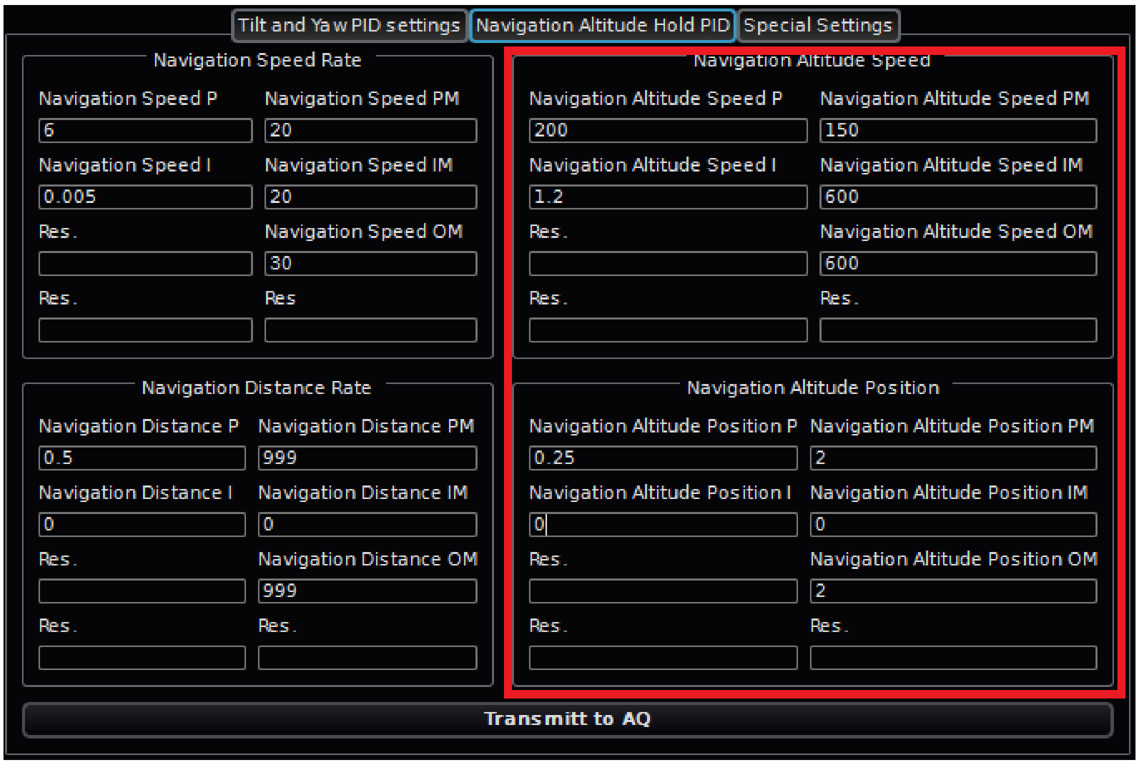
The *navigation distance rate* accepts distance commands (in meters) and calculates the error depending on current position. This PI outputs a horizontal speed command (m/s), which *enters the navigation speed rate PI*. Here the speed is compared with current speed from sensors and the error is regulated and outputted as a tilt angle (in degrees).

In this example (actually the defaults), every one meter error in distance will result in a 0.5 m/s velocity. This velocity is entered into the speed PI, which gives 6 degs of tilt for each m/s input. Another 0.005 degs is added to the output every 1/200′s of a second it takes to reach the target, i.e., each one second away from the target will add 1 degree to the output. The maximum allowed tilt change, is 20 degs for P term and 20 degs for I term. However, the maximum allowed tilt change from the whole PI is only 30 degs.

So if we are 100m away from the target, the first controller will try to set it on 0.5m/s x 100m = 50m/s.
The second controller will try to get that 50m/s x 6 degrees = 300 degs. But, it is limited to 20 degrees and 1 x 100 = 100 degs are added each second.
However, the whole output is limited to a combined 30 degs. With these values, when the target is 100m away, the copter will tilt a maximum of 30 degs.

The navigation Speed PM,IM,OM values are in degrees. They behave as standard PID controls.
The PM is used to navigate in gusting wind condition, the IM in steady wind, the OM limits the total

Vertical controller (Navigation Altitude PID)



Navigation Speed Rate

Navigation Speed P: 6	Navigation Speed PM: 20
Navigation Speed I: 0.005	Navigation Speed IM: 20
Res.:	Navigation Speed OM: 30
Res.:	Res:

Navigation Altitude Speed

Navigation Altitude Speed P: 200	Navigation Altitude Speed PM: 150
Navigation Altitude Speed I: 1.2	Navigation Altitude Speed IM: 600
Res.:	Navigation Altitude Speed OM: 600
Res.:	Res.:

Navigation Distance Rate

Navigation Distance P: 0.5	Navigation Distance PM: 999
Navigation Distance I: 0	Navigation Distance IM: 0
Res.:	Navigation Distance OM: 999
Res.:	Res.:

Navigation Altitude Position

Navigation Altitude Position P: 0.25	Navigation Altitude Position PM: 2
Navigation Altitude Position I: 0	Navigation Altitude Position IM: 0
Res.:	Navigation Altitude Position OM: 2
Res.:	Res.:

The same concepts applied early are utilized here. In the above settings (screen), every one meter error in altitude will result in a 0.25 m/s vertical velocity. The maximum speed is 2 m/s.
The vertical velocity is entered into the navigation altitude speed PI, which gives 200 microseconds of throttle PWM for each m/s input.
Another 1.2 microseconds is added to the output every 1/200′s of a second it takes to reach the target, i.e., each one second away from the target will add 240 microseconds to the

output.
The maximum allowed PWM width, is 150 microseconds for P term and 600 microseconds for I term.
The maximum allowed PWM width from the whole PI, however, is 600 microseconds of throttle response.

[Link to embedded object]

By now you sure to be bored to death from all this reading! OK, let’s get into what’s important. Let’s see how PIDs can save your (and your copter) life, and what kind of problems they can solve for you.
Move over to the practical tuning and troubleshooting.

Magnetic Inclination and Declination

Magnetic Inclination and Declination

Magnetic declination is the angle between magnetic north (the direction the north end of a compass needle points) and true north. The declination is positive when the magnetic north is east of true north.

Magnetic inclination is the angle made by a compass needle when the compass is held in a vertical orientation. Positive values of inclination indicate that the field is pointing downward, into the Earth, at the point of measurement.

For the AutoQuad calculations the magnetic *inclination* is vital. For flying, the magnetic *declination* is also needed. The page will display the values for your location in degrees/minutes. The AutoQuad ground station software has a built-in widget for converting those to decimal values, which is what you will use in the calculations and flight parameters.

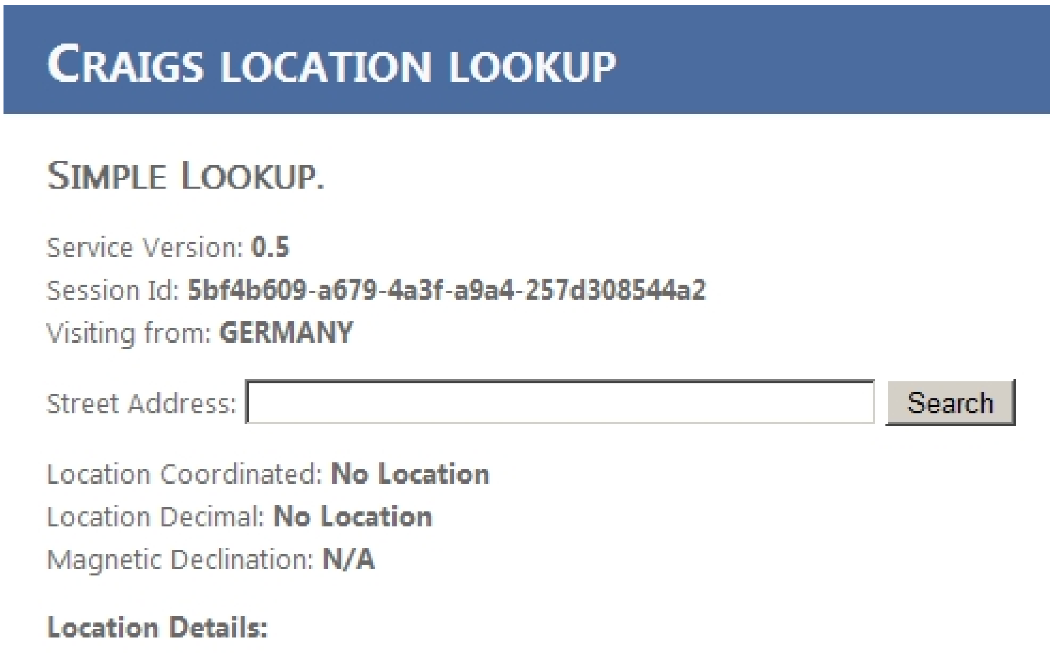
Remember the numbers in the example? 67.6 and 0 3 East. Entering the numbers in the converter will give: inclination 67.1 and declination 0.05.

Inclination is always reversed in AQ, meaning that on the northern hemisphere number will always be negative in the defines and positive for the southern hemisphere.**For the *inclination* value, always reverse of what the online tool comes up with for inclination.**

For *declination*, use the same sign (negative or positive) as indicated in the online tool — do not reverse this.

The final numbers for use in this example will be : -67.1 and 0.05
The declination (0.05) will be used in the final setup. The inclination is needed to complete the calibration calculations.

A great tool to determine the correct declination is Craig’s Location Lookup Tool:



CRAIGS LOCATION LOOKUP

SIMPLE LOOKUP.

Service Version: **0.5**
Session Id: **5bf4b609-a679-4a3f-a9a4-257d308544a2**
Visiting from: **GERMANY**

Street Address:

Location Coordinated: **No Location**
Location Decimal: **No Location**
Magnetic Declination: **N/A**

Location Details:

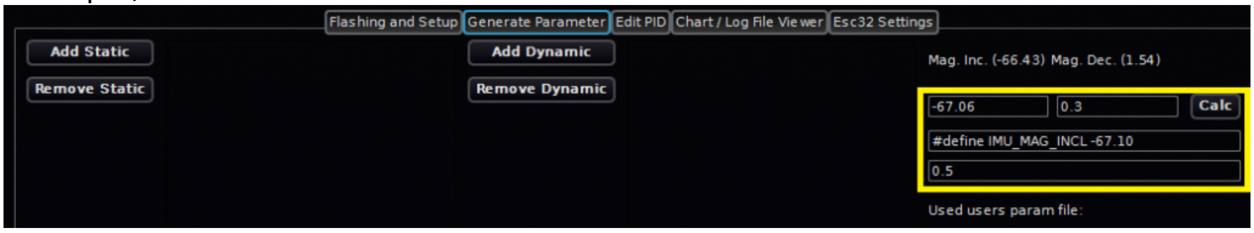
Magnetic Declination Setting

Magnetic Declination Setting

Only perform this step after all the calibrations and calculations are done.

After the generated parameter file has been uploaded, the final step is to set the magnetic declination.

The result you got from the magnetic declination and inclination page must be entered using the GCS. The inclination was already used in the calculations and is included in the generated .param file, so you do not need to update it, but the declination (0.5 in the example) needs to be set.



Mag. Inc. (-66.43) Mag. Dec. (1.54)

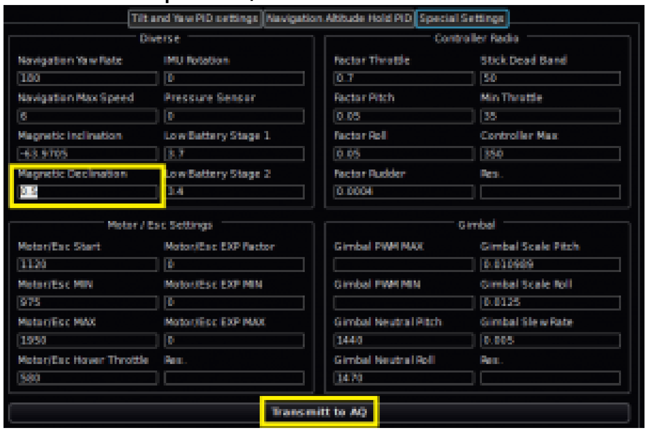
-67.06 | 0.3

#define IMU_MAG_INCL -67.10

0.5

Used users param file:

So, enter the correct declination value in the Edit PIDs -> Special Settings -> Diverse -> Magnetic Declination field, and press ‘*Transmit to AQ*‘ and then reply ‘*Yes*‘ to the prompt about saving values to ROM. This will permanently store the value in AQ (until the next firmware update).



Diverse

Navigation Yaw Rate: 180	IMU Rotation: 0
Navigation Max Speed: 6	Pressure Sensor: 0
Magnetic Inclination: -63.9705	Low Battery Stage 1: 3.7
Magnetic Declination: 0.5	Low Battery Stage 2: 3.4

Controller Radio

Factor Throttle: 0.7	Stick Dead Band: 50
Factor Pitch: 0.05	Min Throttle: 35
Factor Roll: 0.05	Controller Max: 350
Factor Rudder: 0.0004	Res.:

Motor / Esc Settings

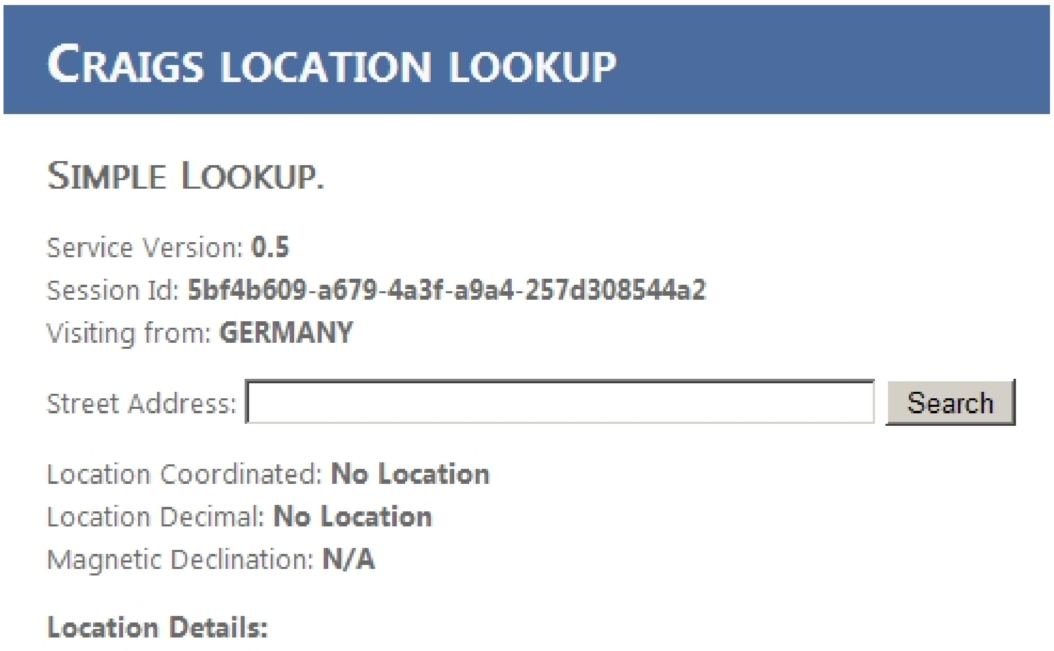
Motor/Esc Start: 1120	Motor/Esc EXP Factor: 0
Motor/Esc MIN: 975	Motor/Esc EXP MIN: 0
Motor/Esc MAX: 1950	Motor/Esc EXP MAX: 0
Motor/Esc Hover Throttle: 580	Res.:

Gimbal

Gimbal PWM MAX:	Gimbal Scale Pitch: 0.010989
Gimbal PWM MIN:	Gimbal Scale Roll: 0.0125
Gimbal Neutral Pitch: 1440	Gimbal Slew Rate: 0.005
Gimbal Neutral Roll: 1470	Res.:

That’s it, you should now be ready for the first careful flight test. Again, make sure you have done all the calibration steps before flying.

A great tool to determine the correct declination is Craig’s Location Lookup Tool:


CRAIGS LOCATION LOOKUP

SIMPLE LOOKUP.

Service Version: **0.5**
Session Id: **5bf4b609-a679-4a3f-a9a4-257d308544a2**
Visiting from: **GERMANY**

Street Address:

Location Coordinated: **No Location**
Location Decimal: **No Location**
Magnetic Declination: **N/A**

Location Details:

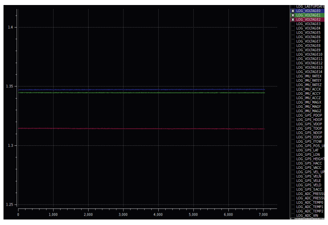
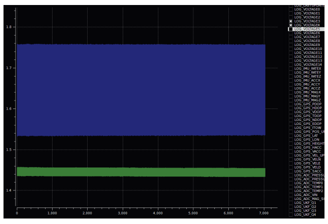
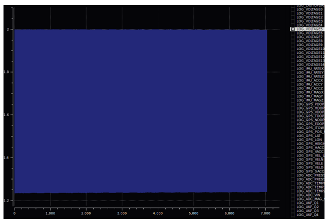
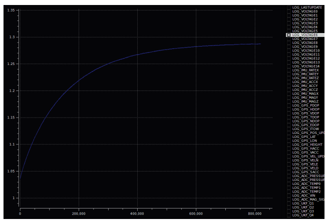
Verifying Sensor Voltages

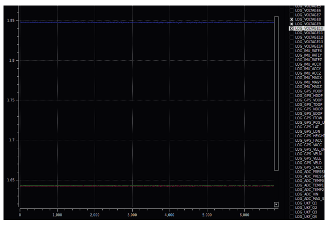
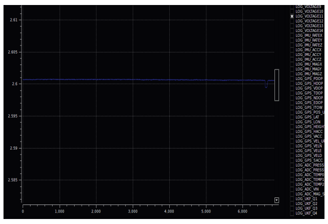
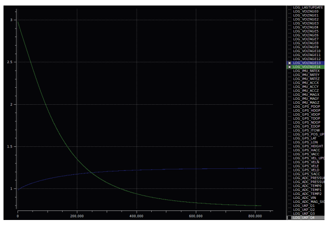
Verifying Sensor Voltages

Here is an overview of the ADC voltages expected on a new AQ board. Every board has been tested, so this is mostly posted for reference in case of suspected calibration problems.

To check the ADC voltages, mount the board with some standoffs, place it level and make a 30 second log. Then use the log viewer to check each individual value. Allow for around a 10% tolerance on each value, and that the values will vary with temperature and/or other factors, as noted.

The reference values below were taken with the AQ board in a level position, at room temperature (~20C).

Sensor	V_Log	Expected Value	Image
RATEX	0	Rate gyro output, static reading around 1.35 volts	
RATEY	1		
RATEZ	2		
MAGX	3	This looks like a band of voltage somewhere in the 1-2V range. It should not be zero, a straigh line, or 3.3 volts.	 
MAGY	4		
MAGZ	5		
TEMP1	6	Centigrade temperature of IDG500. Around 1.25 volts at room temp	
VIN	7	Voltage read from R3, R4 voltage divider. Rev1 board (4.7k): LOG_VOLTAGE7 / .221 = Vin; Rev2 board (8.45k):	

Sensor	V_Log	Expected Value	Image
		LOG_VOLTAGE7 / .136 = Vin;	
ACCX	8	Acc output — around 1.65 volts.	
ACCY	9		
ACCZ	10	Acc Z output — around 1.83 volts.	
PRES1	11	This is the normal pressure sensor installed and should be about 2.65 volts at sea level. It will go lower in voltage with higher altitude.	
PRES2	12	Pressure sensor with tube (not installed on beta boards) reads about 2.65 volts at sea level and goes lower with higher altitude. If installed, it should be within a .2 volts of other sensor.	
TEMP2	13	Centigrade temperature of ISZ500. Around 1.25 volts at room temp. Image shows board heating up from freezing temperatures.	
TEMP3	14	This is Kelvin temperature represented by a thermister on the board. Coming out of the freezer it can be near 3 volts. As the thermister heats up, its resistance goes down and the voltage will go below .5 volts. Image shows board heating up from freezing	

Sensor	V_Log	Expected Value	Image
		temperatures.	

Profiling battery discharge

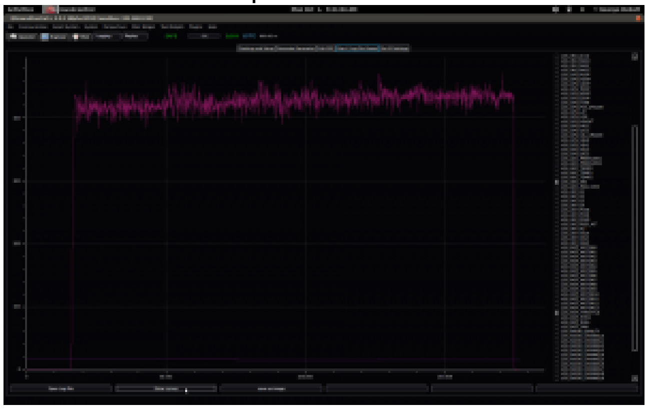
Profiling battery discharge

AutoQuad has a facility to estimate the State of Charge (SoC) of the flight battery in real time. The SoC value is not presently used by any functions; however it is displayed in the GCS' HUD. In the future, it could be used to determine remaining flight time or if there is enough battery charge to complete a mission plan.

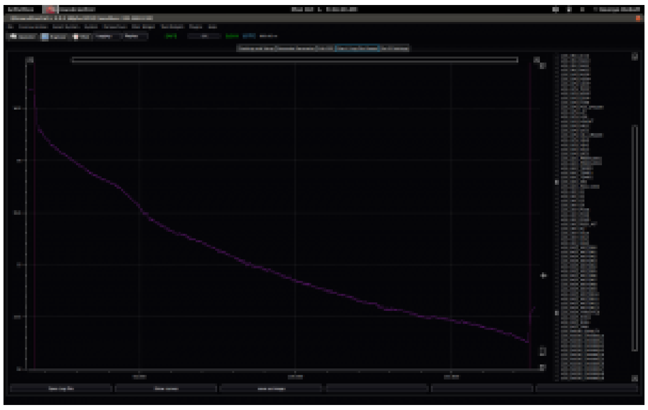
In order to determine the SoC, the flight battery's discharge curve has to be profiled. The default curve will not match yours.

With a fully charged battery pack and SD card logging enabled, lift off and maintain a smooth hover to the lowest pack voltage you would normally fly to. If your PositionHold is well setup, you can use that to maintain hover. Pick a windless day and stay out of ground effect.
After the flight, take a look at the log to verify that you achieved fairly linear motor output and smooth battery discharge.

Example:
Linear motor output overall. There was some breeze during this flight.



Battery voltage curve during the flight. The idea is to place a constant flight load on the pack to achieve a smooth discharge.



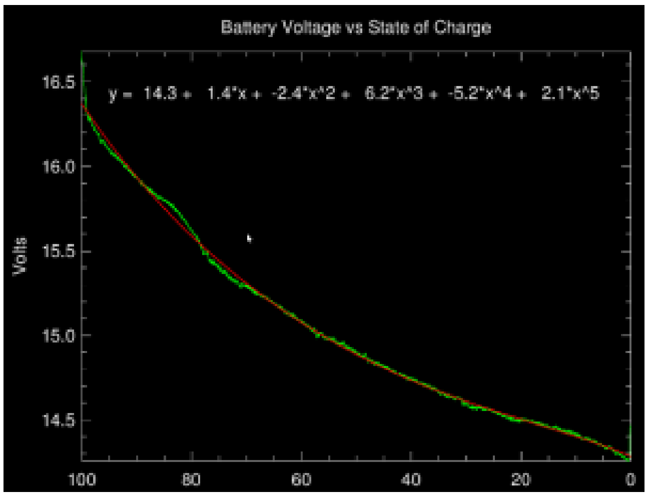
‘batCal’ – the battery profiler utility is located within the /ground directory of the AutoQuad source package. It is currently only available as source code. You will have to compile it yourself. Under Linux, you might have to install a few dev libraries for it to compile successfully.

Run ‘batCal’ with your hover log file as a parameter. If you want to use a higher voltage value than the end voltage of the flight as the zero SoC, you provide that as an additional ‘–zero=*value*‘ command line parameter.

Answer the prompts and ‘batCal’ will come up with a curve function for the pack’s discharge profile. The displayed SPVR_BAT_CRV values can now be used for that battery pack. Modify the onboard parameters within the SPVR section and save to ROM.

Naturally, if you fly with packs of different characteristics or age, they must be profiled individually and then modify onboard parameters when flying with that pack. If you change significant parts of your system, such as weight or props; you should re-profile.

If you told batCal to generate a plot, it might look something like this.



AQ uses the parameters to build a lookup table to determine the SoC (100% – 0%) based on the pack voltage.

If desired, you can modify the code to transmit the SoC via MAVLink.

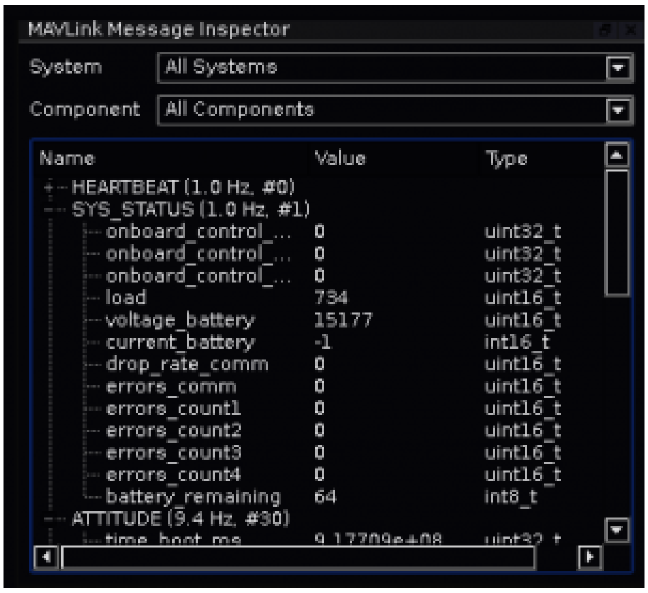
In aq_mavlink.c, change:

1

to

1

QGCS will then show the SoC (battery_remaining as a percentage)



```
MAVLink Message Inspector
System      All Systems
Component   All Components

Name                          Value         Type
+-- HEARTBEAT (1.0 Hz, #0)
--- SYS_STATUS (1.0 Hz, #1)
     |-- onboard_control_...  0             uint32_t
     |-- onboard_control_...  0             uint32_t
     |-- onboard_control_...  0             uint32_t
     |-- load                 734           uint16_t
     |-- voltage_battery      15177         uint16_t
     |-- current_battery      -1            int16_t
     |-- drop_rate_comm       0             uint16_t
     |-- errors_comm          0             uint16_t
     |-- errors_count1        0             uint16_t
     |-- errors_count2        0             uint16_t
     |-- errors_count3        0             uint16_t
     |-- errors_count4        0             uint16_t
     `-- battery_remaining    64            int8_t
--- ATTITUDE (9.4 Hz, #30)
     |-- time_boot_ms         9.17709e+08   uint32_t
```

Part 4 - Flying & Missions
Flying

Flying

Ok, everything is built up, tested and now you’re ready for the next step: flying.

You should always remove your propellers prior to starting, testing or connecting your battery for the first time.

Several flight modes are available:
Manual : attitude control is always enabled, but stick override. You can just fly around and when sticks are centered the controller will stabilize the multirotor. **Only in this mode you can arm / disarm the flightcontroller.**
Altitude/Position hold : With GPS lock position hold will be engaged with vertical override, without GPS lock (indoors?) pressure based altitude hold will be engaged with vertical override. Vertical override means; throttle controls altitude
DVH : Dynamic Velocity Hold, this mode cannot be engaged by an switch but is automatically activated when you override pitch and/or roll during position/altitude hold. When in position hold, you can just fly around and it will keep altitude (when you not use the throttle to control altitude).
Mission : Full automatic mode. At this moment the following events are possible;

- **Return to Launch** : return to set home position (fw 6.5)
- **Goto waypoint** : fly to waypoint (fw 6.5)
- **Orbit** : Circle mode with center heading (fw 6.6)
- **Take off** : Ground start to defined altitude (fw 6.6)
- **Land** : Engage landing with adjustable descend speed. Bump detection for ground. (fw 6.6)

Remember that you always will start in manual mode. To enforce that you can only arm the AutoQuad when all switches are in manual mode.

Flap switch in manual (down) and Aux 2 in normal (center) position. (Flying switches)

1. Set all switches in manual mode
2. Power on the AutoQuad – don’t touch the AutoQuad during calibration (1-5 sec).
3. **Wait for a solid GPS lock** (blue LED lit solid)
4. Watch your surroundings, spool up very gently and watch if the AutoQuad reacts without strange movements.
5. If all’s well, spool up to hover point and fly.

If the AutoQuad will not give a ready signal (green led flashing) it could be that it detects to much movement for calibration.
In this case try to find a spot with less disturbance but **without an iron mass**. That will upset the magnetometer.
Placing it behind a car is ***not*** the best place to start.

If the GPS lock is not solid but intermittent the altitude estimation could be incorrect!

After a successful flight the first GPS controlled flight should be the Return to home test.

Select from the following info pages:

- * *GPS navigation special notes * *
- Flight radio switches
- Vertical altitude override
- Failsafe events
- Mission Flight
- Tuning & Troubleshooting
- Logfile analysis

GPS Navigation notes

As you know by now, AutoQuad relies on GPS for accurate naviagtion and even in stabilized flight.

During startup the ublox GPS will acquire satellite fixes and if enough it will report the 3D lock. But for very precise location (both altitude as lon/lat) another fix is needed, the SBAS fix.
If you need ultimate accuracy, you should let the machine sit on the ground for 5 or so minutes soaking up all the sat info it can get before flying. If your backup battery is working, subsequent flights should have most of the ephemeral info stored from previous flights. One exception to this is the SBAS ephemerals, which for some reason are not stored. This means that the SBAS shift will happen at some point every time. Also note that this shift is not limited to vertical position, it will happen to a lesser degree for the horizontal. This is why you sometimes come back home from a mission to find you are 1.5m away from where you started.

US and EU SBAS

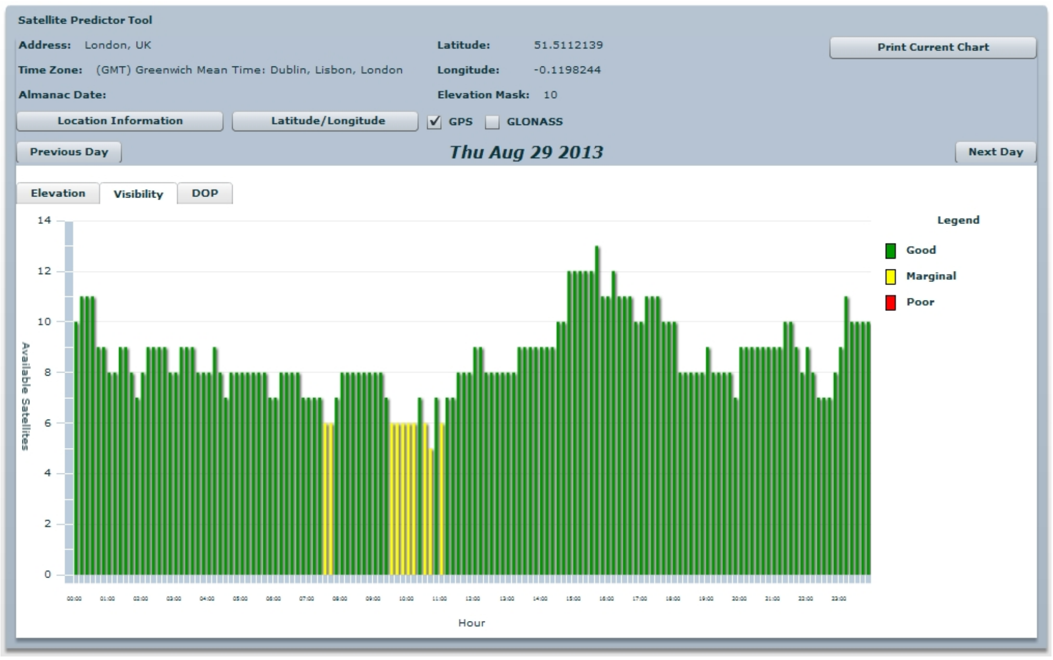
The system used in EU is EGNOS and is currently broadcasted on 120 (AOR-E), 124 (Artemis) with 126 and 131 coming online at some point in the future.
The system most used in the US is broadcasting on 135 & 138

The UBLOX LEA recievers can track 3 SBAS sats and will choose the best available
u-blox receivers are capable of receiving multiple SBAS satellites in parallel, even from different SBAS systems (WAAS, EGNOS, MSAS, etc.). They can be tracked and used for navigation simultaneously. At least three SBAS
satellites can be tracked in parallel. Every SBAS satellite tracked utilizes one vacant GPS receiver tracking channel. Only the number of receiver channels limits the total number of satellites used. Each SBAS satellite,which broadcasts ephemeris or almanac information, can be used for navigation, just like a normal GPS satellite.
For receiving correction data, the u-blox GPS receiver automatically chooses the best SBAS satellite as its primary source. It will select only one since the information received from other SBAS GEOs is redundant and/or could be inconsistent. The selection strategy is determined by the proximity of the GEOs, the services offered by
the GEO, the configuration of the receiver (Testmode allowed/disallowed, Integrity enabled/disabled) and the signal link quality to the GEO.

References for this highly technical subject:

- GPS Information
- What is SBAS
- Real Time Kinematic

GPS Predictor Tool

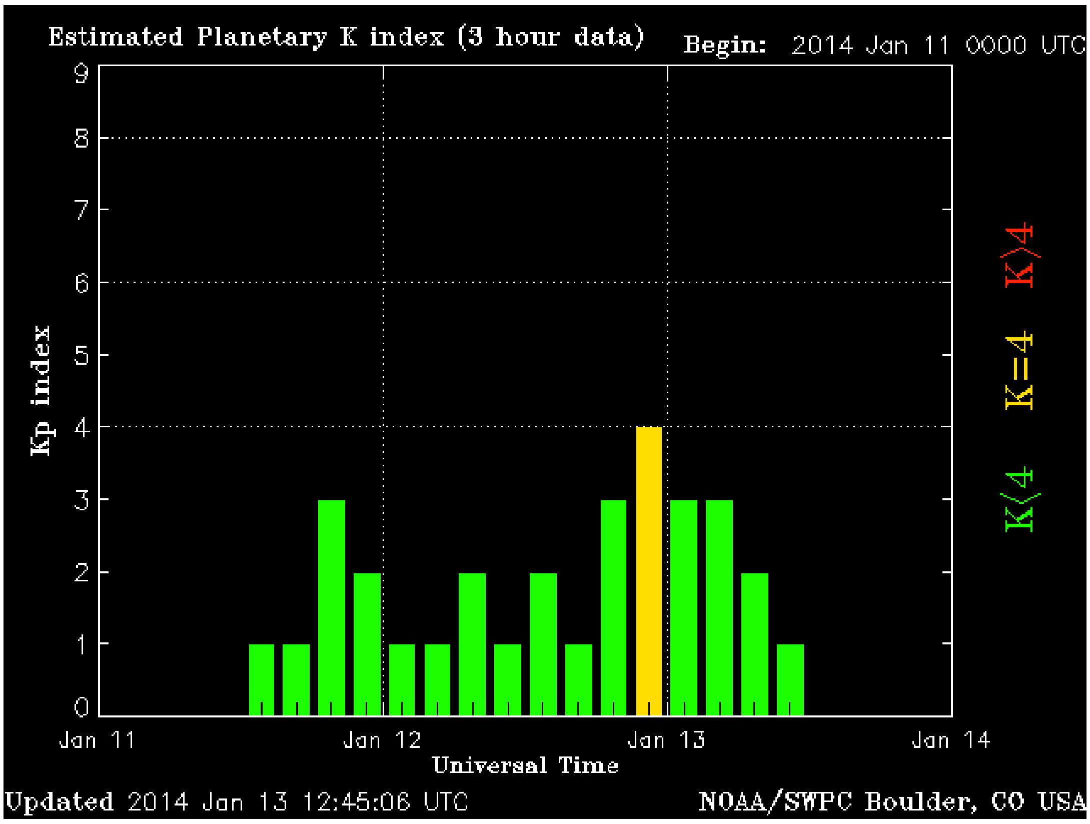


http://satpredictor.navcomtech.com

Note: beware of solar flares!

It has been reported numerous times that solar flares (sun storms) might affect your GPS reception and the accuracy of your automomous features that heavily depend on precise GPS data.

Here’s a graph displaying the actual solar flares. Be careful when the days are marked as yellow or red!



[Get more info from NOAA]

Solar X-rays:

ACTIVE

Geomagnetic Field:

UNSETTLED

Vertical altitude override

Vertical altitude override

In all flying modes other then manual a controlled vertical override is available.
In position hold for example you can increase the throttle and the copter will climb with a vertical speed proportional to the throttle, decreasing the throttle will cause the copter descending with a set maximum velocity around 1 m/s.

In short, any mode with automatic altitude control, the throttle will control the **altitude** with a throttle deadband in the middle to maintain the current height.
This is great to use in photographic situations, just use the throttle to climb or fall while staying put in the locked position.

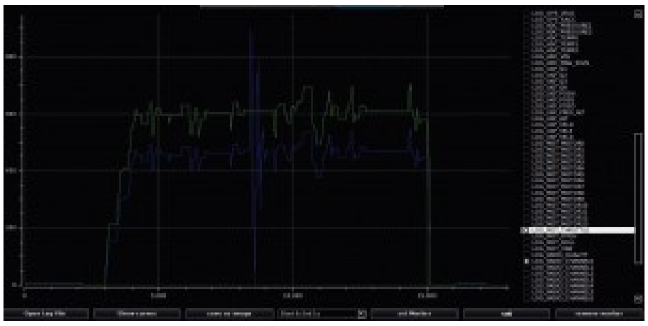
To prevent that any throttle input will cause a change in vertical position, a dead band is present around the throttle center position.
A deadband (sometimes called a neutral zone) is an area of a signal range or band where no action occurs.
The deadband is a settable parameter.

When entering an altitude controlled mode, the current throttle will be used as a baseline but in most cases that is around the center position.
Make sure that the copter will hover around that center position, if you need a lot more throttle to hover the copter you should change the physical powers like bigger props (diameter increase) or increase the lipo cell count.

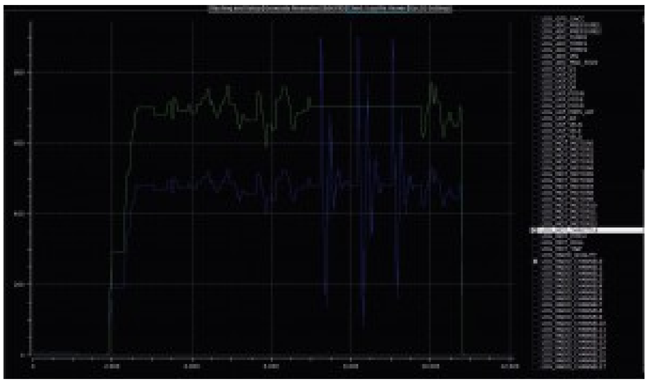
Warning: You cannot land and turn off the motors in this mode, since the throttle stick now controls the altitude, not the motors directly. Switch to Manual Mode to take control, land and disarm the copter.

The following is the process to calculate and adjust the FACTOR_THROTTLE parameter to set your Manual hover at mid-stick.

View a LOG file with a good level hover and take the value LOG_MOT_THROTTLE from the graph. If it’s correct it would be around 700.



The formula Corrected Factor Throttle = Current Factor Throttle * Value read from graph / 700. If the value read from the graph was 700 it would make no change. If your hovering throttle value is low like in the pictures then the throttle factor was too high and needed to be reduced by the proportion of 600/700



Return to home

Return to home

Like most available flight controllers with GPS, the AutoQuad has a **return to home** function.

The home position is determined on the first position hold or when the *set home position* switch is utilized. The home position stores the longitude and altitude position **and** the current altitude.
Warning! When you set the home point on the ground the copter will hit it when you engage RTH!

If the first position hold was engaged on +2 meters, returning to home will be done at +2 meters. If you fly in an area with a lot of vertical obstacles it is probably best to set the home position on an altitude above the obstacles.

Testing

1. Arm the AutoQuad
2. Wait for a solid GPS lock (blue led lid solid)
3. Take off and fly
4. Enter position hold on the required altitude or hit the *set home position* switch.
5. Fly a couple of meters away from that point.
6. Enter position hold (or keep still)
7. Switch to *return to home*

The copter should now fly to the home position in a straight line with a speed set in the parameters (default 6 m/s), slowing down on approach and ending up at the altitude that was recorded during the *set home position*

Heading-Free DVH Mode

Heading-Free (aka. “CareFree”) DVH Mode

Introduction

“Heading-Free” (HF), or “CareFree” as it is sometimes referred to in other flight controllers, is the ability to control the flight direction of a multi-rotor (MR) craft without any regard for where the front of the vehicle is actually pointing.

For example, typically when you push the pitch stick forward, the MR will lean in the direction it is facing, which becomes the forward direction of movement. But in HF mode, moving the pitch stick forward will always fly the MR in the same direction, regardless of where the front is currently pointing. The direction in which the MR will move based on stick inputs can be set manually, or by default is related to North/South/East/West orientation (so pushing stick forward will always fly North, left will fly West, etc).

The reason we call it it Heading-Free *DVH* mode is because the feature can only be active when in position hold, or more specifically while Dynamic Velocity Hold (DVH) is active. DVH is active when moving the direction stick while in PH mode, and is what gives you the ability to still move the MR around while in PH (similar to Vertical Altitude Override).

Heading-free mode is available in AQ 6.7 versions starting at r192/build 1317 and higher.

Usage Scenarios

Of course HF could be used for any purpose one may dream up, including just for fun, but a few specific scenarios are most often discussed.

- **Aerial photography/video** – The purpose here is to avoid a lot of the complexity involved with allowing full 360 degree panning of a camera/gimbal, by the pilot and/or a separate camera operator. The main difficulty with such setups is landing gear which needs to be either retracted or rotated with the whole gimbal assembly. Secondary can be things like tangled wires and limited servo travel. By decoupling the heading of the aircraft from the movement direction (HF mode), yaw control (rudder) can be used to pan the camera freely without affecting the pilots ability to maneuver. An independent camera operator could be provided with control over yaw using either a “buddy box” RC transmitter setup, or directly from a channel on a separate RC receiver on-board the aircraft (in the latter case the channel would need to be mixed into the final radio output to the AQ, eg. using a PPM or SBUS adapter).
- ~~**Beginner pilots**~~ – Yep, this is crossed out on purpose. It may be tempting to get up and flying “quickly” using HF mode w/out going through the major learning curve of flying a heli/multicopter. Don’t do it. If you get used to flying like this, you will only damage your ability to learn proper orientation and control. Sooner or later you will need that ability, most likely sooner.
- **“Non-pilot” operators** – What? Well, under the proper circumstances, and under

a pilots supervision, one may want to hand control of a well-behaved AQ to someone who is less adept at flying, or in fact has never flown anything before. (And is not training to become a pilot. 😉) AQs extremely precise position/altitude hold and DVH mode, combined with HF, does make it possible for anyone with decent hand/eye coordination to control a multicopter. Ideally this would be done with a "buddy box" RC transmitter setup, or in some other manner so the pilot can quickly resume control should anything go wrong. In the future, automation modes could be improved to provide assistance to non-pilots (eg. automatic takeoff and landing, emergency recovery, etc).

Definitions

When discussing HF operation, it is important to first define some terms. Some of this will (hopefully) make more sense as you read the rest of this page, but this covers the basics:

- **Reference Heading** – Defines which way is "forward" in HF mode. Since the front of the craft is no longer used as a reference point, we need to know what the new reference is. By default, the reference heading is North, as mentioned above. But it can also be set specifically to any desired heading. A reference heading can be set two different ways, based on either:
    a. The current direction the front of the craft is facing.
    b. The current bearing from home position to the craft. This way "back" direction becomes back towards home, forward becomes away from home, etc.
- **HF Locked Mode** – This is the "traditional" way to fly "care free" (eg. in MultiWii, others). When in locked mode, the reference heading does not change from whatever it was originally set to (or North, if it wasn't set). Basically this is like the "on" switch for HF mode, and is the most basic mode of operation. If the pilot stays facing in roughly the same direction, it should be intutitive to control directional movement w/out worrying about the actual craft heading.
- **HF Dynamic Mode** – In this mode the reference heading is continuously adjusted. This can be useful when the reference heading is set based on home position, as described in point (b), above. For example, someone standing near home position could control the flight direction without ever worrying about the craft orientation OR the original reference heading, as long as they kept facing the vehicle.

Setup

To use HF mode, you need an available radio channel and a control on your transmitter. Ideally this would be a 3-position switch, although you could also make-do with mixing two 2-position switches, using a slider, or using only one 2-position switch (if you can set its end-points). The control positions are preset in the AQ firmware, so you need to set up your radio to match.

Set HF control channel

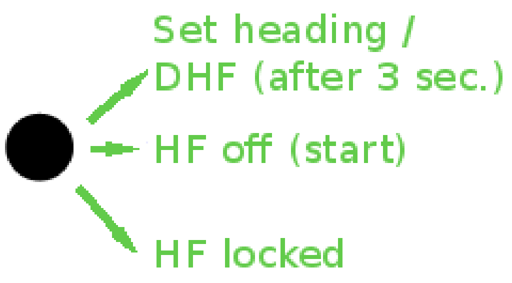
In the AQ firmware, you can set which channel to use to control HF via the parameter NAV_HDFRE_CHAN (or do it via QGC in the Controls Setup area). By default this value is zero, which means HF can not be used. Set this to the actual radio channel number you want, but **make sure it does not conflict with any other vital control channels** (throttle/attitude/flight mode/home actions). The QGC interface will validate your choice for you.

HF control positions

- **Channel to middle** (radio value ~0) – HF is off. Forward movement is in direction of craft heading, like normal.
- **Channel to low** (radio value < -250) – HF Locked Mode is on. Forward movement is in the direction of the previously set reference heading (or the default). Locked mode can only be enabled when the AQ is armed.
- **Channel to high** (radio value > 250) – This position has 2 modes of operation:
    1. **Initial activation** – Set reference heading. The heading used is based on a number of factors, such as AQ armed status, flight status, and location, as described in more detail below.
    2. **After 3 seconds** (from initial activation) – Turn on Dynamic mode. This also turns HF on, but in this case the reference heading is continuously updated (see *Flying in Dynamic Mode*, below). Dynamic mode can only be turned on when the AQ is armed.

<u>While the AQ is disarmed, you can set the reference heading using either the high or low channel position.</u> This allows someone to use one 2-position switch to control HF locked mode on/off while retaining the ability to set a reference heading (but not enter dynamic mode or set reference while in flight). The radio would need to be set up so that the switch toggles the channel value between mid and low positions.

<u>AQ will not arm unless the HF Control is at middle (off) position.</u>



Operation

Setting the reference heading

If you do not specifically set a reference heading for HF, the default is North. That’s great if you are flying/facing to the North, but this may not always be the case. Additionally, you may want to re-set the reference heading while flying, for example if you change the direction you’re facing and the controls become no longer intuitive.

You always set the reference heading by toggling the HF control channel to high position (with one exception as mentioned above, while disarmed, heading can also be set by toggling the control to low position). Which heading is used depends on the current armed/flying state and, in some cases, the location of the AQ relative to home position. Here are the various scenarios and their effects:

- **While disarmed** – The actual craft heading (direction the front is facing) is used.
- **Armed and not flying** – The actual craft heading (direction the front is facing) is used.
- **Flying and within 2 meters of home position** – Again, the actual craft heading (direction the front is facing) is used.
- **Flying and beyond 2 meters of home position** – In this case, the bearing from home position to current craft position is used. In other words, “forward” becomes away from home, “back” becomes towards home, and left/right are likewise relative to home position.

Flying

If you’ve kept up with us so far, you hopefully know everything you need to at this point. Here is an outline of the general procedure to help put it all together. This is just one possible scenario… ultimately it’s up to you how you want to use HF mode.

Boot up the AQ, and set your initial reference heading based on where the craft is pointing (either before or after arming, but before takeoff, as described above). (Due to current AQ behavior, it is best to wait for a GPS lock before doing this.) You will probably want to point the craft directly away from you, towards the direction you will be flying in, so that the stick directions are intuitive once you’re flying in HF mode (“back” is towards you, “left” is to your left, etc.).

Take off and enter position hold, as usual. Keep in mind that home position is automatically set the first time you enter PH (or of course you can specify a home point by using the home action switch). Try to stay facing in your initial direction, with the craft roughly in front of you.

Once you’re satisfied with PH performance, you can choose to turn on HF Locked Mode at any time by toggling the control channel to low position. Your control directions are now based on the reference heading you set before takeoff. If you’re still facing in the same direction, this should be very easy to control. Try it out with small movements to make sure you understand which direction is where, and that everything is working as you expected. Keep the AQ close initially so you can always turn off HF and regain normal control w/out loosing orientation.

Now let’s say you change the direction you’re flying in and the directional controls no longer make sense. You have several options… if you’re standing near home position, then setting reference heading based on bearing from home position works very well and is quick to do (so, assuming the AQ is > 2m from home, just toggle the control channel to high and then back to low). However if you’re not standing close to home, this will probably make the direction controls more confusing. In this case, bring the AQ back to home position, point it in the direction you want, and toggle the control channel to high to set the new reference heading based on craft orientation (not bearing from home). Alternatively you can also land the AQ, anywhere, and then set the reference heading (which is then always based on current craft heading). If you only have 2-position control set up, then you must land somewhere first, and disarm, before setting the new reference.

Flying in Dynamic Mode

The only difference between “dynamic” mode and “locked” mode is that in the former, the reference heading is continuously re-set while you are flying. The reference heading is set based on the same criteria as described above, meaning it depends on how far from home position the AQ is. Obviously then, dynamic mode is only really useful/interesting when flying more than 2 meters away from home position. Also, it is only really intuitive if the pilot is standing at, or near, home position (or is able to orient their thinking appropriately 😃).

Given these 2 conditions are met, dynamic mode can be very intuitive to fly, and useful if you constantly need to rotate yourself to face the craft (like when flying around you vs. in one general direction). So regardless of which way you turn or where the craft is in relation to you, pulling the pitch stick back will always bring the AQ towards you (or rather the home position). Forward will always send it away from you of course.

Holding the pitch stick left or right in dynamic mode gets a little more interesting though! Since these directions are always relative to home position, they are re-calculated once for every 1 degree in change of bearing. This means those directions will not make a straight line, but will make an arc around home position. This could either be a cool feature, or annoying, depending on what you are trying to achieve. For flying perfectly straight lines in any direction, it won’t work (use Locked Mode for that). On the other hand, you can hold left/right stick and fly in a circle around home position (roughly… it comes out more like a spiral). Of course you can always give it some forward movement at the same time in order to keep the line straighter, but it’s tricky.

So to sum it up, the main attraction of dynamic mode is that it easy to maintain intuitive control direction without having to remain facing and flying in one general direction. But it is only useful if the pilot is standing near home position, and when the results aren’t negatively affected by the arcing effect during left/right flight.

Points to remember

- If the HF control channel parameter is non-zero, AutoQuad cannot be armed unless that radio channel’s value is at around middle position (~0).
- Position hold, DVH, and subsequently HF, all depend on a 3D GPS lock. It is currently not possibly to utilize HF mode indoors, for example. Note that once a valid GPS lock is achieved, the AQ will typically NOT drop out of PH mode even if the GPS signal degrades. This will keep DVH/HF active, but of course might have negative effects on PH performance and the ability of HF to calculate proper bearing to home position.
- HF Dynamic mode is NOT ACTIVE while the Home Action switch is turned high/low (in other words when it is at Set Home or RTH positions). We recommend only using that switch as a momentary contact, moving it back to middle position right after activating Set Home or RTH.
- When flying in Dynamic HF mode and moving to within 2 meters of home position, the reference heading may change unexpectedly (will now be based on craft heading instead of bearing from home). This is generally harmless, although could be a little surprising. Avoid high-speed approaches to home position while in

dynamic mode, at least until you get an idea of how it will behave.

- Tip: To quickly re-set reference heading based on actual craft heading (not bearing from home) w/out landing, first engage RTH, wait for the AQ to arrive, yaw it to the desired direction, and set the new reference heading by toggling the switch to high position and back to off or locked mode.

Mission flight

Mission flight AutoQuad is born to perform mission flights. But first a statement about safety:

The AutoQuad is able to perform autonomous missions.Please observe (local) regulations like maximum altitude, safe distances to populated areas and traffic. Never loose sight of the aircraft, use a spotter when flying (autonomous) FPV missions. Never plan missions directly over attended audience. In general, use common sense when planning a mission.There are (currently) no restrictions build into the system and being opensource the code could easily removed anyway

AutoQuad is capable of performing the following mission commands:

Command	Minimum firmware	Explanation
take off	6.6	take off to a settable height above ground with settable vertical speed, perform a 'loiter' for a settable time
land	6.6	land at GPS position with settable descend speed
waypoint	6.1	Go-To GPS location, settable altitude, settable horizontal speed, settable wait time
return to home	6.1	Execute a return to the stored home position at stored altitude with pre-defined horizontal speed
orbit	6.6	Execute an orbital flight around a GPS point, settable radius, settable horizontal (orbital) speed, settable altitude

The orbit command has additional features; it will keep the heading (front of multirotor) pointed to the center and if the altitude of the point of interest and altitude of the AutoQuad is different, it will keep the camera gimbal tilted to the altitude of point of interest. This will provide a great way to circle around an object while keeping the camera pointed to the object.

[Link to embedded object]

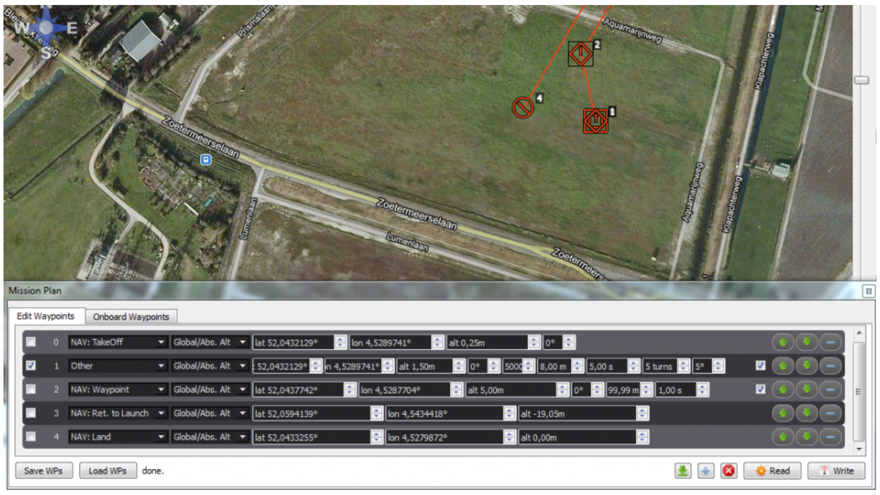
Missions can be created, uploaded, retrieved and stored offline using the Ground control station(s). It is even possible to load or store them on the uSD card so missions can be created at home.

There are no limits in the mission waypoints, no altitude or position distances. Currently there is a 25 waypoint limit in the code because we simple don't know yet the flash limitations for storing complex missions. The limit can obviously be changed in the firmware code for developers.

Create a mission

Missions can be created using the GroundControl station windows or the Android GCS. Both the GCS types will allow to place a waypoint somewhere on the map, select the type and upload the mission plan to the AutoQuad flightcontroller. In this example we concentrate on the windows GCS.

First connect to the AutoQuad flightcontroller using the FTDI interface. Obviously connecting with a bluetooth or Xbee device to upload/retrieve the mission is possible.



☐	0	NAV: TakeOff	Global/Abs. Alt	lat 52,0432129°	lon 4,5289741°	alt 0,25m	0°					
✓	1	Other	Global/Abs. Alt	52,0432129°	n 4,5289741°	alt 1,50m	0°	5000	8,00 m	5,00 s	5 turns	5° ✓
☐	2	NAV: Waypoint	Global/Abs. Alt	lat 52,0437742°	lon 4,5287704°	alt 5,00m	0°	99,99 m	1,00 s			✓
☐	3	NAV: Ret. to Launch	Global/Abs. Alt	lat 52,0594139°	lon 4,5434418°	alt -19,05m						
☐	4	NAV: Land	Global/Abs. Alt	lat 52,0433255°	lon 4,5279872°	alt 0,00m						

Place a waypoint on the map using the mouse and select the type in the menu. You can move waypoints sequence in the list, edit them or delete them.

Forum discussion: Step by Step mission planning via PC QGC

Upload / retrieve a mission

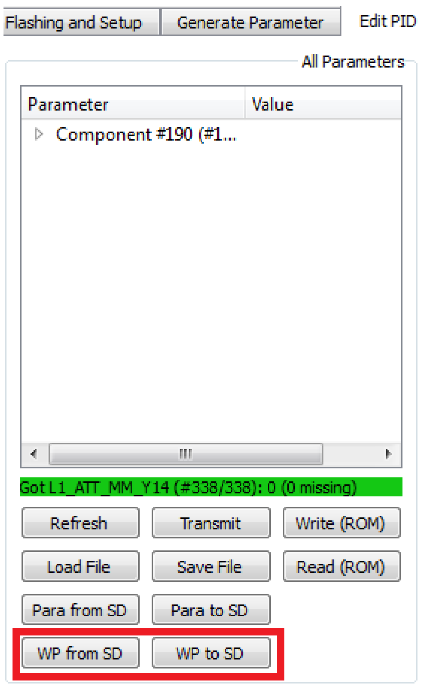
Once satisfied you can use the write button to upload the mission to the AutoQuad flightcontroller. The buttons Save/Load WP will create or load an offline file with the created mission. Retrieving a stored mission can be done with the read button.

A mission is store in volatile memory, it will be lost after a power cycle or reboot.

Store on uSD

Using the Groundcontrol station(s) an option is available to store a just created mission on the AutoQuad onboard uSD. For that a uSD must be placed in the AutoQuad flight controller. Inserting the card will trigger the log function but an store command from the ground control stations will cause the waypoint list to be stored on the uSD card. This is also available on the Android GCS in the mission planner menu.

For the windows GCS select the engineer tab, AutoQuad widget.



Here you will find the 'All parameter widget' with two buttons: WP from SD and WP to SD for loading the missionplan or saving the mission plan to the uSD card.

Execute the mission

Most of the missions will be engaged while in flight, either a hover or position hold. The mission is a sequential set of commands and will simply start at command 1. After all commands in the queue are done the mission will transition into position hold. Start a mission by switching to the mission mode (aux 2) and ***center throttle stick*** . Below or above center will command an ascend or descend of the multirotor. Don't worry about the precise midpoint, a default deadband around center of 35us will give you a neutral zone around center where nothing will happen. If that is to narrow, you can increase the deadband parameter in the configuration.

Failsafe events during mission: during the execution of the mission, a radio loss signal is ignored until the last mission command is executed.

Abort an mission

At any time during the mission you can abort the execution by switching to manual mode. At that time the radio fail safe event is also active again.

Take off command

Once stored in the AQ as a takeoff command, a special sequence is needed to execute this command and the next mission commands. Start with the multirotor on the ground, arm the AutoQuad, switch to mission mode (AUX2 – mission mode) and.. this is very important, center the throttle stick. Without centering, the navigation routine will think it is in descending command, every mission should be flown with the throttle in center position. Because the throttle commands altitude change in navigation modes, a throttle not in center will command ascending or descending.

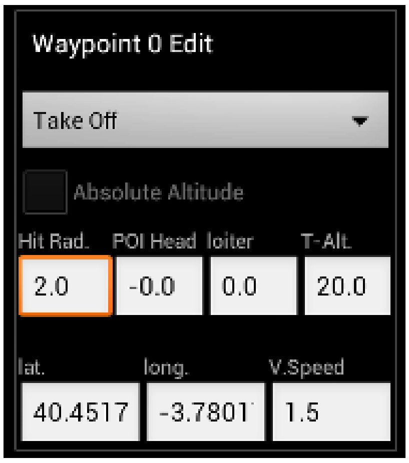
[Link to embedded object]

Parameters in mission commands

	Waypoint					
Number	P1	P2	P3	P4	P5	P6
Description	Latitude	Longitude	Altitude	Delay	Hit Radius	Yaw Angle
Units				ms	meter	degree
	Go to Home					
Number	P1	P2	P3	P4	P5	P6
Description	Free	Free	Free	loiter Time	Hit Radius	Free
Units				ms	meter	
	TAKE OFF					
Number	P1	P2	P3	P4	P5	P6
Description	MaxHorizSpeed	Hit Radius	Poi Heading	loiterTime	Target Alt	Free
Units	m/sec	meter	degree	ms	meter (absolut)	
	Land					
Number	P1	P2	P3	P4	P5	P6
Description	MaxHorizSpeed	Free	Free	Free	Free	Free
Units	m/sec					
	Orbit					
Number	P1	P2	P3	P4	P5	P6
Description	Latitude	Longitude	target Altitude	MaxHorizSpeed	Hit Radius	loiterTime
Units			meter	m/sec	meter	ms

Troubleshooting and tips for autonomous flights

Problem: when sending the copter on a mission with several waypoints it seems to stop after the first WP is reached.
Solution: If you selected a too small value for the target radius (Hit Rad.) the aircraft tries to reach the exact point. Especially under windy conditions it might take long until it hits the position precisely. Set the “Hit Rad.” to 2m or more like shown in the image below. The delay should disappear then and the copter will always continue its flight to the next WP.


Waypoint 0 Edit

Take Off

Absolute Altitude

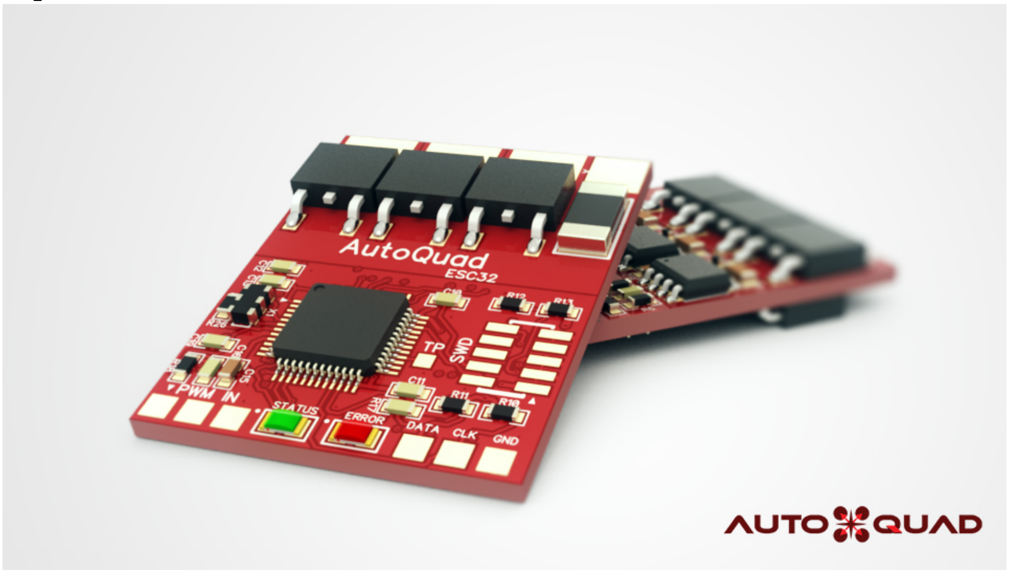
Hit Rad.	POI Head	loiter	T-Alt.
2.0	-0.0	0.0	20.0

lat.	long.	V.Speed
40.4517	-3.7801	1.5

Problem: the copter doesn’t turn towards the waypoint on a mission flight but keeps the heading i.e. flying sidewards.
Solution: Select a negative value in the Yaw box, like -1.0

Part 5 - ESC32
ESC32

The AutoQuad ESC32 - a yet unseen electronic speed controller



Why the need for specialized ESCs:

Most standard ESCs are not designed for multirotor applications. Most ESCs are build and configured with an airplane application in mind, they are programmed to increase and decrease the throttle command towards the motor in a gentle way for obvious reasons. A motor fast accelerating and decelerating in an airplane or helicopter would either break the gears or put great stress on the prop and airframe.
A multirotor is different. We would rather have a total linear curve and able to change the motor speed, and thus the thrust, as fast as possible to achieve a perfectly stable platform.

While the current ESCs used in most multrotor applications work considerably well, there is a lot of room for improvement. Using the same ESC but loading it with dedicated optimized firmware will increase the handling and stability of your multirotor greatly.

AutoQuad and most other flight controllers will benefit from ESCs that can operate at PWM frequencies of at least 400hz and are stripped from a controlled throttle curve.

Thats where the ESC32 comes in, a superior 32 bits ESC with a lot of interfaces and room for future enhancements.

AutoQuad ESC32 firmware and most of the related software is

published under the GNU GPLv3 software license. Please read the license carefully before you apply changes to the code.
More info on the AutoQuad licensing.

Learn more about the ESC32 in our Wiki

Specifications for version 2

- STM32F103 72MHz 32bit ARM
– All N-FET design with gate drivers
– 2S through 5S battery voltage
– Option to power logic side via UART or PWM IN +5v
– CAN transceiver hardware support onboard
– Firmware written completely in C
– Cortex SWD connector pads for real-time debugging
– Communications ports: PWM IN / UART / I2C / CAN Bus
– Communications protocols: PWM IN / CLI / binary / 1-wire / I2C** / CAN**
– 4KHz to 64KHz PWM out
– Current sensing / limiting with real shunt resistor
– Virtual current limiter
– Regenerative braking (experimental)
– Closed loop control modes
- Lot of available RAM / FLASH for experimentation and development

** I2C and CAN drivers have not yet been written

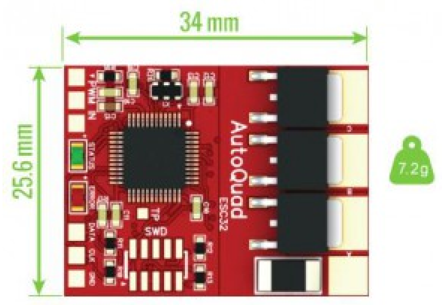
More info on it’s own Wiki pages.

3D animation: use your mouse to turn the board 360deg. left & right

ESC32 hardware setup

Assembly of ESC32:

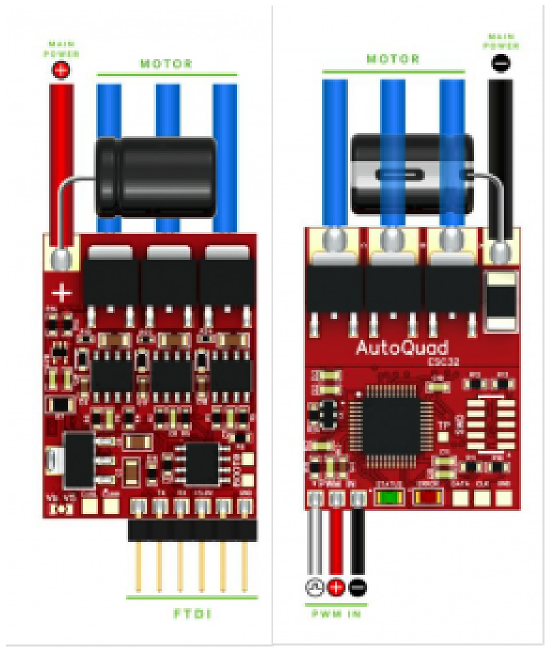
Autoquad ESC32 comes as a “SMT-preassembled” kit. This means the enduser has add all connections themselves. We did this to keep cost low, and to allow you to configure ESC32 for your exact needs.



Kit contents:

- ESC32, SMT-preassembled and factory tested board.
- 25 cm servo wire with open end.
- Input power wires
- 330uF/35V lowESR capacitor

Its up to you to configure the wiring so it fits your need and frame. We have shown one way to do it here:



Input capacitor:

The input capacitor is a critical component for ESC operation. The Supplied 330uF/35V capacitor in the kit is safe for “normal” use on up to 4S with continuous currents below 10A. Increase capacitance (add another in parallel or increase size) if:

- Running long wires from battery to ESC (more than 25CM)
- Running high continous currents (remember cooling)
- Using 5S (use 50V capacitor for 5S)

Input capacitor should be Low ESR type always!

The Input capacitor(s) is soldered directly to the + and – pads on the ESC.

Observe the input capacitor polarity! The MINUS (-) mark on the capacitor must be connected to the ‘-’ terminal of the ESC32. Failing this can blow up the capacitor and damage the ESC32!

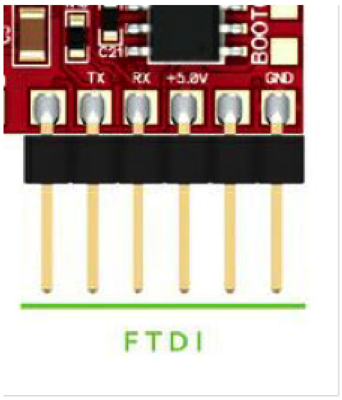
Make sure the two capacitor legs are as short as possible and that they cannot be short-circuited to each other – consider using a small piece of shrink-wrap on each leg. Check for solder bridges and shorts across the capacitor legs.

Power connection:

There are multiple ways the main input wires can be connected. We leave it up to you to determine what works best for you. Make sure the there is no shorts across the main power input or the motor phases!

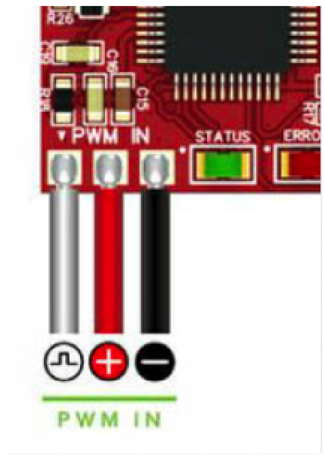
UART connection

The UART connector matches a standard FDTI style 6-pin connector.
It is used for flashing firmware and connecting to GUI/CLI or running the ESC over the UART interface.



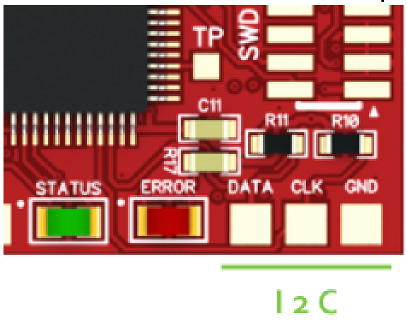
PWM connection

Solder the PWM servo wire to the PWM pads or add header pins:



i2c connection

The ESC32 offers a I2C bus connection for future use. Support for MK I2C control will be released as soon as we had a chance to test it properly – talented people are working on this – a beta release is expected soon.

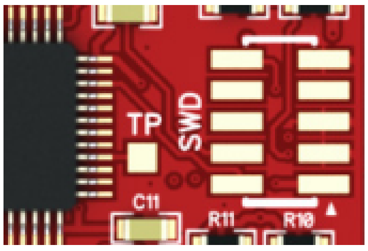


CAN-connection (Optional)

ESC32 can be fitted with an optional CAN transceiver (Texas Instruments SNHVD232) to provide CAN-bus control and bidirectional communication at up to 1mbit/s. Currently CAN is not supported in the Firmware. It is also a work in progress, and we will release a driver for it when we had more time to implement and test CAN control.

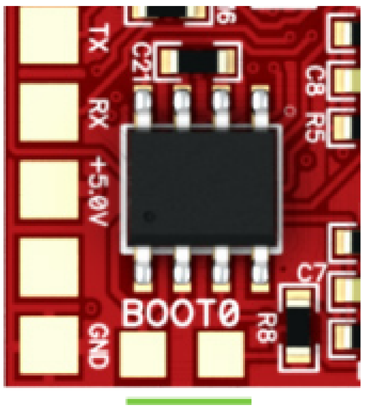
Cortex SWD connection. (J4)

A standard 10-pin SMT cortex SWD connector (Samtec FTSH-105-01-F-DV) can be soldered to the board for realtime debugging of ESC32.



Boot jumper

The boot jumper is used to place the STM32 bootloader in flash mode for uploading new firmware. Short these two pads during powerup to activate the Bootloader and place ESC32 in flash mode *(Hint: you can use a pair of tweezers, if you are bit handy – you dont need to keep the pads shortened after the ESC has been powered up)*



Select power for logic side

The ESC32 has two options for powering the logic side:

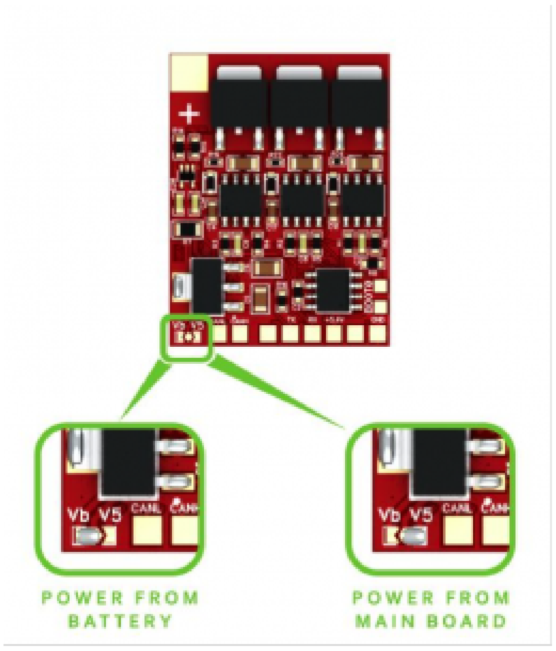
• On-board 3.3V LDO powered by main battery (Vb)
• 5V coming from UART or PWM interface (V5)

In most cases (especially when using LiPo batteries larger than 3S) it is recommended to power the ESC MCU with 5V from the PWM or UART interface – either from an external regulator or a UBEC hooked up to the servo rail on your FC or receiver.

Selecting 5V, also means you can power up the logic side separately when flashing firmware or altering settings over 1-wire or UART interfaces, without the need to power up the motors.

When using the 5V option to you must always **DISCONNECT MAIN INPUT FIRST** before disconnecting the 5V power to PWM or UART interface – disconnecting 5V while the main is still powered can damage the ESC.

To select the power source for MCU, locate the small solder jumper in the corner on the power side. It’s a simple 3-way solder jumper and works by con-necting the middle pad to either the “Vb” or “V5” pads.



Final checks before powering up

All ESC´s are tested in factory, before being released. But its a delicate piece of electronics running high currents, so you should Inspect your soldering carefully and make sure there is no shorts or solder splashes before powering up the first time. Measure with multimeter (if you have one) for shorts across the main power, PWM and the 3 motor phases.

First time you power up the ESC, we recommend to have the 5V option selected and hook it up to the AutoQuad widget in Qgroundcontrol or CLI via the UART interface without main power connected.

Alternatively you can hook ESC32 to a simple servo tester or receiver with 5V on the PWM interface. Don't hook main power yet!

Note: Your servo tester needs to support >50Hz! Cheap/old units only support up to 40Hz.
When using your receiver, trim your throttle channel down on your transmitter to make the ESC32 initialize and arm.
If the ESC gets 5V and a PWM pulse width of at least 750 microseconds, the green LED will light up showing you that the ESC is receiving a valid signal from your FC, RX or servo tester.

You can now hook a power source to main power and check if the motor runs when you increase throttle. (Its a very good idea to use a small, limited power source like a 9V/500ma DC adapter to perform the first tests. This will secure nothing gets burned in case you have a short in your soldering somewhere. If you hook directly to a high current source (like a lipo battery) and there is a short, you can fry both ESC, battery and motor very quickly.

Inspect your soldering carefully and test the ESC on a limited supply before

hooking it to a big battery.

ESC32 is now ready to run as a standard PWM ESC with very high motor refresh rate and a 1000-1950us throttle range with motor start at 1100uS. As such it can now be mounted to your favorite multi and flown right away.

Current and cooling considerations

How much current you can actually draw from ESC32, depends on how good they are cooled and where they are mounted.

Our general recommendation is to mount the ESC32 under or near the motors where they get cooled by the prop wash. Continous current drawings of 20A result in a slightly warm ESC. At under 10A per motor, cooling considerations are not needed. The gate-driven design means that the ESC´s wont warm up considerably when run at low currents.
The upper limit is probably upwards of 50A, but that will require extremely good cooling.
The factory default for the current limiter in the firmware is 20A continuously. Going beyond that limit will require attention to proper cooling and temperature of the ESC.

Bear in mind that most “normal” setups under 3kg will stay well under 20A continuously per motor in normal flight conditions and hover (or something is very wrong with your choice of motors and props).

Always test your individual setup and if you see signs of the ESC heating up it means you should add cooling. The better you cool your ESC´s, the more efficient they will also be so! In theory, they should continue to operate up to around 70-80 celsius, but they will age faster when run at high temps!

The best strategy keep ESC´s cool in a multirotor, is to place them in the wash from the rotors (Ideally right under the motors). In frames where that is not possible because of wiring considerations or if running high currents, heatsinks should be applied to the FETS on both sides. In general its not good practice to place ESC´s in tight places without any airflow, especially if running high currents!

LED signals

red flashing	no valid PWM signal (<750us or > 2100us) prop strike over current
green	ready / armed
off	running mode

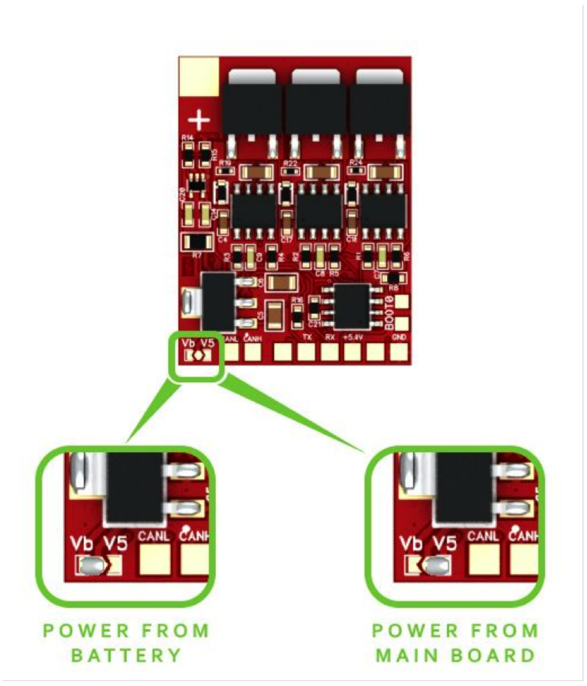
Please be very careful running a calibration!
Your motors will perform certain patterns up to full power. Make sure to run these tests only in a secured environment!
Only use it in closed rooms or boxes where propellors can fly off without hurting anyone or damaging anything!

Note: You can download these instructions as PDF file from our Download section.

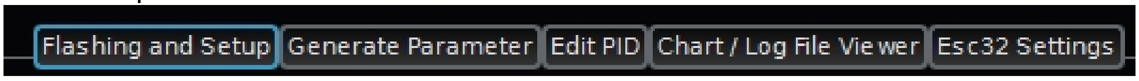
ESC32 firmware flashing

ESC32 Flashing Quick Guide

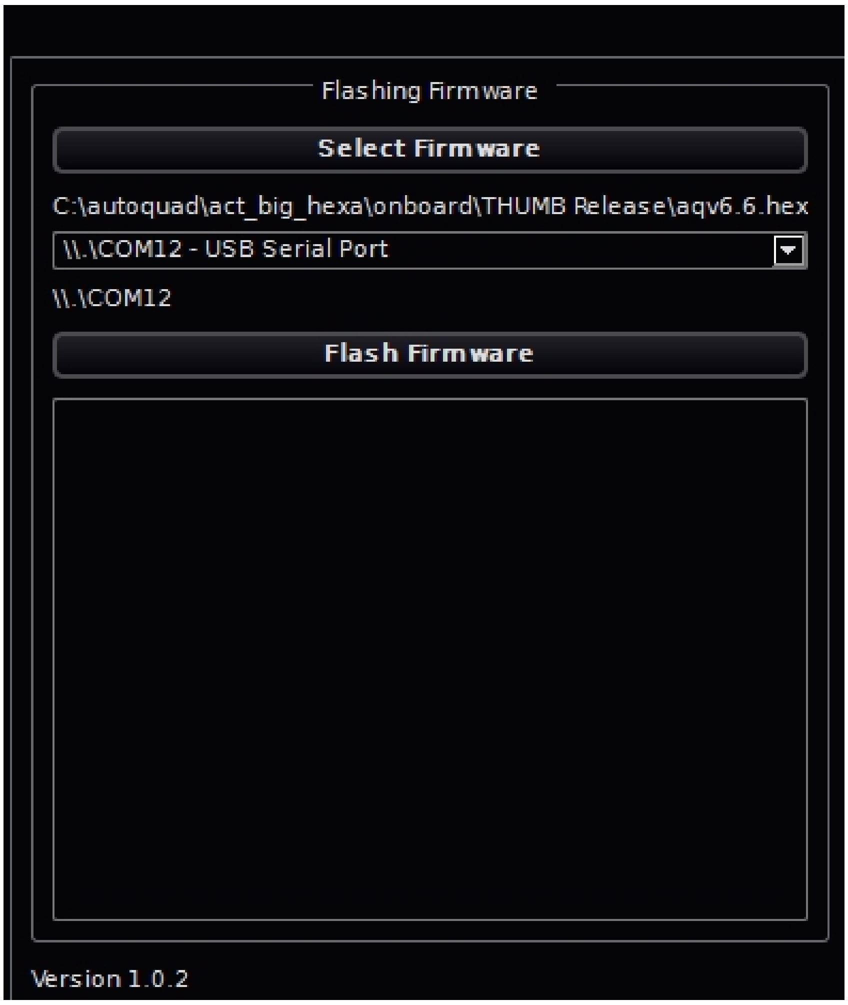
Preparations:
Make sure you have a correctly compiled *.hex file at hand.
ESC32 needs to have the headers for the FTDI/USB adapter and boot jumper soldered – also don’t forget to set the solder jumper for the power supply (either 5V or Vb)
Connect your FTDI/USB adapter and the boot jumper, then power up the ESC. Read more…



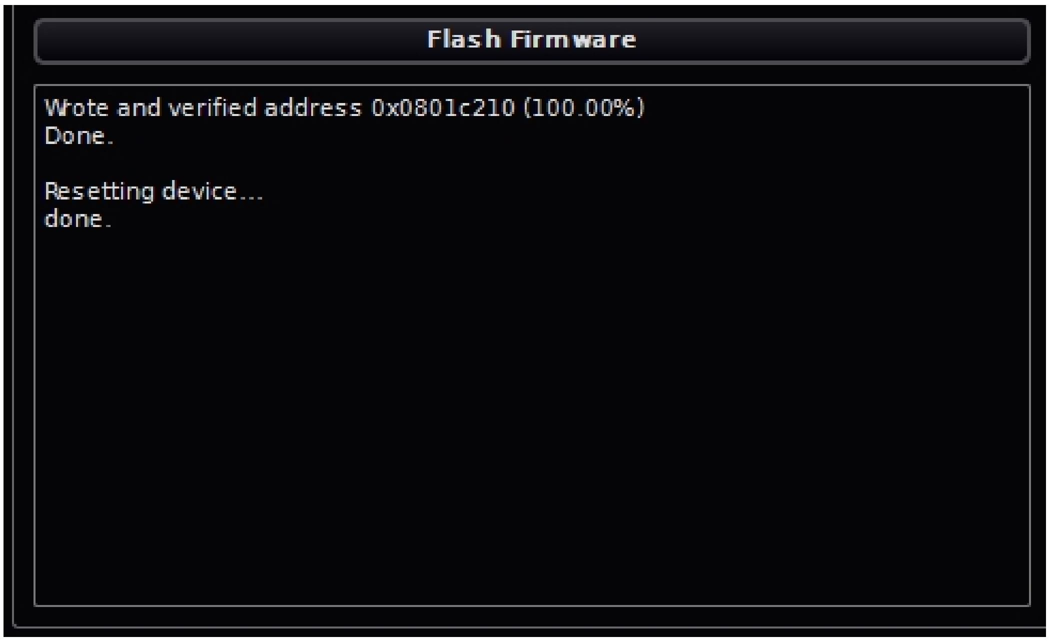
In the AQ QGC select the “Engineer” tab with the AutoQuad Main Widget. Select “Flashing and Setup”



In the “Flashing Firmware” part of the screen select your firmware you want to flash, select the appropriate COM port and click on “Flash Firmware”.


Flashing Firmware
Select Firmware
C:\autoquad\act_big_hexa\onboard\THUMB Release\aqv6.6.hex
\\.\COM12 - USB Serial Port
\\.\COM12
Flash Firmware
Version 1.0.2

In the message field above the Version number you should see the process of the upload, followed by a confirmation that flashing was performed successfully and the ESC will be rebooted.



```
Wrote and verified address 0x0801c210 (100.00%)
Done.

Resetting device...
done.
```

To make sure everything went fine, disconnect the ESC from the power source and the FTDI/USB, close and reopen the QGC and connect your ESC again.
Open the “ESC32 Settings” tab, click on “Connect” and see if all values and settings are fine.

You can now proceed with the ESC32 parameter configuration.

ESC32 easy programming

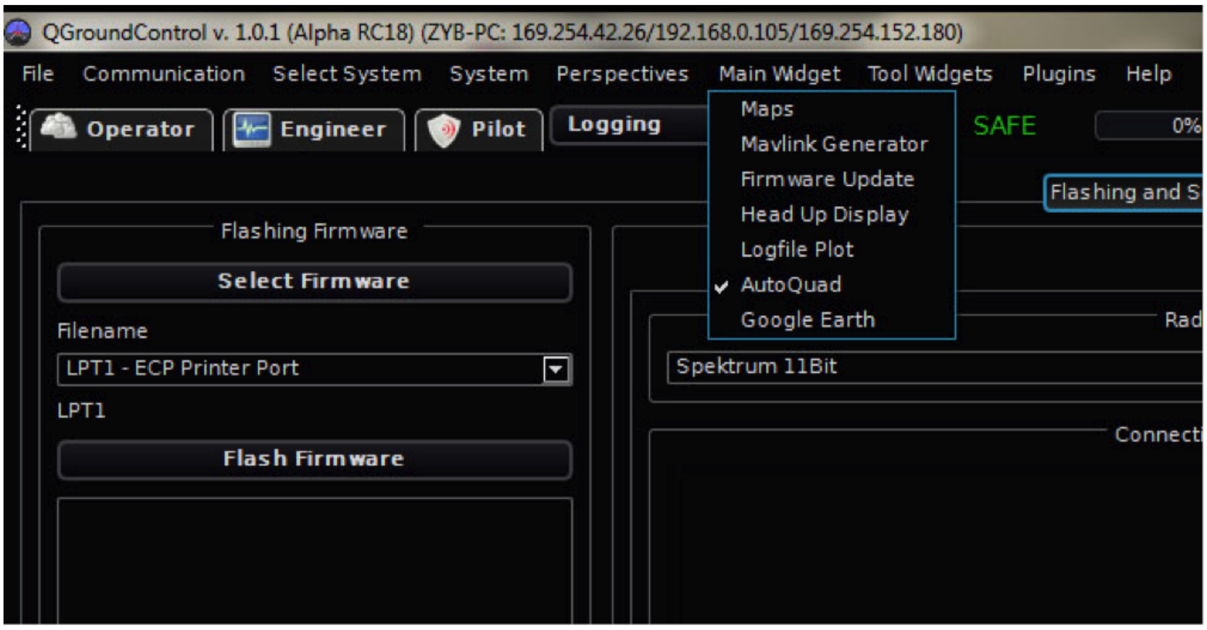
Added : program all attached ESC32′s with settings using the AutoQuad flight controller

ESC32 configuration and modes

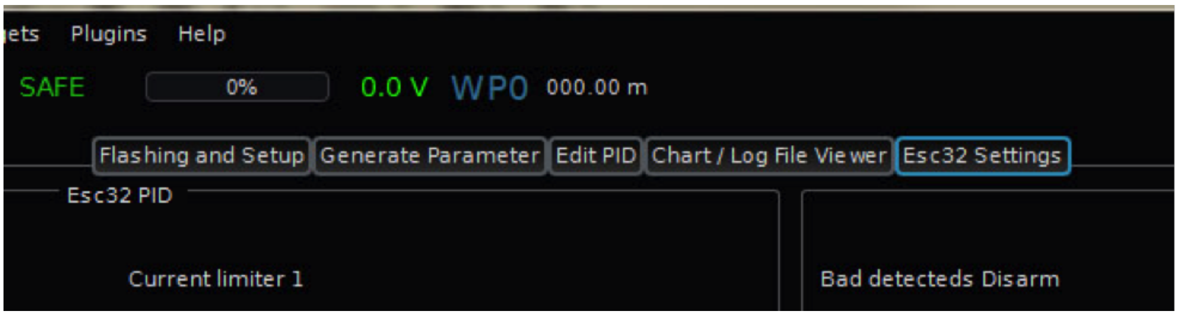
Connect ESC32 to autoquad widget in QGroundControl

The Autoquad version of QGroundControl (QGC) has a widget included to alter settings of ESC32

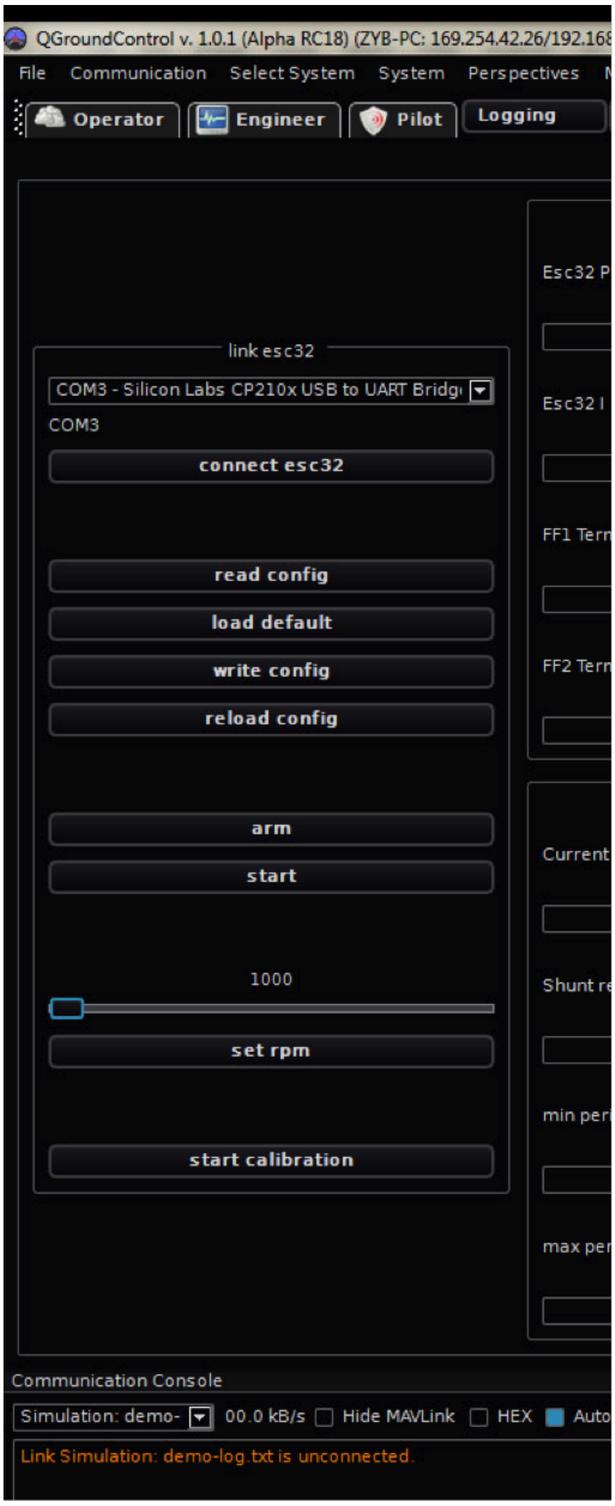
Open QGC, and choose “AutoQuad” from the main widget menu:



Then go to the tab “ESC32 settings”



On the left side of the screen you can see the “Link ESC32″ Tab:



Link Simulation: demo-log.txt is unconnected.

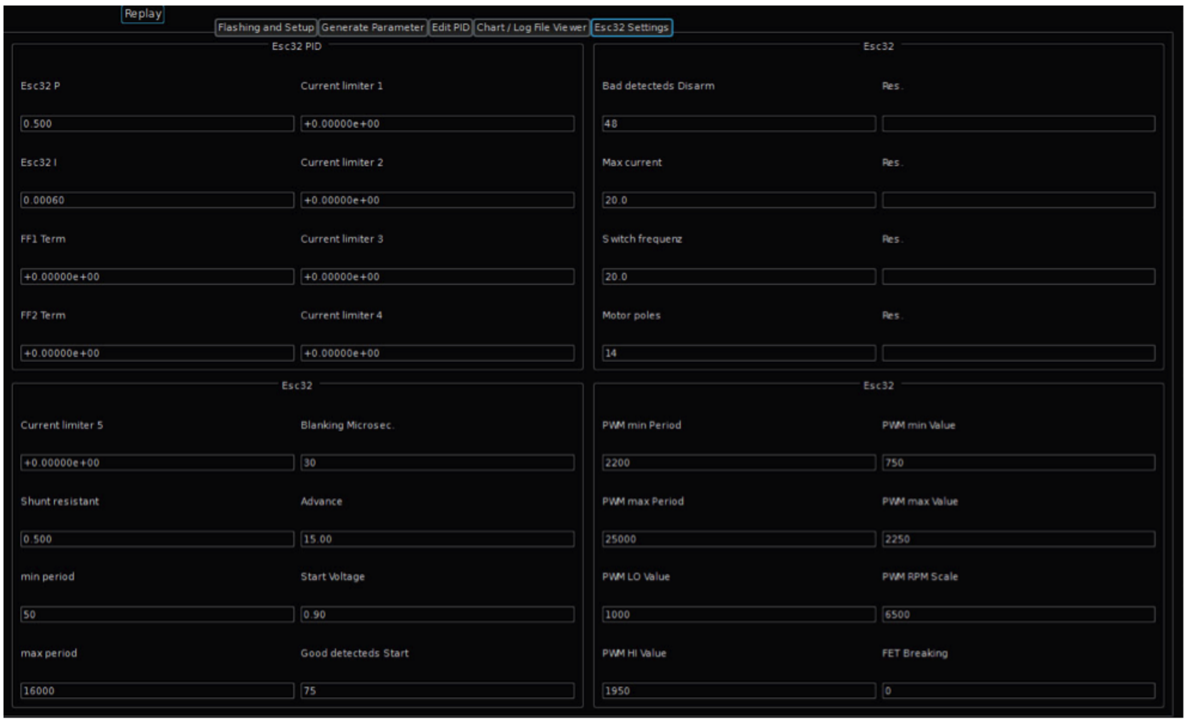
Connect UART adapter to ESC32 and computer and choose the UART adapter you are using and click connect.

When you connect the ESC to UART, it will go into disarmed state (Red LED blinking). Try and click “arm” – the ESC green LED should light up. If there is a main power source and motor hooked to the ESC you can run the motor by clicking “Start” and the motor will run at low duty **(Remove props always when working with your multirotor)**

Now you can alter the parameters in ESC32, but

Make sure you don’t change parameters unless you know how the different parameters affect operation. Certain parameter combinations can be fatal for the ESC and motor! If you want to use the ESC32 with other FCs like MultiWii, APM2 or DJI NAZA please read the comments concerning certain values.

Refer to the User configurable parameters section for more information on what each parameter controls.



Esc32 PID

Esc32 P	0.500
Esc32 I	0.00060
FF1 Term	+0.00000e+00
FF2 Term	+0.00000e+00
Current limiter 1	+0.00000e+00
Current limiter 2	+0.00000e+00
Current limiter 3	+0.00000e+00
Current limiter 4	+0.00000e+00

Esc32

Bad detecteds Disarm	48
Max current	20.0
Switch frequenz	20.0
Motor poles	14
Res.	
Res.	
Res.	
Res.	

Esc32

Current limiter 5	+0.00000e+00
Shunt resistant	0.500
min period	50
max period	16000
Blanking Microsec.	30
Advance	15.00
Start Voltage	0.90
Good detecteds Start	75

Esc32

PWM min Period	2200
PWM max Period	25000
PWM LO Value	1000
PWM HI Value	1950
PWM min Value	750
PWM max Value	2250
PWM RPM Scale	6500
FET Breaking	0

When parameters have been altered, you can upload them to ESC32 by clicking “write config”

User configurable parameters:

Generally, no parameters will need to be changed for normal out of the box PWM operation. However, there are various parameters that can be set. Each parameter is a single numerical value.

Note that some combinations of parameters can cause damage to your motor or ESC. It is recommended to leave most values at their default values unless you are sure of how they impact function and operation

Parameter	Factory Default	Description
STATUP_MODE	0	This is the run mode that the ESC defaults to when first powered on. 0 == normal open loop operation, 1 == closed loop RPM mode, 2 == closed loop thrust mode (not yet implemented)
BAUD_RATE	230400	The UART baud rate. Allowable range 9600 to 921600
PTERM	0.5	The P term for the RPM PI controller
ITERM	0.0006	The I term for the RPM PI controller
FF1TERM	0.0	Feed forward terms used for the RPM controller. Closed loop RPM mode will not function until FF1 & FF2 terms have been set.
FF2TERM	0.0	These values should be calculated using the esc32Cal program with the –r2v option

CL1TERM	0.0	CL1TERM through CL5TERM are used by the virtual current limiter which will not function until they are set.
CL2TERM	0.0	If CL1TERM through CL5TERM is not set, a PI controller will be used for current limiting instead.
CL3TERM	0.0	CL1TERM through CL5TERM can be calculated using the esc32Cal program with the –cl option.
CL4TERM	0.0	
CL5TERM	0.0	
SHUNT_RESISTANCE	0.5	This parameter should **NOT be changed** from the factory default!!! (Value in milliohms)
MIN_PERIOD	50	The minimum commutation period allowed in microseconds
MAX_PERIOD	12000	The maximum commutation period allowed in microseconds
BLANKING_MICROS	30	The number of microseconds to ignore back EMF after a commutation.
ADVANCE	15	The amount of timing advance in electrical degrees. There are 60 electrical degrees in a commutation cycle. This value can be set from 0 to 30 degrees.
START_VOLTAGE	1.1	The amount of voltage presented to the motor during startup. Allowable range is 0.1v to 3.0v
GOOD_DETECTS_START	75	Once started, the number of good, in order zero crossings needed to be detected before the motor is considered to be in the running state.
BAD_DETECTS_DISARM	48	The number of missed zero crossing detects allowed before the ESC considers the motor not to be running at which point will go into the disarmed state.
MAX_CURRENT	20	The maximum amount of current in amps that the ESC will allow. Current is dynamically regulated. **Always set this value low and only increase it if you know what you are doing.**
SWITCH_FREQ	20	The output PWM pulse frequency used to power the motor windings in KHz. Valid range is from 4KHz to 64KHz.
MOTOR_POLS	14	The number of magnetic poles used in the motor's construction. This value only needs to be set correctly if you want to use the RPM closed loop mode.
PWM_MIN_PERIOD	2200	The minimum period in microseconds that the ESC will consider an input PWM waveform to be valid. Default value represents approx 450Hz. **For MultiWii and ArduCopter APM this value needs to be changed to 2000!**
PWM_MAX_PERIOD	25000	The maximum period in microseconds that the ESC will consider an input PWM waveform to be valid. Default value represents approx 40Hz
PWM_MIN_VALUE	750	The minimum input PWM pulse length in microseconds which the ESC will consider to be valid. **For DJI Naza this value has to be 900!**
PWM_LO_VALUE	1000	The input PWM pulse length in microseconds for the lowest throttle setting.
PWM_HI_VALUE	1950	The input PWM pulse length in microseconds for the highest throttle setting.
PWM_MAX_VALUE	2250	The maximum input PWM pulse length in microseconds which the ESC will consider to be valid.
PWM_MIN_START	1100	The input PWM pulse length in microseconds at which the motor will be started. Once running, the throttle can be brought as low as PWM_MIN_VALUE as long as the motor does not stall.
PWM_RPM_SCALE	6500	The scale of the input PWM pulse length. In closed loop RPM run mode, PWM_LO_VALUE will indicate 0 RPM and PWM_HI_VALUE will indicate this RPM. Closed loop modes only!
FET_BRAKING	0	Turns on regenerative braking in closed loop modes (experimental), 0 == off, 1 == on. Closed Loop modes only!!

Closed loop calibration

The ESC32 offers a closed loop RPM mode and a Closed loop thrust mode (Experimental). It requires calibration of the specific motor and prop you will be using.

The calibration sequence is at an early stage in QGC and can be run under Linux and OSX from a terminal.

Closed loop calibration instructions

Please be very careful running a calibration!
Your motors will perform certain patterns up to full power. Make sure to run these tests only in a secured environment!
Only use it in closed rooms or boxes where propellors can fly off without hurting anyone or damaging anything!

Refer to forum.autoquad.org for more details on how to perform closed loop calibration of ESC32 under Linux, OSX or from within QGC.

ESC32 Current limiter calibration

ESC32 Current limiter calibration

Introduction

The purpose of ESC32 current limiter calibrations is to generate the Current Limiter (CL) terms for a particular motor/prop/powersource combination. This is done by a automated procedure in which the motor is asked to rapidly change speed and current is monitored.
The purpose of the current limiting calibration is to allow the motor a very fast response on smaller duty increases while still limiting the current if a large increase is demanded.
The biggest advantage of ESC32 (in open loop mode) is that there is no throttle shaping that traditional ESCs implement. That alone is enough to get good performance. However, without this traditional protection, current must be limited as you cannot give a motor 100% throttle changes in a single timestep. So the tradeoff for having quick throttle response is simply having to pick and implement an appropriate current limit.

How does the current limiter work

The job of the current limiter is to limit the maximum current allowed on ESC32. This limit is important to set correctly –overcurrent events can damage motors or make them loose sync leading to a rotor failure. The current limiter uses an onboard high-precision shunt to measure the current flow.
ESC32 has 2 options for current limiting: PI controller (default) or calibrated mode with CL terms. The intended behavior of the current limiter are the same whether you are using PI or CL terms – to reduce currents that could be harmful for motor and esc or make the motor loose sync.
The default setting is PI controller. In this mode, the current limiter is set to a maximum current allowed for RPM increase and overall current. The PI controller reads the current from the shunt and limits it according to the P and I terms of ESC32. But this is not ideal, and while this method works good for most motors and props, there is some motor and prop combinations that simply require individual calibration to make sure sync and commutation is not lost on rapid RPM increases.
If CL terms are generated and used, the current is limited by proactively reducing the throttle step change before it is seen instead of reacting to an over current situation as the PI controller does. This extra safety means that you can run closer to the limit without fear of going over current between timesteps
Current limiter calibration is always recommended to do for you specific motor/prop/power source combination.

Spikes and constant current

To understand current limiting, we have to differentiate between constant current and current spikes:
· Constant current is the current needed to maintain a given RPM.
· Current spikes are the short and large spikes that occur when the ESC is rapidly increasing RPM. It requires more energy to make a mass change speed than it takes to make that

object maintain its speed.
So even if your motor and prop only draws 10A at full speed, it may take 3-5 times that current to get from 10 to 100 percent duty in a single timestep (1/400 of a second) of the ESC and that may burn the ESC and motor if allowed. So we need to limit those spikes.
However, since we very rarely need to increase duty with more than 20-30 percent in a single timestep, we can allow the ESC such increases in speed very fast while still setting a limit for what is allowed if a larger increase is demanded.
This means we can have crisp response in the “normal range” and still be sure that if a large increase in speed is ever demanded, it then gets smoothed out over a few timesteps to avoid an overcurrent event that could cause the motor to loose sync or burn something out.

Setting the limit

There are 2 parameters you need to understand: Calibration current limit and overall current limit.
The calbration current limit defines how much current is allowed in one timestep during calibration. This is set during the CL calibration. If the motor looses sync during the calibration run, you should start over with a lower calibration current limit.
Overall limit is the maximum allowed current allowed. This is set in the ESC32 general settings. If the current limiter is not calibrated this limit should maximum be a few A more than what the motor draws constantly at 100 percent duty.
If the current limiter has been calibrated then set the overall limit at same or slightly above limit set in calibration.
The calibration procedure will output the result of the calibration expressed as 5 CL terms. It will also show the duty cycle increases and the currents resulting.
When the CL terms generated are loaded to the ESC32, they will be used instead of the PI controller to limit the maximum current increase in a single timestep. Maximum constant current is still limited by the overall current limiter setting.
So in rare cases, you could need a higher overall setting but in most cases the calibration and overall limits should be roughly the same or slightly higher for the overall limit
So even though a motor may be rated 25A from manufactorer, if you ramp throttle duty from 20 to 70 percent in a single timestep, the current needed by the ESC may very well exceed those 25A for a short while and that can be enough for the motor to loose sync or for the ESC and/or motor to burn out.

Example:

MT 3506-25 with 1150 Graupner Eprop – the 3506 is a known “trouble motor” that looses sync on rapid RPM increases with fast update ESC´s. But it works perfectly with ESC32, if correctly calibrated.

To get this motor to run properly the CL calibration was run on 4S 8000 25C, with a limit of 13A for the calibration. This still allows about 35 percent duty changes in 1 timestep without loosing sync. But calibration was run at higher current limit, the duty increases would exceed 35 percent and the motor would loose sync and stop.
The overall current limiter was set to 14A, to allow the motor to reach 100 percent duty with a little overhead, but the calibrated current limit will still limit the fast changes to 13A.

This is result is very much in line with specs for MT3506-25. A current of 14A on a 4S is

235W. As the motor is rated max 14.6A and 260W we are pretty close to that. This combination is flying a FCP-HL hexa at 1.8.-2.5 kg with no problems and the motors can reach 100 percent, but ramps exceeding 35 percent increase in a single timestep is smoothed out to keep things sane.

ESC32 technical glossary of terms

ESC32 technical glossary of terms

CLI

Command Line Interface, using a serial terminal (putty set on serial) the user can access the CLI using 230400bps. All configuration settings can also be done in the CLI.
There is also a binary protocol for high speed machine to machine communications up to 921600 baud.

closed loop RPM mode

Put simply, the input throttle requests are treated as a linear RPM request (which the controller is responsible for achieving and maintaining) instead of voltage requests. Only the ESC knows the true RPM (and therefore thrust) so it can close the loops and provide exactly what is requested. It can also do it very quickly with a 1KHz control loop.
The ESC utilizes a feed forward augmented PI controller to get you to the requested set point as quickly as possible but still remaining within the set current limit.

Regenerative Breaking

Regenerative braking uses the diodes in the FETs to rectify the flyback voltage to the power rails. Due to the conservation of energy principal, the motor’s kinetic energy is reduced and power is sent back into the main rail. In reality the extra power would mostly go towards speeding up the opposite motor during maneuvering. But still, recycled energy.

To get an idea of the concept, you can take a small BLDC motor and spin the spindle between your fingers. Now short one ore more phase wires and try again.

virtual current limiter

The problem with a lot of hobby ESCs is that they have to be careful of how much current they allow the motor to draw during acceleration events. The amount of current drawn by the motor depends on the motor’s size, RPM and prop load which is not known by the ESC and most do not have a current sensor.

The **virtual current limiter** will give the user a way to profile the motor and prop being used on the bench. This data can also be looked up from a database that is being compiled of known motor/prop configurations. With this information, the onboard current limiter can **pre-calculate** how much current a given throttle (voltage) request will take at the current RPM, and if it would result in an over current situation, the implemented throttle request is trimmed to fit under the limit. As the motor speeds up, the current demand drops even if the requested throttle has not changed so the limit is re-evaluated at 1KHz trying to give the maximum amount of acceleration possible with the given restrictions. The results are the fastest throttle response possible while staying under the set current limit. Finally, if the profiling parameters have not yet been set, a regular PI controller is used to limit current. It is still better than throttle filtering methods, but not as good as the virtual current limiter as it is a reactionary strategy.