Betaflight / Deutsch

Dies hier ist eine Sammlung an Dokumenten die im Laufe meiner kurzen FPV-Fliegerei entstanden sind und auch weiter entstehen.

Hintergrund für diese Ansammlung der Dokumente ist, dass ich mehr Verständnis für mich selbst schaffen möchte und dadurch ein besseres Verständnis über die Zusammenhänge und dem Flugverhalten von FPV-Coptern zu erhalten.

Weiterhin möchte ich kein endloses Wiki in deutsch erstellen sondern einzelne Themenbereiche herauslösen - auch wenn es dadurch Redundanzen gibt. Für mich hat es den Vorteil, dass ich bestimmte Dokumente ausdrucken kann und mit zum Fliegen nehmen kann, wenn ich einen Copter optimieren möchte.

Tja und zum Schluß - ich kann mir einfach nicht alles im Kopf behalten, daher lese ich gerne nach und dann helfen mir kleinere Dokumente mehr als große.

Die einzelnen Seiten stehen auch zum Download als PDF zur Verfügung.

Alternative: Downloaded dieses Repository komplett und startet über Euren Browser `index.html`

Vielleicht hilft es dem ein oder Anderen

Du kannst alle Seiten als PDF downloaden über .....

Viel Spass und propwash freies Fliegen

LunaX - August 2020

Handhabung dieses Github-Repositories

Dies Repository basiert auf MKDocs (https://mkdocs.org). Alle geschriebenen Seiten wurden als einfache Markdown-Dateien erstellt.

```
mkdocs.yml (Konfigurationsdatei für MKDocs)
docs/
    xxxx.md Dateien (einzelne Markdown-Dateien zu den Themen)

sites/
    pdf/combined.pdf (hier liegt die Gesamt-PDF-Datei)
    <folder> (diverse Subfolder, enthalten HTML-Seiten zum Offline-Lesen)
```

BF 4.2 - Neuerungen

Historie

Version	Datum	Inhalt
0.1	August 2020	initial

Allgemeine Anpassungen

- BF-Configurator 10.7.0 ist notwendig
- Blackbox-Explorer 3.5.0 für BF 4.2
- Neue Tuning-Möglichkeiten (Slider)
- **Bei einigen Targets downgeloaded werden müssen, daher ist 10.7.0 wichtig.**
- **Keine Konfigurationen aus älteren Releasen importieren** Einige Default-Werte haben sich geändert
- Es wird kein Motorprotokoll automatisch ausgewählt. Dies muss nach einem Update manuell durchgeführt werden.
- Wenn bidirektionales DShot ausgewählt wird, die Geschwindigkeit des Protokolls wird halbiert um die Bidrirektionalität aufrecht zu erhalten. Dies gilt **nicht für DShot600**. Bei `DShot300` muss die PIDLoop auf 4k gesetzt werden. Bei `DShot150` auf 2k. Demnach bei `DShot600` auf 8k
- Bevor gearmt werden kann muss der Accelerometer calibriert werden.
- Die Strommessung ist nun am Throttle-Signal gekoppelt, das ermöglicht besser Vorhersagen zum Stromverbrauch
-

Major

- Überarbeitung der Gyro-Loop. Diese läuft nun in der Geschwindigkeit der nativen Beschwindigkeit des Gyros
- zwei neue Rates auswählbar `ACTUAL` und `QUICK`
- VBat Kompensation, wenn die Batteriespannung nachläßt
- Neuer NFE Race Modus

Minor

- OSD Logo nun auch beim Armen sichtbar
- Unterstützung für verbesserte OSD-/CMS-Geräte hinzugefügt, wodurch es möglich wurde, das Highligting von Text oder Symbolen zu unterstützen
- Unterstützung für FrSkyOSD OSD-Geräte hinzugefügt
- Unterstützung für das Redpine RC-Protokoll auf Geräten mit einem über SPI angeschlossenen CC2500-Chip (FrSky SPI) hinzugefügt

BF 4.1 - Neuerungen

Historie

Version	Datum	Inhalt
0.1	August 2020	initial

Allgemeine Anpassungen

- **Keine Konfigurationen aus älteren Releasen importieren** Einige Default-Werte haben sich geändert
- BF-Configurator 10.6.0 ist notwendig
- Blackbox-Explorer 3.4.0 für BF 4.1
- DSHot für bidirektionalen Einsatz mit RPM-Filtern wurde verbessert. BLHeli_32 ab Version 32.7 ist erforderlich
- VTX-Tabellen sind nun nach einem flashen neu zu laden. Hiermit können nun die VTX deutlich besser konfiguriert werden
- Eine Anpassungen an den OSD-Fonts

Major

- **NEU** Feedforward 2.0 (https://github.com/betaflight/betaflight/wiki/Feed-Forward-2.0)
- Überarbeitetes bidirektionales DShot
- Dynamisches Leerlauf-Managment nutzt nun RPM-Telemetrie
- voll konfigurierbare VTX über VTX-Tabellen

Minor

- Unterstützt nun Spektrum SRXL2 Protokoll
- Board spezifische Defaultwerte
- Unterstützung von willkürlichen Gyro & Magnetometer Ausrichtungen

BF 4.0 - Neuerungen

Historie

Version	**Datum**	**Inhalt**
0.1	August 2020	initial

Allgemeine Anpassungen

- **Keine Konfigurationen aus älteren Releasen importieren** Einige Default-Werte haben sich geändert
- BF-Configurator 10.5.1 ist notwendig
- Blackbox-Explroer 3.3.1
- BF 4.0 enthält

Major

- Echtzeit RPM feedback und Notch-Filter basieren auf RPM
- DTerm Management mit `d_min`
- Throttle basierendes dynamic gyro und DTerm Filterung
- Start Kontrolle (launch control)
- Umschaltbare OSD-Profile
- Vereinheitliche Zielsysteme

Minor

- Kaskadierende dynamische Notch-Filter
- Schub Linearisierung
- Integrierte YAW Kontrolle
- Umschaltbare LED_Strip Profile
- Gimbal-Overlay im OSD
- Profil-Switching basierend die Anzahl der Lipo-Zellen
- Pro Profil Limitierung des maximalen Motor Outputs *

FILTER Ein mal Eins

Inhaltsverzeichnis

{{TOC}}

Historie

Version	**Datum**	**Inhalt**
0.1	August 2020	initial

Allgemeines

Typische Frequenzen

Vibrationen die am Copter auftreten lassen sich in verschiede Frequenzbereiche unterteilen

- 0 ~ 20Hz: normaler Frequenzbereich während des Fliegens (keine Filterung der Signale)
- 20 ~ 80Hz: Vibrationen niedriger Frequenz (häufig Propwash)
- 80 ~ 1000Hz: hochfrequente Vibrationen (Noise) verursacht durch
    - Motorvibrationen & Prop-Resonanzen. Hier sieht man auch das typische Bild des Motor-Vibrations-Bandes Harmonics
    - Frame-Resonanzen
    - ...

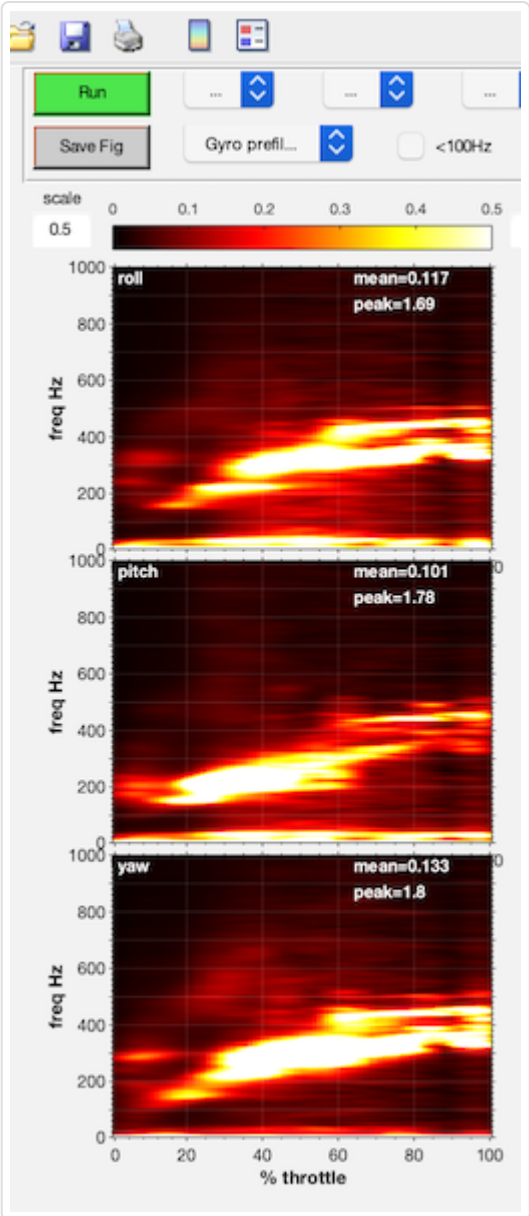
Noise / Vibrationen

Ist eine Bezeichnung für Rauschen bzw. Gyro-Signale die nicht zu den gewünschten/benötigten Signalen sind, die durch den FC verarbeitet werden müssen. Also Messwerte die irreguläre sind und die Performance des Copters negativ beeinflussen. Noise (Rauschen) kann ausgelöst werden durch: * äußere Einflüsse (z.B. Wind, Kontakte mit Ästen, ...) * Vibrationen am Chassis (z.B. lose schrauben, dünne schwingende Arme, ...) * Vibrationen durch Motoren(z.B. Lagerschäden, Unwucht, ...) * Vibrationen durch Props (z.B. Unwucht, Prop-Wash, ...) * Kombinationen aus allem •.

Noise wird durch den Gyro und anschließend durch den PID- Controller verarbeitet und kann zu fehlerhaften Verhalten führen.

Eingesetzte Filter LPF[4], NOTCH[3], RPM[1],DTerm[5] versuchen dieses Rauschen zu eliminieren.

Eine Deiner Hauptaufgabe beim Tunen Deines Copters ist, dass du diese Vibrationen in den Griff bekommst ohne deutliche Delay zu bekommen.



Hinweis:

Der DTerm-Anteil des PID-Controllers verstärkt Vibrationen deutlich. Daher ist eine Abstimmung der Filter mit dem DTerm gut abzustimmen.

Um Vibrationen zu analysieren musst du eine Blackbox-Analyse durchführen oder Tools wie `Blackbox-Explorer`, `PIDToolbox` oder `Plasmatree` verwenden.

Um einen ersten Eindruck von Vibrationen zu erhalten empfehle ich eine Spektral-Analyse (https://github.com/mrRobot62/PIDtoolbox/wiki/SpectralAnalyzer). Hier sieht man sehr deutlich in welchem Frequenzband Vibrationen an Deinem Copter auftreten.

Filter helfen, diese Vibrationen möglichst aus den Signalen für den Flight-Controller herauszufiltern um anschließend den Motoren möglichst saubere Signale zu übermitteln.

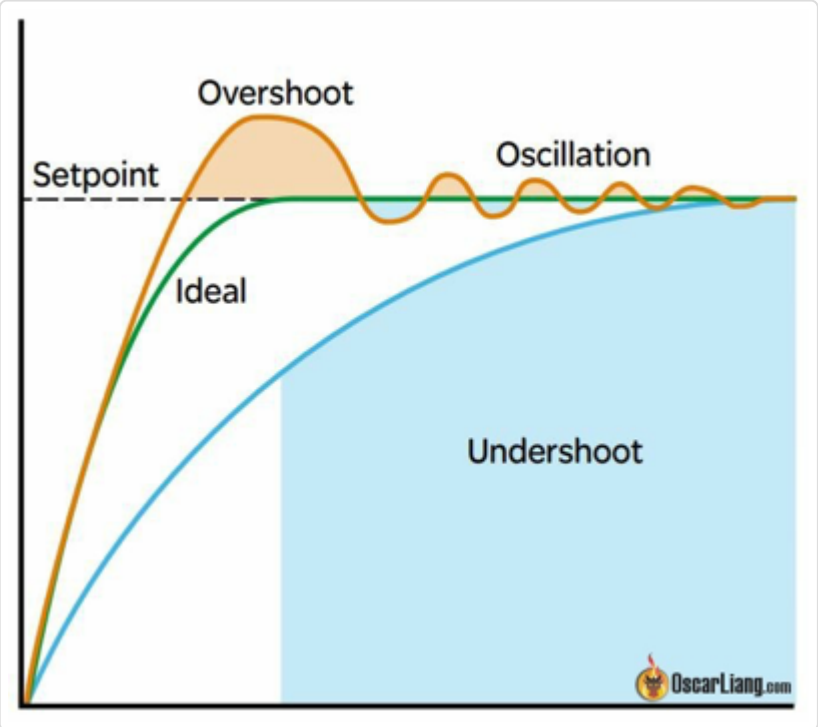
Tip

> Bevor du damit startetest die PID-Werte anzupassen, stelle Deine Filter optimal auf Deinen Copter ein. Erst dann fängst du mit den PID-Werten an.

Versuche durch Deine Filter ein maximales Gyro & DTerm Delay von <5ms zu erreichen. Du kannst das sehr einfach mit `PIDToolbox` erkennen.

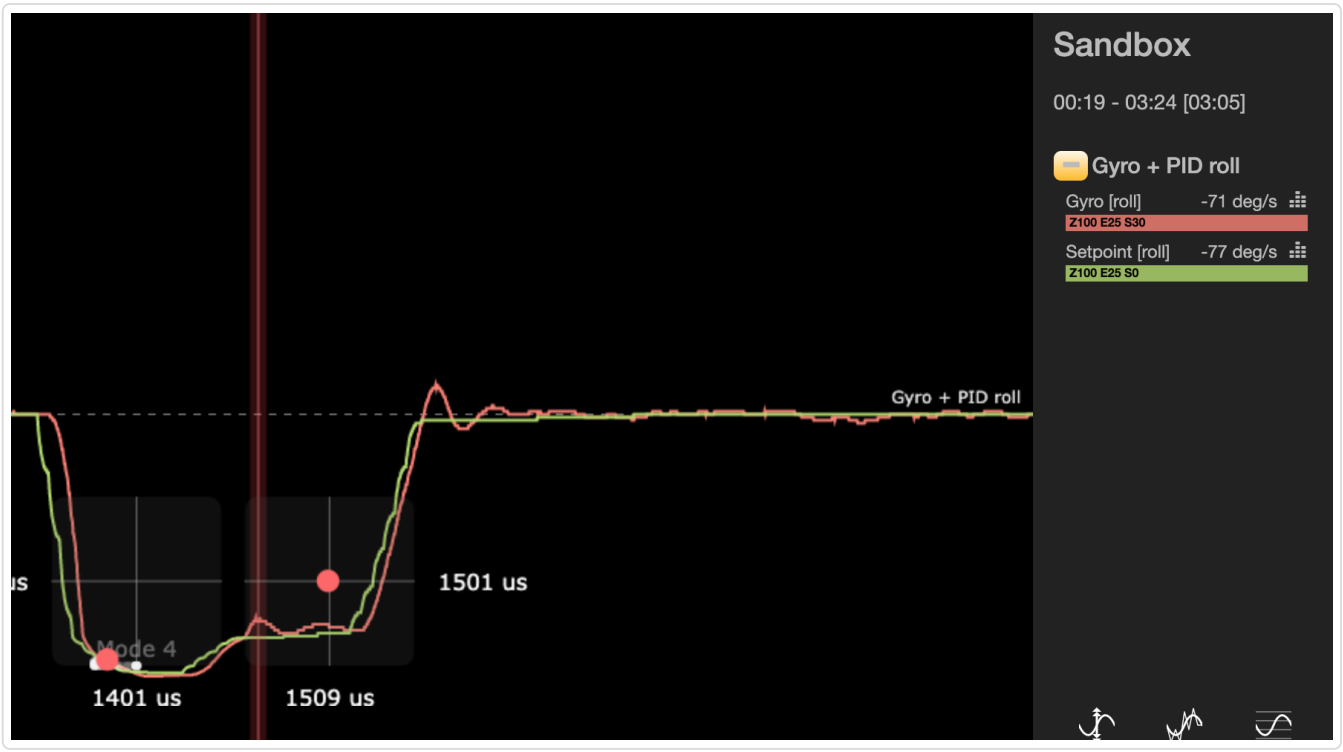
Oszillation

Eine Oszillation ist eine Schwingung. In Bezug auf den PID-Controller ist es ein Über- und Unterschwingen zur Ideallinie.

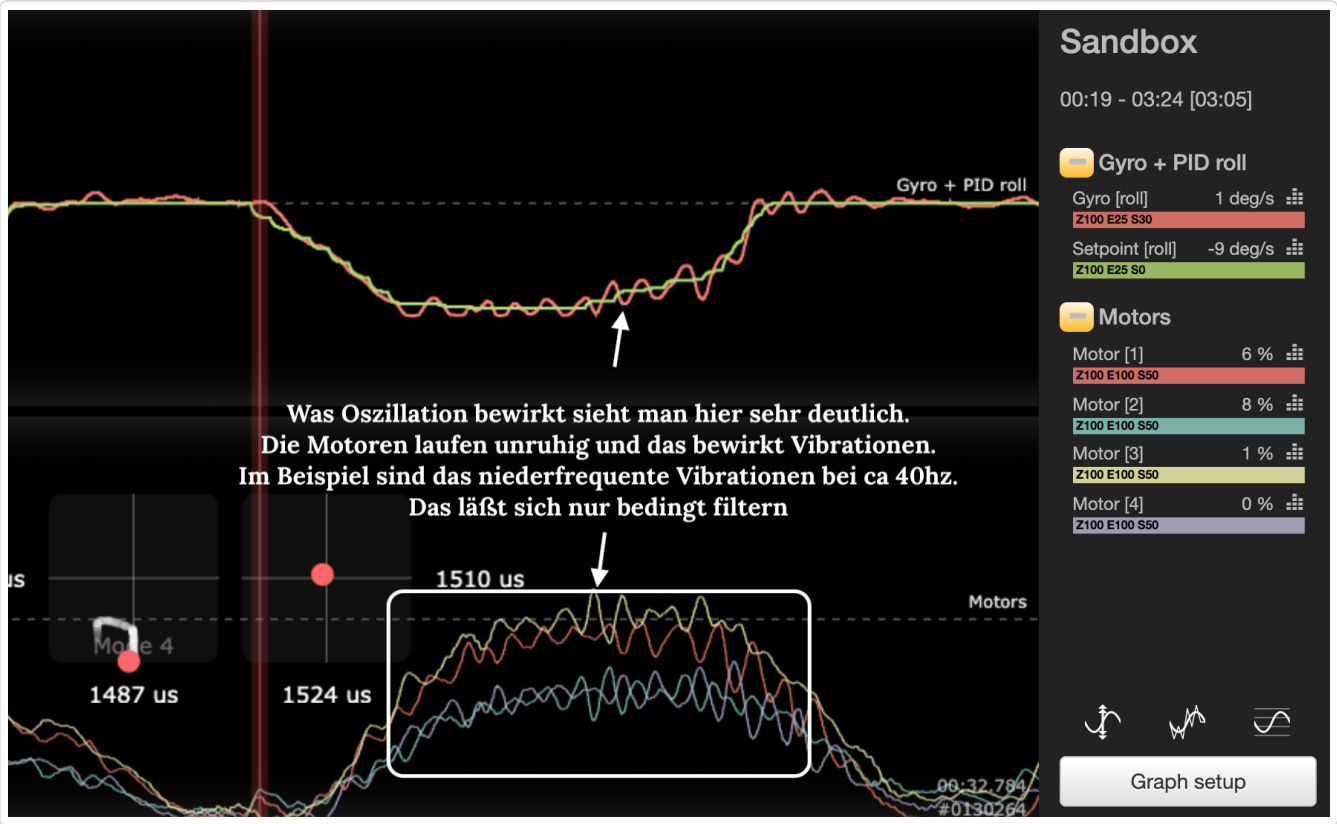


- **SetPoint** = SOLL-Wert = gestrichelte Linie (das ist das RC-Kommando z.B. 700deg/sec)
- **Orangene-Linie** ein Versuch sich möglichst schnell an den Soll-Wert heranzutasten. Man überschießt den Soll-Wert korrigiert und fällt unter den Soll-Wert, korrigiert dann wieder überschießt man usw. bis man irgendwann den Sollwert Erreicht.
- **Ideal** ist es, wenn man die **grüne Linie**) möglichst schnell ohne bzw. geringer Oszillation den Sollwert erreicht
- **Blauer Bereich** Schafft es der PID-Controller überhaupt nicht den Soll-Wert zu erreichen dann bewegt man sich in diesem Bereich. Schnelles Oszillieren wird als Vibration empfunden.

Nachfolgend ein Bild aus dem Blackbox-Explorer bei dem man sehr deutlich diese Oszillation sehen kann


Sandbox
00:19 - 03:24 [03:05]
Gyro + PID roll
Gyro [roll] -71 deg/s
Z100 E25 S30
Setpoint [roll] -77 deg/s
Z100 E25 S0
Gyro + PID roll
1501 us
Mode 4
1401 us
1509 us

Im nachfolgenden Bild sieht man im oberen Graphen die Oszillation auf der Roll-Achse und welche Auswirkungen diese auf die Motoren haben. Dies läßt sich nur bedingt durch Filter berreinigen, sondern über die PID-Einstellungen.

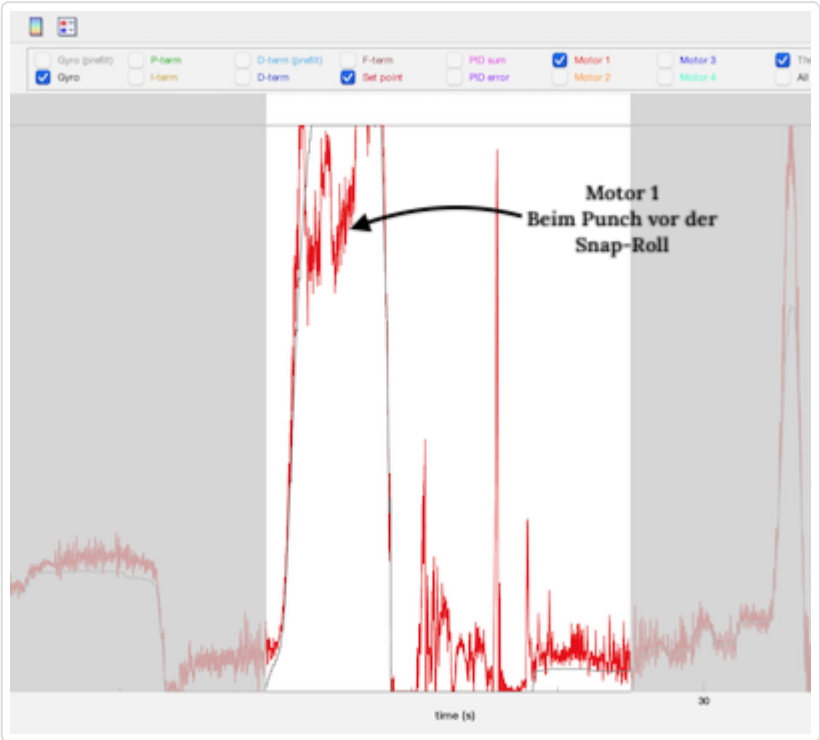


Nachfolgende Bildfolge zeigt die obigen beschriebenen Vibrationen der Motoren bei einem Punch. Analyse über PIDToolbox[^PDT]

Roll

Info

Snap-Roll, deutlich sieht man die Oszillation beim Erreichen des Setpoints. Interessant sind auch die Ausschläge kurz vor der Snap-Roll. Diese Ausschläge sind vermutlich durch Vibrationen der Motoren indiziert



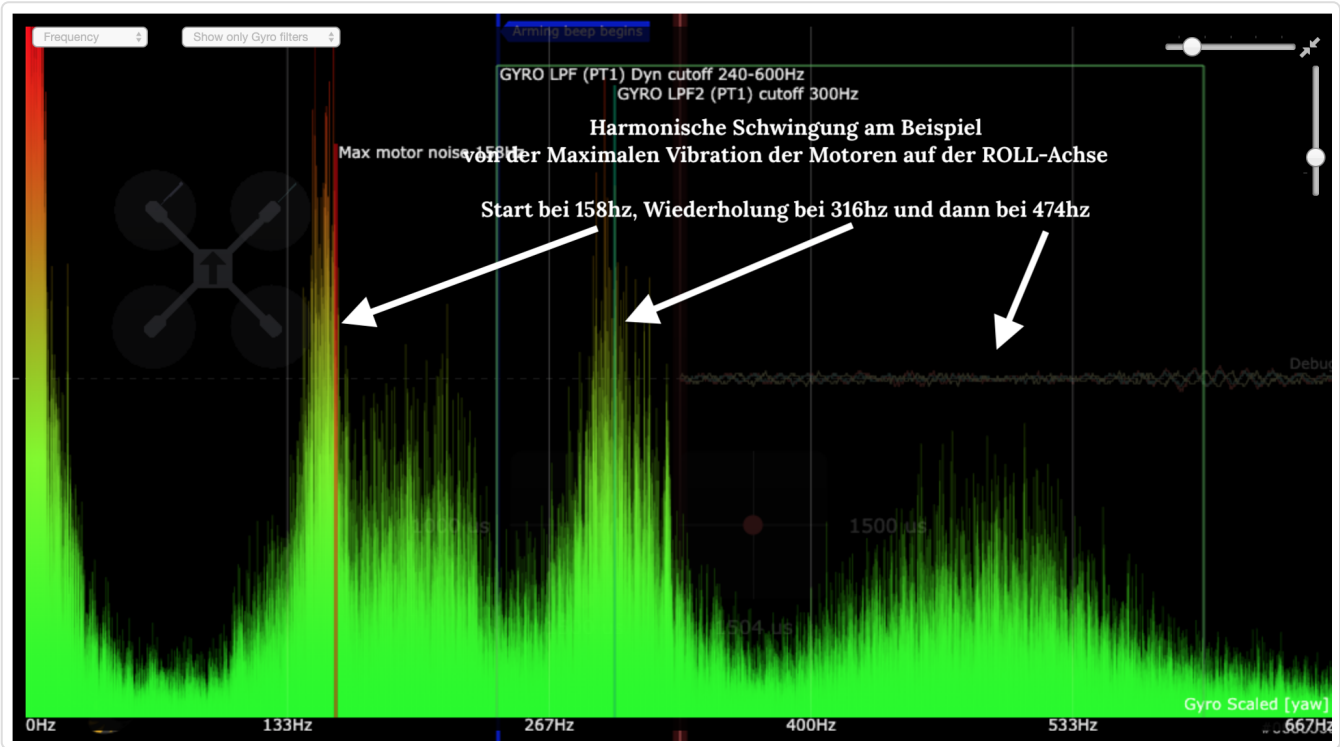
Dieses Bild zeigt Motor1 beim Punch kurz vor der Snap-Roll, man sieht das Vibrationen den PTerm und DTerm beeinflussen, was in Schwankungen der Motordrehzahl zeigt. Dadurch entstehen auch Vibrationen.

Harmonics

Harmonics oder Harmonische sind wiederkehrende Amplituden (Vibrationen) in den gleichen Frequenzabständen.

Das nachfolgende Bild zeigt deutlich drei sich im gleichen Abstand wiederholende Frequenz-Peaks von Vibrationen (Motor Vibrationen)

Beginnend mit dem Motor-Noise-Peak bei 158hz, dann die erste Harmonische Amplitude bei 316hz (2x158hz) und die dritte Amplitude bei 474hz (3x158).



Gyro-Data-Filtering

Signal & Prozessfluß zur Filterung

graph LR classDef dbg fill:#ddd classDef LP fill:#f96 LS[Loop-Start
d/t]:::LP --> GY(Gyro) GY -.-> DBG1[[DebugMode
Gyro_Raw
Notch
FFT]]:::dbg GY --> DNF DBG1 -.-> DNF(Dynamic Notch Filter) DNF -.-> DBGFFT[[DebugMode
FFT]]:::dbg DNF --> SNF(Static Notch Filter) DBGFFT -.-> SNF
graph LR classDef dbg fill:#ddd SNF(Static Notch Filter) --> GLPF SNF -.-> DBGGY[[DebugMode
Gyro]]:::dbg DBGGY -.-> GLPF(Gyro LPF
LowPass Filter) GLPF --> PID(PID Loop)
graph LR classDef dbg fill:#ddd PID(PID Loop) -.-> DBGDF[[DebugMode
DFilter]]:::dbg PID --> DTLPF(DTerm
LPF) DBGDF -.-> DTLPF DTLPF --> DTNF(DTerm
Notch Filter)
graph LR classDef dbg fill:#ddd classDef LP fill:#f96 DTNF(DTerm
Notch Filter) --> M(Motoren) M --> LE[Loop-End]:::LP

Grundlage Filter-Arten

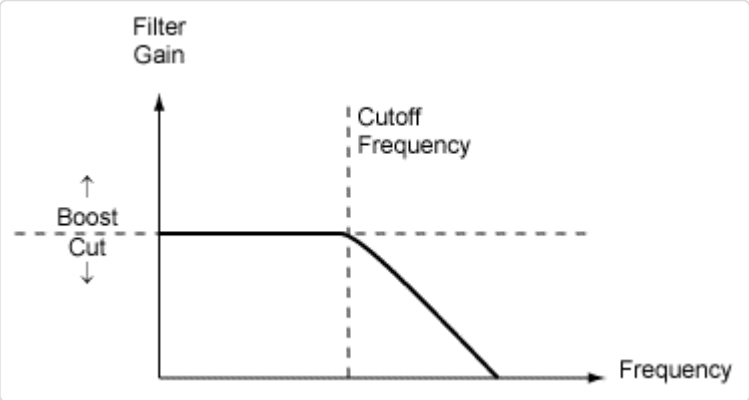
Lowpass-Filter

Niedrige Frequenzen werden durch den Filter durchgelassen, hohe Frequenzen werden gedämpft.

Hohe Frequenzen sind in der Regel im System nur Rauschen bzw. Vibrationen und werden für die Flugdaten nicht benötigt, daher versucht man sie herauszufiltern.

Mit dem LPF[4] wird eine Grenzfrequenz (`cutoff`) angelegt und der FC reduziert die Signale die oberhalb dieser Grenzfrequenz liegen.

Die Dämpfungskurve ist eine Steigung. d.h. je höher die Signalfrequenz desto stärker die Dämpfung



Für den Einsatzbereich des Quadcopters ist der Frequenzbereich von 0-80Hz relevant, alles darüber sollte möglichst effizient weg gefiltert werden.

Beachten

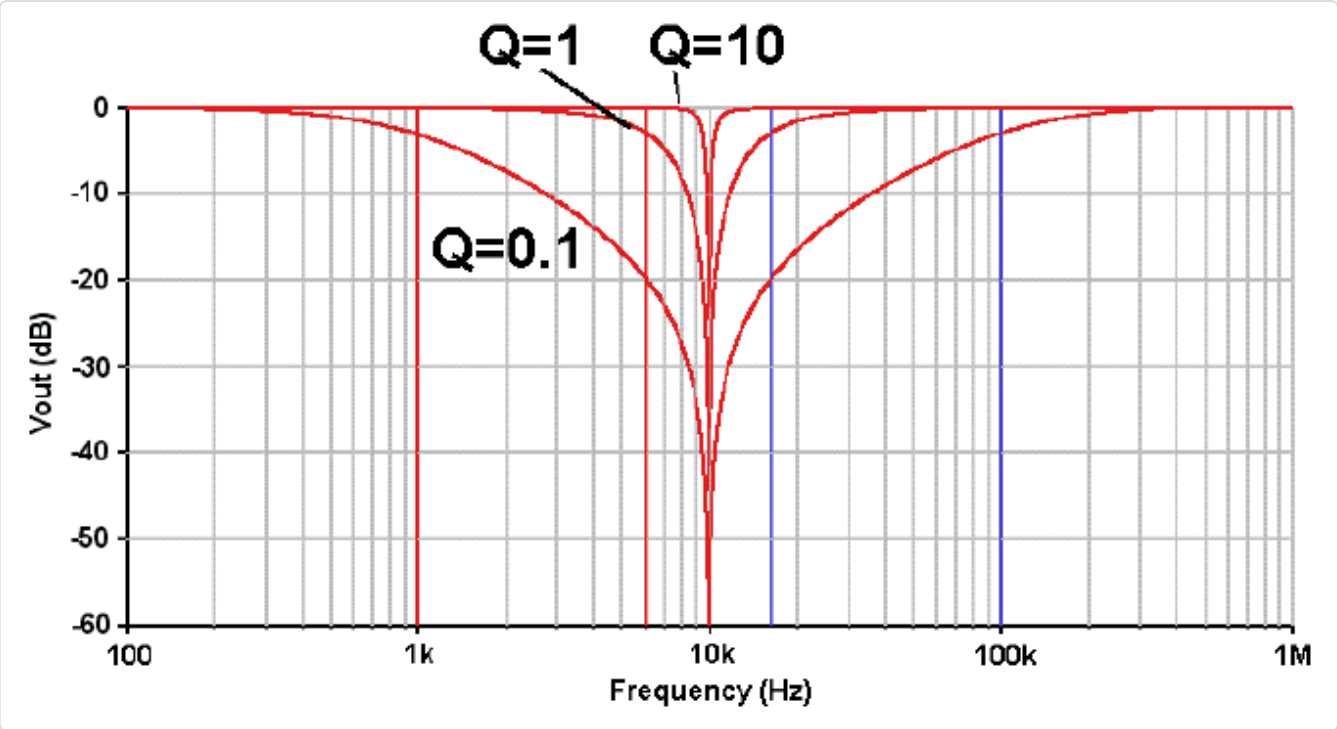
> Ein fundamentaler Aspekt ist, je niedriger der Schwellwert(`cutoff`) für den Lowpass Filter ist - umso mehr muss gefiltert werden.
>
> Das wirkt sich auf dem Gesamtperformance aus. Daher gibt es mehrere andere Filter die andere Algorithmen verwenden und das System effizienter gestalten.

Notch-Filter

Notch-Filter[3] eigenen sich hervorragend zur Unterdrückung von Rauschen in einem sehr spezifischen Frequenzband.

Notch-Filter sind in der Regel effektiver zur Reduzierung von Motorrauschen als LPF- Filter aber ggf. müssen manuelle Abstimmungen durchgeführt werden um die Bandbreite und die Mittenfrequenz zu bestimmen (Notch=Kerbe)

Weiterhin können Notch-Filter zur Reduzierung von Propwash genutzt werden.



Der `Q-Faktor` (Quality-Factor) gibt die Güte des Notch-Filters an und beschreibt die Weite des Filters. Je größer der Q-Faktor, desto schmaler der Notch-Filter.

Beispiel:

Es wird festgestellt das wir ein Vibrationsspitze bei ca 260Hz haben (z.B. hervorgerufen durch Propwash), dann kann der Notch-Filter `cutoff` bei ca. 200Hz beginnen und bei 300Hz enden

Static-Filter

Im Rahmen von Betaflight werden statische Filter als LowPass-Filter oder als Notch-Filter genutzt

Dynamic-Filter

Dynamic Filter reduzieren ebenfalls Rauschen, wenn die entsprechenden Parameter richtig eingestellt sind. Schwingungen, Motorgeräusche, ... können durch die Dynamic-Filter reduziert werden.

Ein Dynamic-Filter ist ein Algorithmus, der die Frequenz des Rauschens erkennen kann und er kann den Notch-Filter verwenden, um ihn automatisch zu reduzieren.

Nachteil von Dynamic-Filters ist die Erhöhung der CPU-Last und die Delays.

Betaflight-Filter

RPM-Filter

RPM-Filter [1] sind eine besondere Filtertechnik in Betaflight. RPM-Filter basieren auf mehreren Notch-Filter die präzise darauf ausgerichtet sind Motorvibrationen und ihre `Harmonics` zu filtern und möglichst komplett zu elimenieren. Für jeden Motore können bis zu drei Notch-Filter (max 3 Harmonics) generiert werden (Pro Achse). Das heißt insgesamt steht eine Bank von 36 Notch-Filtern zur Verfügung. 3 Achsen * 3 `Harmonics` * 4 Motoren = 36 Notch-Filter.

Die RPM-Filter basieren auf die Drehzahl der Motoren und dem bidrektionalen DShot-Protokoll in Zusammenspiel mit den Gyrodaten - daher auch `Gyro-RPM-Filter`

BEACHTEN

> RPM-Filter benötigen für BLHeli32 ESC die aktuellste Firmware `>= 32.7`
>
> Für BLHeli_S ESCs empfehle ich die FW von (auch wenn sie Geld kosten) JFlight[2]

BF - Dynamic-Lowpassfilter

Neu ab [BF 4.0] (https://github.com/betaflight/betaflight/wiki/4.0-Tuning-Notes#Dynamic-lowpass-filtering). BF bietet nun die Möglichkeit, die Grenzfrequenz des LPF[4] bei Vollgas im Vergleich zu 0-Throttle sanft auf einen höheren Wert zu verschieben. Die dem ersten Gyro und dem D-Lowpass-Filter zugewiesene Grenzfrequenz steigt nun dynamisch mit zunehmender Drosselung entlang einer Kurve an, die effektiv die Motordrehzahl beeinflusst. Dadurch wird die Verzögerung bei Vollgas verringert, und der Dynamische-Notch Filter [3] kann die Motor-Vibrations-Peaks besser verfolgen.

BF - Static Notchfilter

Statische Notchfilter werden explizit mit der Center-Frequenz auf auf die Herzzahl der höchsten Vibrationen (Peak) gesetzt. In BF kann man zwei dieser Notch-Filter aktivieren. Angegeben werden jeweils die cutoff-Frequenz und die Center-Frequenz.

BF - Dynamic-Notchfilter

Ab `BF 3.1` kann man auch `Dynamic Notch Filter`[3] einsetzen. Dieser hocheffiziente Dynamische Notch-Filter setzt sich automatisch auf den Motor-Peak - also den aktuelle höchste Frequenz der Motor-Vibrationen, basieren auf die jeweils aktuelle FFT-Analyse. Hierdurch wirkt dieser Notch-Filter in allen Throttle-Stellungen und wandert demnach durch das Frequenzband.

Dynamische Notch-Filter haben eine geringere Latzenzzeit als LP-Filter. Notch-Filter sind sehr effektiv und bei einer guten Abstimmung kann man sogar auf LPF-Filter verzichten. Dadurch steigert sich die Gesamtperformance des Copters.

LP-Filter aber einfach abschalten sollte wirklich mit bedacht gemacht werden.

Der Dyn-Notch-Filter ist per default eingeschaltet.

Es gibt zwei Notch-Filter für den GYRO. Einer oder beide können bei Bedarf abgestellt werden.

Default-Values sind auf 200Hz-400Hz. Diese Grenzfrequenzen arbeiten sehr gut und funktionieren bei den meisten Coptern.

Für eine Feinabstimmung ist es notwendig eine Blackbox- Auswertung durchzuführen (Blackbox-Explorer, PIDToolbox, Plasmatree)

BF - Static-Lowpassfilter

In Betaflight wird zwischen zwei Static-LPF Filtern unterschieden

- PT1
- BIQUAD

Um einen LPF-Filter zu nutzen muss grundsätzlich die untere Grenzfrequenz (`cutoff`)) angegeben werden. Der Filter beginnt dann ab dieser Frequenz zu arbeiten. Die Cutoff-Frequenz sollte **nicht** unter 80Hz liegen, da hier die normalen Flugfrequenzen liegen.

PT1

Dieser Filter hat eine etwas sanftere Kurve und ist ein LPF-Filter 1. Ordnung und hat dadurch eine geringere Latzenzzeit. Der Nachteil dieses Filters, er filtert nicht so stark Vibrationen aus dem Signal (bedingt durch seine Kurvenausprägung)

TIP

> verwende für den ersten LPF-Filter die Einstellung **PT1**, denn er ist der schnellere Filter.

BIQUAD

Dieser Filter hat eine deutliche steilere Kurve und filtert besser als ein PT1. Er ist ein Filter 2. Ordnung. Dadurch ist die Latzenzzeit schlechter, das Filterergebniss besser.

BF - Gyro-RPM-Filter

Ein mächtiges neues Feature, das mit BF 4.0 eingeführt wurde und in den nachfolgenden Releasen weiter verbessert wurde. Die RPM Filter wurden weiter oben schon beschrieben, nachfolgend eine Reihe von Detailinformationen.

BF - Gyro-LowPass Filter

BF - DTerm LowPass Filter

Der DTerm verstärkt alles - auch Vibrationssignale - somit können ungefilterte DTerm Werte die direkt zu den Motoren gesendet werden dazu führen, dass die Motoren heiß werden oder sogar überhitzen und zerstört werden.

Hier kommen jetzt die DTerm-Lowpass Filter -> Einstellung wie bei den Static-Lowpassfilter

Allgemeinen ist ein Biquad-Filter das sinnvolle Minimum an Filterung mit einer Frequenz um 100 Hz bis hinunter zu 80 Hz, wenn man heiße Motoren hat.

Beachten:

> Den DTERM-LPF solltet ihr grundsätzlich **nicht** entfernen in Eurer Konfiguration!

---

1. BF-RPMFilter (https://github.com/betaflight/betaflight/wiki/Bidirectional-DSHOT-and-RPM-Filter) ↩ ↩
2. https://jflight.net/index.php ↩
3. Notch-Filter = Kerb-Filter ↩ ↩ ↩ ↩
4. Lowpass-Filter (Tiefpassfilter) ↩ ↩ ↩
5. Filter auf den DTerm-Wert ↩

BF Blackbox-Analysen

Inhaltsverzeichnis

{{TOC}}

Allgemeines

PID - Controller

Inhaltsverzeichnis

{{TOC}}

Historie

Version	**Datum**	**Inhalt**
0.1	August 2020	initial

Allgemeines

Berechnung des Fehlerwertes zwischen dem SOLL und IST-Wert Der jeweilige Fehlerwert wird mit einer Konstanten (Kp, Ki, Kd)[1] multipliziert. Die Summe aller der Fehler ergibt den Gesamt Fehler PIDError

Soll-Wert

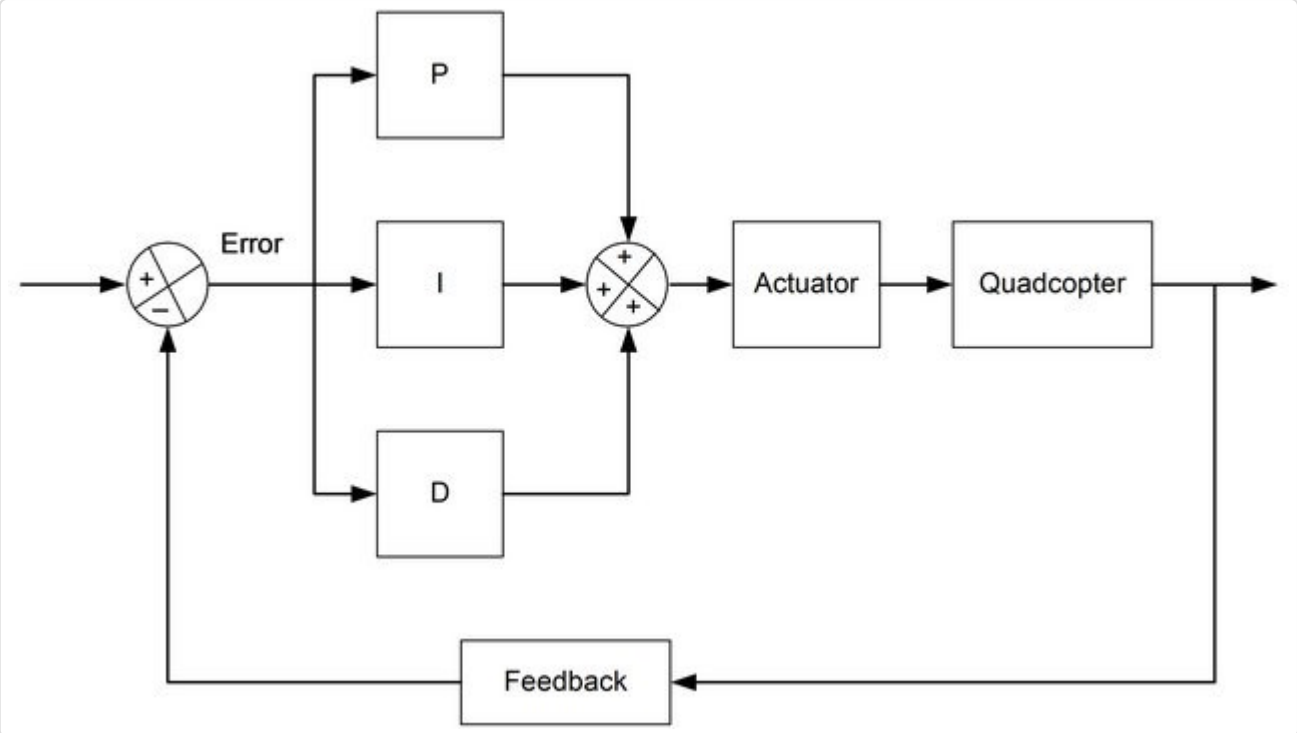
In Falle von Coptern ist der Sollwert, der Wert der durch die Gimbals vorgegeben wird (RC-Command). Entspricht also dem Zielwert, den der Copter erreichen soll (z.B. der Copter soll ein Rolle mit 700deg/sec durchführen.

IST-Wert

Ist der Wert, der durch das Gyro über alle drei Achsen gemessen wurde

Soll & **IST** werden in der PID-Loop kontinuierlich gelesen und ausgewertet. Signale werden geglättet und gefiltert. Das Endresultat entspricht für die aktuelle Zeiteinheit (d/t) den anliegenden PID-Error[2]

- P proportionaler Anteil
- I integraler Anteil
- D derivativer Anteil



PID-Loop

Die PID-Loop in Betaflight beinhaltet (für **alle** Achsen) folgende Punkte (vereinfachte Darstellung und nicht 100% vollständig). Die PID-Loop beschreibt auch die maximale Looptime

graph LR classDef LP fill:#f96 ls[Loop
Start d/t]:::LP le[Loop
ende]:::LP RC(RC-Cmd) RPM(RPM-Tele) GY(Gyro) GF(Gyro
Filter) DT(DTerm
Filter) MIX(Mixer) MOT(Motoren
1-4) SM(Smooting) PERR((PID
Error)) PSUM((PID
Sum)) PID(P-I-D
Terms) ls--> RPM & RC & GY RC --> SM --> PERR GY --> GF --> DT RPM --> GF GF & DT --> PERR subgraph PIDCTRL [PID-Controller] PERR --> PID --> PSUM end PSUM --> MIX MIX --> MOT MOT --> le

P-Term (Proportionaler Fehler)

Der PTerm versucht den proportionalen Fehler möglichst schnell auf 0 zu reduzieren.

Vereinfacht gesagt: Wie hart der FC daran arbeitet den Fehler zu korrigieren. Je höher der PTerm um so schärfer ist die Reaktion. Ein zu hoher PTerm führt aber zu Oszillation und Überschwingen.

Eingabewerte:

- SOLL ist der RC-Command Wert (Stick)
- IST ist der GYRO Wert

Beispiel:

```
Kp = 0.2
Input = 100
GYRO = 60
Perr = Kp * (60-100) => 8
```

I-Term (Integraler Fehler)

Ist eine Aufsummierung aller bis dato aufgetretenen Fehler über die Zeit (d/t).

Mit dem ITerm wird eingestellt wie hart/schnell der FC reagieren soll gegen Umwelteinflüsse (z.B. Wind) um eine definierte Lage/Höhe beizubehalten.

Somit wird ein stetiger Fehler, der anliegt und durch den P-Wert nicht korrigiert werden konnte, durch den I-Wert kompensiert, um den Gesamtfehler möglichst schnell auf 0 zu bringen.

ITerm: Driftet der Copter ohne Steuerbefehl, dann den ITerm erhöhen.

Musst man sehr häufig die Flugbahn korrigieren (besonderen bei höheren Throttle) dann ist der ITerm zu niedrig

Beispiel:

> Wenn bei schnellen Throttle-Bewegungen der Copter nicht stabil bleibt, ist häufig der ITerm zu niedrig.

`d/t = Zeiteinheit – im Beispiel gehen wir von 1 aus (einfacher zu rechnen)`

```
Im Beispiel gehen wir davon aus, dass IerrSum = -1 ist
    Ki = 0.02
    Input = 100
    Gyro = 80
    IerrSum = IerrSum + (80-100
    Ierr = Ki * IerrSum * d/t
    Ierr = 0,05 * -21 * d/t
    Ierr = -1,05
```

D-Term Derivativer-Wert (Vorhersage)

Der DTerm ist im Prinzip der Gegenpart zum PTerm und versucht eine Vorhersage zu treffen, wie der Fehlerwert in der Zukunft ist und versucht diesem entgegen zu wirken.

P & D hängen eng beieinander.

Der DTerm ist ein Dämpfungsglied für ein Überkorrigieren des P-Reglers und versucht „Overshoots“ zu minimieren. Ähnlich einem Schock-Absorber.

Den DTerm erhöhen kann eine Oszillation mehr glätten. Zu hohe DTerm führen aber zu heißen Motoren und können bis zur Zerstörung des ESCs oder des Motors führen.

Extensive D-Werte führen auch zu einer Vermindung des Anspruchverhaltens des Copters.

```
Im Beispiel gehen wir davon aus, dass DerrAlt = -4 ist
    Kd = 0.1
    Input = 100
    Gyro = 80
    DerrTmp = (80-100) - DerrAlt = -16
    Derr = Kd * (-16) * d/t
    Derr = -1.6
```

Looptime (d/t)

Den Zyklus den der PID-Controller benötigt das Eingangssignal (Eingangswert) und der daraus resultierenden Kalkulation und einen Ausgabewert zu berechnen bezeichnet man als „Loop“.

Die dazu benötigte Zeit wird „Looptime“ genannt Looptime wird in ms (Millisekunden) berechnet bzw. in Hz

```
1sek = 1000ms = 1Hz = 1 Zyklus
1ms = 0.001sek = 1KHz

4k Looptime = 4000x die Loop durchlaufen pro Sekunde
```

Daher ist es auch wichtig, dass man in BF die Looptime so einstellt das der FC dies auch verarbeiten kann ohne Fehlberechnungen durchzuführen

Beispiel FC F405 4KHz = 4000 Loops pro Sekunde - das schafft der FC problemlos 8KHz = 8000 Loops ist für einige F4 FCs zu viel, wenn zusätzliche Filter eingeschaltet wurden. Bei F7 FCs ist 8K typisch.

---

1. Fehlerkonstanten, werden pro Achse in BF eingestellt. ↩
2. PID-Error, Summe aller anliegenden Fehlersignale. [^DT] ↩

BF Allgemeine Tuning Tips

Inhaltsverzeichnis

{{TOC}}

Allgemeines

Um den Copter zu tunen sollte nachfolgende Reihenfolge versucht werden einzuhalten

1. ***Sauber bauen***. Vermeide schlackernde Kabel. Der FC sollte vibrationsgedämpft verbaut sein. Prüfe ob den Gyro etwas berührt (**strikt vermeiden**). Sind alle Schrauben fest
2. **Versuche auf die aktuelleste BF-Version aufzusetzen.** Mache vor Deinen Änderungen ein Backup der aktuellen FW. Sichere Deine Konfiguration mit `diff all`. Neue BF-Version versprechen Bugfixings und häufig verbesserte Filter/Tunig-Möglichkeiten. Leider ist der Update immer mit Arbeit verbunden.
3. **Sender kalibrieren**. Im Receiver-Tab sollten für alle drei Achsen die Einstellungen zwischenn **1000** und **2000** liegen. Hintergrund ist, liegen die aktuellen Werte unter-/oberhalb wird das RC-Signal beschnitten oder gespreizt, beides sorgt dafür, dass die Signalverarbeitung nicht optimal ist.
4. **Prüfe im BF-Configurator die Lage des Copters.** Der grüne Pfeil symbolisiert **Vorne**. Neige den Copter nach unten (PITCH-Forward), der der simulierte Copter **muss** sich ebenfalls nach unten neigen. Wiederholen für alle Achsen. Der simulierte Copter **muss exakt** das gleiche tun als der echte Copter.

   > Das ist eine **sehr, sehr wichtige** Funktionsprüfung, ansonsten wird der Copter beim Erststart vermutlich direkt einen Salto schlagen

5.

Betaflight BF4.x Tuning-Parameters

Inhaltsverzeichnis

- Feedforward
    - Allgemeines
        - IN
        - OUT
    - Parameter
- VBat
    - Allgemeines
        - IN
        - OUT
    - Parameter
- RC-Command
    - Allgemeines
        - IN
        - OUT
    - Parameter
    - rc_smoothing_auto_smoothness (Default:10)
- Setpoint
    - Allgemeines
        - IN
        - OUT
    - Parameter
- ITerm Parameter
    - Allgemeines

Historie

Version	Datum	Inhalt
0.1	August 2020	initial
0.2	August 2020	neu strukturiert

Tuning-Parameter

Nachfolgende eine Reihe der wichtigsten Tuning-Variablen.

Viel mehr Details findet man hier: BF4.2-Tuning-Notes (https://github.com/betaflight/betaflight/wiki/4.2-Tuning-Notes)

> `ACHTUNG`
>
> > Bei einem Update von BF <4.2 bitte ***KEIN*** Restore von alten Werten die durch `diff all` gespeichert wurden, importieren. Fangt bei **NULL** an

Betaflight - Tuning-Tips

Weitere Tuning-Tips findest du im BF-Wiki der jeweiligen Versionen: * BF4.2-Tuning-Notes (https://github.com/betaflight/betaflight/wiki/4.2-Tuning-Notes) * BF4.1-Tuning-Notes (https://github.com/betaflight/betaflight/wiki/4.1-Tuning-Notes) * BF4.0-Tuning-Notes (https://github.com/betaflight/betaflight/wiki/4.0-Tuning-Notes) * BF4.0-Tuning-Notes (https://github.com/betaflight/betaflight/wiki/3.5-tuning-notes)

DSHOT RPM Telemetrie-Daten

Allgemeines

Ab BF 4.0 werden Telemetriedaten des ESCs ausgelesen und analysiert. Diese Informationen sind elementar wichtig für viele nachgelagerte Filtertechniken und für den PID-Controller. Voraussetzung ist, dass man für den ESC ein DSHOT-Protokoll ausgewählt hat

IN

ESC - **Beachten:** die aktuelle Firmware des ESCs muss RPM-Telemetrie-Daten verarbeiten können.

OUT

RPM-Daten pro Motor

Parameter

Diese Parameter können nicht direkt beeinflußt werden. Zu beachten sind welches **DSHOT** Protokoll verwendet wird.

Bedenke: bei DSHOT300 und einer 8k PIDLoop erhältst du nur jede zweite PID-Loop Daten zugesendet. Daraus folgt, du solltest das passende **DSHOT-Protokoll** auf Deine PID-Loop auswählen

- **DSHOT150**: empfohlen bei 2k PIDLoop
- **DSHOT300**: empfohlen bei 4k PIDLoop
- **DSHOT600**: empfohlen bei 8k PIDLoop
- **DSHOT1200**: 8k PIDLoop

Gyro Filter

Allgemeines

Der Gyro ist das zentrale Bauelement auf dem FC und stellt die aktuellen **IST** Flugdaten zur Verfügung. Diese Daten werden dann bezogen auf die **SOLL** Daten (Die RC-Commands) verrechnet, gefiltert dem PIDController zur Verfügung gestellt. Anschließend gemixt und den Motoren als neue.

Jeder GYRO besitzt werkseitig einen internen LowPass-Filter. Je nach Bautyp des Gyros unterscheiden sich wie gut dieser interne Filter tatsächlich ist.

Die Gyro-Filter Parameter umfassen folgende Filterarten

Bei den weiteren beschriebenen `Gyro Filtern` wird nicht nochmals auf ÌN/OUT`eingegangen.

IN

`gyro_scaled` Daten direkt aus dem Gyro.

OUT

Bereitstellung der Daten für nachgelagerte `DTerm-Filter` und als Mix-Daten für den `P-Controller` => `Vorab-Fehler P-Berechnung`

Gyro Filterarten

Es wird unterschieden zwischen den Gyro-RPM Filtern, einem Dynamic NotchFilter und Statich Notichfilter 1+2, einem Dynamsiche LowPassfilter und einem statischen Lowpassfilter.

- Gyro-RPM-Notch Filter

  `gyro_rpm_notch_harmonics=3`
  `gyro_rpm_notch_q=500`
  `gyro_rpm_notch_min=100`

- Dynamic-Notch Filter `dyn_notch_min_hz`
  `dyn_notch_max_hz`
  `dyn_notch_width_percent`
  `dyn_notch_q`

- Static Gyro-Notch Filter (1 und 2) `gyro_notch1_hz`
  `gyro_notch1_cutoff`
  `gyro_notch2_hz`
  `gyro_notch2_cutoff`

- Dynamic Gyro LowPass Filter `gyro_lowpass_type`
  `gyro_lowpass_hz`
  `dyn_lpf_gyro_min_hz`
  `dyn_lpf_gyro_max_hz`

- Static Gyro LowPass Filter `gyro_lowpass2_type`
  `gyro_lowpass2_hz`

Gyro Filter => GYRO-RPM Notch Filter

Allgemeines

Der Gyro-RPM Notch Filter nutzt die vom ESC zurückgegeben RPM-Daten und liegt als erste Filterstufe direkt hinter dem `gyro_scaled` Daten.

BF 4.1/4.2 Bidirectional DSHOT and RPM Filter Guide (https://github.com/betaflight/betaflight/wiki/Bidirectional-DSHOT-and-RPM-Filter#Tuning)

Parameter

Parameter	BFDefault	Bezeichnung
`gyro_rpm_notch_harmonics`	3	Schwingungen treten in wiederkehrenden Amplituden auf. Eine harmonische Schwingung kann durch eine Sinusfunktion beschrieben werden (https://de.wikipedia.org/wiki/Schwingung#Harmonische_Schwingung). **Das bedeutet, dass eine Vibration sich alle xHz wiederholt!** Betaflight generiert somit pro Motor 3 (Anzahl Harmonics) Notch-Filter, somit werden alle Motordaten durch 12 individuellen RPM-Notch-Filter analyisiert und schon vorgefiltert. Diesen Sachverhalt kann man in einer Blackboxauswertung sehr gut sehen (FFT-Spektrogram).
`gyro_rpm_notch_q`	500	Der Q-Faktor des Notchfilters gibt die Breite der Kerbe (Notch=Kerbe) an. Je größer die Zahl (max 1000) umso schmaler wird der Notchfilter. Der `Q-Faktor` wird auch als Güte-Faktur bezeichnet. Je höher die Güte-Faktor (`Q-Faktor`) desto geringer die Dämpfung, desto schmaler der Notch-Filter. Kleine Q-Faktoren vergrößern den RPM-Filter Delay - was unerwünscht ist
`gyro_rpm_notch_min`	100	Beschreibt die untere Grenzfrequenz des Notch-Filters

Gyro Filter => Dynamic-Notch Filter

Allgemeines

Dieser Filter ist dem RPM-Filter nachgelagert und filtern nochmals bestimmte Frequenzen aus. Ohne RPM-Filter wird der Filter als Doppel-Notch Filter betrieben, dies wird über `dyn_notch_width_percent` > 0 definiert. Ist dieser Wert 0, wird nur ein Notch-Filter erzeugt.

Das besondere an dynamic-notch-Filter ist, dass sie dynamisch sich an der aktuellen *RPM* des Systems orientieren und so laufend rund um den höchsten Frequenzbereich der Motor-Vibrationen arbeiten.

Grundsätzlich gilt, dynamische Notch-Filter haben eine geringere Latenzzeit als statische Notch-Filter. Bei einem gut abgestimmten Copter können andere Low-Pass Filter deaktiviert werden.

Parameter

Parameter	BFDefault	Bezeichnung
`dyn_notch_min_hz`		Beschreibt die untere Grenzfrequenz dieses Notch-Filters in Hz
`dyn_notch_max_hz`		Beschreibt die obere Grenzfrequenz dieses Notch-Filters in Hz
`dyn_notch_width_percent`		Beschreibt (wenn > 0), wie weit beide Notch-Filter voneinander getrennt sind. Der Prozentsatz berechnet sich aus der Breite des Notch-Filters.
`dyn_notch_q`		Q-Faktor des Notch-Filters. (siehe hierzu Beschreibung weiter oben

Gyro Filter => Static Gyro-Notch Filter (1 und 2)

Allgemeines

Zwei statische Notch-Filter für ein bestimmtes Frequenzband. Dieses Frequenzband wird während des Fluges nicht mehr angepasst (statisch).

Parameter

Parameter	BFDefault	Bezeichnung
`gyro_notch1_hz`		Center-Frequenz des Notch-Filters
`gyro_notch1_cutoff`		**todo**
`gyro_notch2_hz`		Center-Frequenz des Notch-Filters
`gyro_notch2_cutoff`		**todo**

Gyro Filter => Dynamic Gyro LowPass Filter

Allgemeines

Die Auswahl den Dyn-Notchfilters Frequenzbereiches kann über drei Auswahlmöglichkeiten voreingestellt werden * **LOW** : `dyn_lpf_gyro_max_hz` liegt bei 334hz oder ist 0 (deaktiviert) * **MEDIUM** : `dyn_lpf_gyro_max_hz` liegt bei 610hz * **HIGH** : `dyn_lpf_gyro_max_hz` liegt bei > 610hz

Die durchschnittlichen Werte für optimale Werte für diese Ranges liegen * **LOW** : 80-330hz (für Copter mit niedrigen Drehzahlen oder wenn Resonanzen in niedrigen Frequenzen auftreten * **MEDIUM** : 140-550hz (für gut eingestellte 5" Copter * **HIGH** : 230-800hz (für Copter mit hohen Drehzahlen 2,5" - 3")

Ab BF 4.0 wird zusätzlicher `dyn_notch_min_hz` Parameter zur Verfügung gestellt. Dieser Wert fängt den Bereich unterhalb des Dyn-LPF ab und hat seinen Default bei 150Hz.

Um 100Hz Peaks heraus zu filtern muss `LOW` aktiviert werden und der

Parameter

Parameter	BFDefault	Bezeichnung
`dyn_notch_min_hz`		
`gyro_lowpass_type`		LOW/MEDIUM/HIGH (siehe Beschreibung)
`gyro_lowpass_hz`		Static Gyro LPF, sind dyn_lpf gesetzt, dann ist der Static LPF deaktiviert
`dyn_lpf_gyro_min_hz`		untere Grenzfrequenz es DynLPF
`dyn_lpf_gyro_max_hz`		obere Grenzfrequenz des DynLPF

Gyro Filter => Static Gyro LowPass Filter

Allgemeines

IN

Throttle-Daten

Parameter

Parameter	BF	Default	Bezeichnung
gyro_lowpass2_type	4.0	PT1/ BIQUAD	
gyro_lowpass2_hz	4.0		unter Grenzfrequenz des LPF in Hz, wenn auf 0, dann ist der LPF deaktiviert

DTerm Filter

Allgemeines

Der DTerm-Filter besitzt eine Reihe von Parametern die dazu genutzt werden, das DTerm-Ausgangssignal zu bearbeiten und von Störungen (Vibrations-Frequenzen zu befreien). **Wichtig:**: Der DTerm des PID-Controllers verstärkt Vibrationen, daher ist es wichtig, dass dieses Signal möglichst frei von Störungs- / Vibrationssignalen ist.

D verstärkt höhere Frequenzen, der D-Anteil wird aber dringend benöigt um Vibrationen zwischen 30-80hz (Z.B. Propwash) auszugleichen. Das bedeutet wir benötigen soviel wie möglich D-Anteil bis 100hz und so wenig wie möglich über 100hz. DTerm-Filter sollten immer in der ersten Stufe als BIQUAD und in der zweiten Stufe als PT eingestellt werden.

DTerm Filter Daten sind zeitabhängig (`d/dt`)

Folgende DTerm-Filter werden genutzt: * **Dynamic D lowpass**

```
```
dterm_lowpass_type
dyn_lpf_dterm_min_hz
dyn_lpf_dterm_max_hz
dyn_lpf_dterm_curve_expo
```
```

- **Static D lowpass** `dterm_lowpass2_type` `dterm_lowpass2_hz`
- **Static D notch** `dterm_notch_hz` `dterm_notch_cutoff`

IN

Daten kommen aus den `Gyro Filter` Berechnungen

OUT

Daten gehen direkt an den `D-Controller`

DTerm => Dynamic D lowpass

Allgemein

Dynamischer Lowpass filter für den DTerm

Parameter

Parameter	BF Default	Bezeichnung
`dterm_lowpass_type`	4.0	PT1 / BIQUAD, sollte auf BIQUAD für LPF 1 stehen, LPF2 = PT1
`dyn_lpf_dterm_min_hz`	4.0	LowPass min Hz
`dterm_lowpass2_hz`	4.0	LowPass2 min Hz

DTerm => Static D notch

AllgemeiHz

Parameter

Parameter	BFDefault	Bezeichnung
`dterm_notch_hz`		
`dterm_notch_cutoff`		

Feedforward

Allgemeines

Feedforward ist dem PID-Controller nachgelagert und unabhängig vom PID. FF verstärkt bzw. wirkt auf Deine Stickbewegung und hilft den Motoren schneller zu reagieren.

Mehr Infos zu Feedforward

- Feedforward BF 4.1 (https://github.com/betaflight/betaflight/wiki/4.1-Tuning-Notes#feed-forward-boost)
- Feedforward 2.0 BF 4.2 (https://github.com/betaflight/betaflight/wiki/Feed-Forward-2.0)

IN

OUT

Parameter

Parameter	BF	Default	Bezeichnung
`ff_boost`	4.1	15	Der Booster verstärkt den gesamten FF-Wert aber zu einem sehr frühen Zeitpunkt und veringert damit ein Delay
`ff_interpolate_sp`	4.1	2 (Average_2)	siehe Anhang der Tabelle
`ff_spike_limit`	4.1	50	Es wird eine effiziente verzögerungsfreie Dämpfungsmethode verwendet, die Erhöhung des boosts durch Spikes verringert bzw. vermeidet. Liegt der normale Boost-Wert unter dem limit wird er durchgelassen, alles weitere, hohe Boosts die durch Spikes verursacht werden, werden gedämpft.
`ff_max_rate_limit`	4.1	100	`ff_max_rate_limit` unterbricht den Feedforward, wenn die Geschwindigkeit mit dem der Stick bewegt wird wahrscheinlich sein Ende des mechanischen Verfahrensweges erreicht. Dadurch wird ein Überschwingen gerade bei Beginn von Flips reduziert.
`ff_smooth_factor`	4.2	37	Glättungsfaktor für einkommende Signale. Funktioniert wie ein LowPassfilter. 0 = keine Glättung, höhere Werte wie der Defaultwert, erhöhen auch die Latzenzeit und wirkt dem eigentlich FF-Forward entgegen

`ff_interpolate_sp` **Ausprägung** * OFF * ON * AVERAGE-2 : passt für die meisten Copter & Freestyler * AVERAGE-3 : * AVERAGE-4 :

VBat

Allgemeines

Ab BF 4.2 wir mit VBAT-SAG-Kompensation Weitere Informationen: (https://github.com/betaflight/betaflight/wiki/4.2-Tuning-Notes#dynamic-battery-sag-compensation)

IN

OUT

Parameter

Parameter	BF	Default	Bezeichnung
vbat_sag_compensation	4.2	100	100% Kompensation der Batterieentladung
vbat_pid_gain	?	OFF	alte Version sollte immer OFF sein

RC-Command

Allgemeines

IN

Receiver-Daten Signal

OUT

Daten werden mit den eingstellten `Rates` verrechnet und gelten dann als das angewendete `RCCommand-Eingangs-Signal`

Parameter

Parameter	BFDefault	Bezeichnung
`rc_interpolation`		
`rc_interp`		
`rc_inter_ch`		
`rc_inter_int`		

rc_smoothing_auto_smoothness (Default:10)

rc-smoothing-auto-smoothness setzt wie glatt der die RC-Signale sein sollen. Größere Werte erhöhen die Glättung vergrößern aber das RC-Delay. 10 ist optimal für die meisten allgemeinen Flüge. Racer bevorzugen 8 oder sogar 5. Das RC-Delay nimmt zwar ab, dafür können die Motor-Signale etwas unruhiger werden.

Setpoint

Allgemeines

In der weiteren Berarbietung der Eingangssignale werden diese als `Setpoint` bezeichnet und spiegeln das RC-Signal wieder allerdings durch eine Reihe von Parametern geglättet

IN

Aufbereitetes RC-Command Signal

OUT

Daten die mittels `setpoint_smoothing` nochmals geglättet werden werden an * `Vorab-Fehler P-Berechnung` * `d/dt`

Parameter

Setpoint/Setpoint smoothing beinhaltet eine Reihe von Parametern die das eigentlich Signal nochmals aufbereiten.

Parameter	BFDefault	Bezeichnung
`rc_smoothing_type`		
`rc_smoothing_auto_smoothness`	10	setzt wie glatt der die RC-Signale sein sollen. Größere Werte erhöhen die Glättung vergrößern aber das RC-Delay. 10 ist optimal für die meisten allgemeinen Flüge. Racer bevorzugen 8 oder sogar 5. Das RC-Delay nimmt zwar ab, dafür können die Motor-Signale etwas unruhiger werden
`rc_smoothing_input_Hz`		
`rc_smoothing_input_type`		

ITerm Parameter

Allgemeines

ITerm Parameter dienen dazu das I-Signal die PID-Controllers entweder vor der Bearbeitung von ITerm oder nach ITerm zu beeinflußen.

Insbesondere sollen hier Peaks im ITerm eliminiert werden.

Die Nachfolgende Tabelle beinhaltet zwei zusätzliche Spalten **IN** und **OUT** sie bezeichnen woher die Daten kommen (`IN`) und wer sie verwertet (`OUT`)

Parameter	BFIN	OUT	Default	Bezeichnung
`iterm_windup`	MIXER	ITerm		iterm_windup ist eine alte Methode zur Unterdrückung der iTerm-Akkumulation, wenn das Motordifferential einen benutzerdefinierten Schwellenwert überschreitet. In BF 4.2 wirkt `iterm_windup` nur noch auf YAW `iterm_relax` hat `iterm_windup` zur Vermeidung von I-Anhäufungen auf Mini-Quads weitgehend ersetzt. `iterm_relax` ist hauptsächlich zur Vermeidung von bounce-backs
`iterm_relax`	ITERM	PID_SUM		
`iterm_relax_type`	ITERM	PID_SUM		
`iterm_relax_cutoff`	ITERM	PID_SUM		Wenn der Pilot eine Änderung der Drehgeschwindigkeit anfordert, die für das Quad zu schnell ist, eilt das Gyrosignal dem Sollwert (SetPoint) hinterher und daraus entsteht ein mehr oder minder großes Fehlersignal. Der I-Term versucht nun diesen Fehler zu akkumulieren (aufsummieren) und versucht diese zu korrigieren. `iterm_relax` versucht nun diese Akkumulation zu kontrollieren. Reicht der `iterm_relax` nicht aus, sammeln sich die ITerm Fehler immer mehr an. Stoppt nun der Pilot seine Stickbewegung (z.B. in einem Flip), dann wird all der angesammelte ITerm-Fehler eine Gegenbewegung des Copters verursachen (BounceBack), der dann langsam ausklingen wird bis er wieder auf 0 ist. `iterm_relax` für Flip & Rolls und `item_windup` für YAW, versuchen diese Bouncebacks zu kontrollieren und abzumildern. `item_relax_cutoff begrennzt die ITerm` Akkumulation.
`iterm_rotation`	GYRO-Filter	ITerm		

DMin - Handling

Inhaltsverzeichnis

{{TOC}}

Allgemeines

Mit DMin ist es nun möglich unterschiedliche Werte für D zu haben, je nachdem was der Copter gerade macht.

Im Normalflug erlaubt uns DMin mit reduzierten D-Werten zu fliegen und der DMax-Wert wird für schnelle Bewegungen genutzt (z.B. bei Fast-Roles, Flips, Propwash)

Außerdem bleiben die Motoren kühler

Wenn DMin im Konfigurator aktiviert wird, wird D in DMax umbenannt und es gibt eine neue Spalte DMin

DMin ist vom Profil abhängig (genau wie der D-Wert)

`d_min_boost_gain` steuert die Empfindlichkeit des Boost-Effekts (also wenn von DMin der DMax Wert verwendet werden soll, gibt der Booster an wie schnell das gehen soll.

Was bringen geringere D-Werte

Vorteil

- weniger Vibrationen, kühlere Motoren
- Besseres Verhalten der Motoren bei Vollgas
- D-Wert bezogenes Oszillieren wird verringert

Nachteil

- mehr Propwash
- Größeres Überschießen und Bounce-Backs
- P Oszillation bei schnellen Mannövern
- Langsame und Lowlevel Oszillationen bei smoothen Flügen

Setup für den Erstflug

Default für DMin R23, P25, Y0

Beachte

> wenn DMin aktiviert ist, wird der reguläre DMax Wert nur dann genutzt, wenn schnelle Mannöver (z.B. Flips/Rolls) geflogen werden, in langsameren Flügen wird `DMIN` verwendet

Prüfen des D Wertes im Flug

- Anzeige im OSD, `set debug_mode=D_MIN` und im OSD die Anzeige `debug2 on-screen`. Die Anzeige zeigt dir den 10fachen Wert. Beispiel: Anzeige 350 = 35D
- Über die Log-Aufzeichnung. Auch hier `set debug_mode=D_MIN` DEBUG2 im Blackboxexplorer zeigt dir unittelbar D für die ROLL-Achse, DEBUG3 für PITCH.

`DEBUG2` und `DEBUG3` zeigen den D-Wert vor `TPA`.

`DEBUG0` zeigt die Gyro-Anteil

Hinweis

> Bei Verwendung von `DMin` erhöht sich die CPU-Last ein wenig. Implementiert ist ein Biquad-Filter und ein PT1-Filter.

Parameter

Parameter	BF Default	Bezeichnung
`d_min`	0/xx	0 = disabled DMin, Werte > 0 entsprechend dem DMin-Wert. Wenn DMin deaktiviert ist, ist aktuell immer der DMax (D) der genutzte Wert für D
`d_min_advanced`	20	beschleunigt des boost-effekt, wenn sich Änderungen am `setpoint` ergeben (oder Gyro-Veränderungen). Wird unmittelbar bei Veränderungen durchgeführt bevor der Copter überhaupt die Bewegung ausführt. Der Wert kann zum Overshot beitragen, 0 deaktiviert diese Funktion und sollte bei Racern verwendet werden (und bei den meisten Coptern).
`d_min_boost_gain`		Verstärkungsfaktor, wie schnell bei schnellen Bewegungen D angepasst werden soll. 30-35 ist für normale Copter gut geeignet, 40-45 für wirklich sauber gebaute Freestyler. Wenn Propwash das Hauptproblem sind und die Motoren kühl sind, dann DMax erhöhen und den gain-Faktor ebenfalls (moderat!)

Fine-Tuning des DMin Wertes geht nur über eine Blackboxauswertung. Für diese Anlayse sollte der Debug-Mode `D-MIN` ausgewählt werden. Nur dann sieht man die aktuellen `D_Min` Werte im Flug

LunaX

(c) August 2020, Ritterhude