

# dbdis App

ab Version 3.41 und höher

08.02.2021

© Dieter Birklbauer

(Designer, Programmierer, Anleitungsschreiber, Videomacher, Übersetzer und Anwender dieser App)

## 1. Verwendungszweck:

Die dbdis App hat 3 wesentliche Aufgaben:

1. Die übersichtliche Aufbereitung von Telemetriedaten und die Ausgabe von optischen und akustischen Warnhinweisen bei überschreiten von Grenzwerten.
2. Die automatische Führung eines Flugbuches, die pro Flug die wichtigsten Kennzahlen in einer Zeile einer Textdatei abspeichert.
3. Die Speicherung und Dokumentation von Zykluszahl und entnommener Kapazität aller in Gebrauch befindlichen Akkus

## 2. Allgemeines:

Sehr vieles an der App ist intuitiv zu erfassen, und alle die schon mal eine App installiert haben werden keine Schwierigkeiten haben sich damit auch ohne Anleitung zurecht zu finden. Für jene denen auch die dahinterliegende Funktionalität interessiert und für alle anderen ist die Anleitung als Nachschlagewerk gedacht.

## 3. Installation:

Die dbdis App kann auf jedem Jeti Sender mit farbigem Display verwendet werden.

Dazu sind 4 Dateien erforderlich, welche hier herunter geladen werden können:

<https://github.com/ribid1/dbdis>

Im App Verzeichnis auf dem Sender unter ../Apps ist die Datei dbdis.lua und der Ordner „dbdis“ mit den Dateien lang.jsn, Screen.lua, Form.lua und Form2.lua oder die jeweils kompilierten .lc Dateien davon zu kopieren. Ob man die .lua oder die .lc Dateien verwendet ist Geschmackssache, der Vorteil der lua Dateien liegt darin, dass man schnell mal Programmparameter verändern kann um diese auszuprobieren ohne die Dateien neu kompilieren zu müssen.

Im Sender ist unter Zusatzfunktionen/Benutzerapplikation die dbdis auszuwählen. Die .lc Version erkennt man daran dass sie weniger Speicherplatz benötigt.

Im Sendermenü unter Stoppuhren/Sensoren die Telemetrieanzeige aufrufen. Dort das „+“ auswählen und unter „Lua“ findet man die Einträge dbdis -1 – Benutzername und dbdis -2 – Benutzername. Je nachdem wie viele Seiten man anzeigen möchte wählt man entweder nur einen oder beide Einträge. Grundsätzlich ist die Installation bereits abgeschlossen und die App kann verwendet werden, ich würde aber empfehlen zumindest einen Schalter, z.Bsp für die Flugzeit und/oder einen Sensor für die Anzeige zuzuweisen.

## 4. Konfiguration:

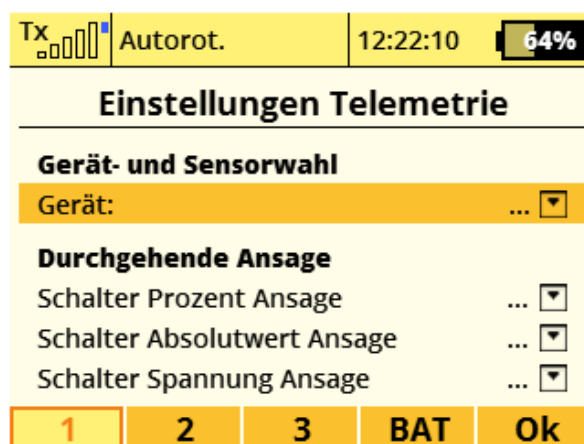
Im Hauptmenü findet man ganz unten den neuen Eintrag mit dem Namen „dbdis“, wo man auf 4 Seiten Zugriff auf sämtlichen Einstellungsmöglichkeiten der App hat. Die App ist so aufgebaut dass alle wichtigen Einstellungen bereits vorkonfiguriert sind. Jede weitere Einstellung die man trifft erhöht die Funktionalität und die Anzahl der Anzeigeboxen in den Fenstern 1 und 2, bzw. passt die App immer besser seinen persönlichen Bedürfnissen an.

Videos dazu findest du hier: <https://youtu.be/Zso-oRc5-Y8> und hier: <https://youtu.be/Qo8YZW3CySw>

### 4.1 Sensoren, Schalter und Grenzwerte: (Seite 1)

#### 4.1.1 Gerät- und Sensorwahl:

Als Sensor kann jeder sich am Markt befindliche Sensor ausgewählt werden, wie z. Bsp. S32, Mtag, RfiD, Unsisens, Muli, GPS, usw....



Hat man einen Sensor ausgewählt erweitert sich das Menü um den Punkt Sensorkategorie und alle entsprechenden Sensorwerte:

Tx	Autorot.	12:22:10	64%
<b>Einstellungen Telemetrie</b>			
<b>Gerät- und Sensorwahl</b>			
Gerät:	S32		
Kategorie:	Alle		
Akkuspannung:	...		
Motorstrom:	...		
mAh verbraucht:	...		
1	2	3	BAT Ok

Kategorie dient dazu etwas Ordnung in unzähligen Anzeigewerte zu bringen, wer sich aber nicht sicher ist wo sein Anzeigewert sein könnte wählt am besten „Alle gängigen“. In dieser Kategorie finden sich alle Sensorwerte bis auf die in „Muli“ und „2.Motor“ da diese nur für wenige Piloten interessant sein dürften:

Tx	Autorot.	12:22:21	64%
<b>Option wählen</b>			
<b>Alle gängigen</b>			
Elektr. getrieben			
Sprit getrieben			
Empfänger			
gemischt			
Muli			
2. Motor			
Esc			Ok

Anschließend braucht man nur noch einem Anzeigewert einen bestimmten Sensorwert zuordnen:

Tx	Autorot.	12:22:10	64%
<b>Einstellungen Telemetrie</b>			
<b>Gerät- und Sensorwahl</b>			
Gerät:	S32		
Kategorie:	Alle		
Akkuspannung:	U Akku V		
Motorstrom:	I motor A		
mAh verbraucht:	mAh mAh		
save	2	3	BAT Ok

Ein Sensorwert kann beliebig vielen Anzeigewerten zugeordnet werden. Wurde ein Anzeigewert bereits belegt sieht man das daran dass beim Wechsel des Sensors dieser nicht mehr belegt werden kann:

Tx	Autorot.	12:22:50	64%
<b>Einstellungen Telemetrie</b>			
Gerät:	RFID-Sensor ▾		
Kategorie:	Alle gängigen ▾		
Akkuspannung:			
Motorstrom:			
mAh verbraucht:			
BEC Strom:	... ▾		
1	2	3	BAT Ok

In der **Kategorie 2. Motor** können folgende Sensorwerte zugewiesen und zur Anzeige gebracht werden:

Tx	Autorot.	12:22:21	64%
<b>Einstellungen Telemetrie</b>			
<b>Gerät- und Sensorwahl</b>			
Gerät:	S32 ▾		
Kategorie:	2. Motor ▾		
Akkuspannung 2:	U Akku V ▾		
Motorstrom 2:	I motor A ▾		
mAh verbraucht 2:	mAh mAh ▾		
Füllstand 2:	... ▾		
RPM 2:	RPM uni rpm ▾		
Temp 2:	U Akku V ▾		
Akku 2 ID:	... ▾		
Status 2:	... ▾		
<b>Durchgehende Ansage</b>			
1	2	3	BAT Ok

Auf der Hauptansicht werden nur jene Anzeigeboxen zur Anzeige gebracht die auch einen Sensorwert zugeordnet bekommen haben. Wird zum Beispiel der Anzeigewert BEC Strom nicht belegt so wird diese Anzeigebox auch später nicht angezeigt auch wenn man diese im Layout unter Layoutkonfiguration ausgewählt hat. Es wird also wirklich nur das angezeigt was auch Sinn macht.

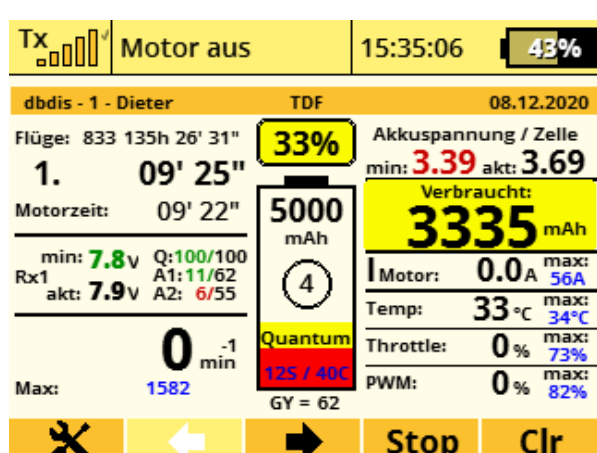
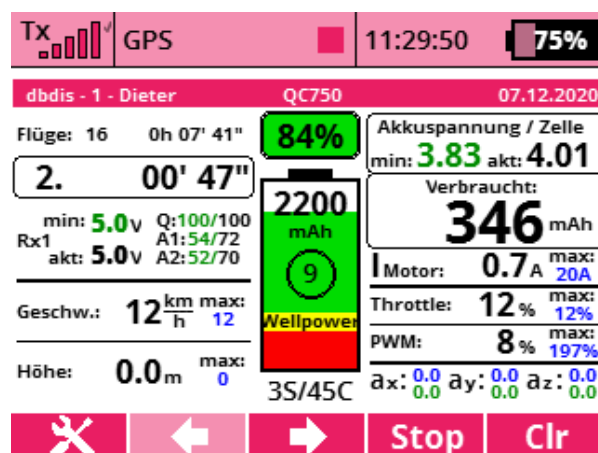
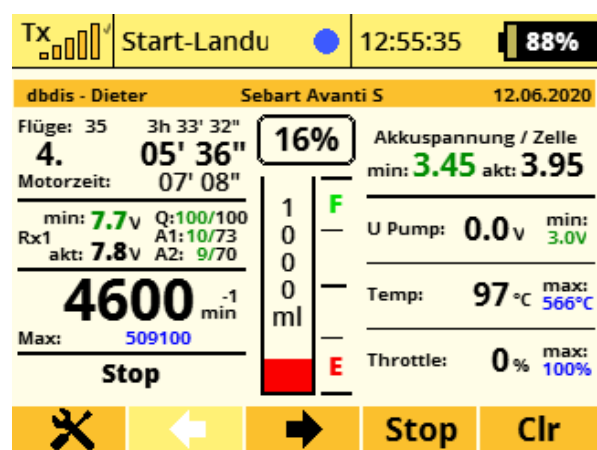
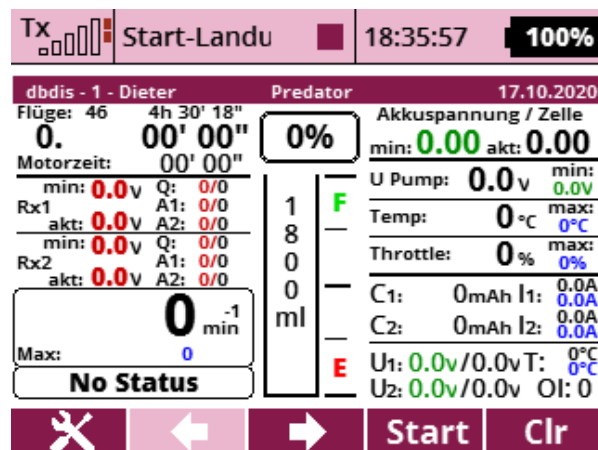
Die derzeit verfügbaren Anzeigeboxen stellen eine vorläufige Auswahl dar, die laufend erweitert wird, aber bereits sehr umfangreich ist. Auch die Designs der Anzeigeboxen sollen als Vorlage dienen und können geändert oder ergänzt werden. Die derzeitigen Boxen siehst du anhand der Screenshots:

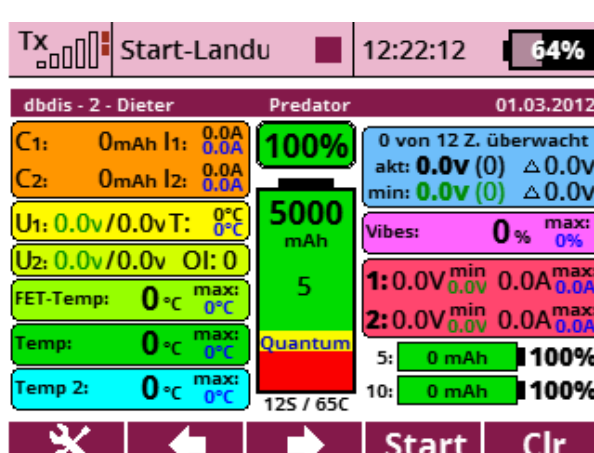
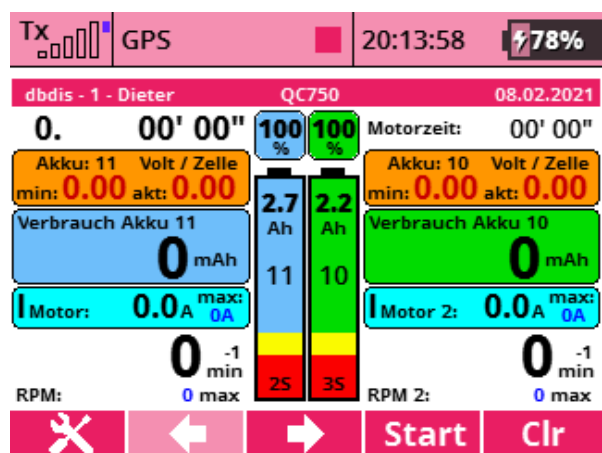
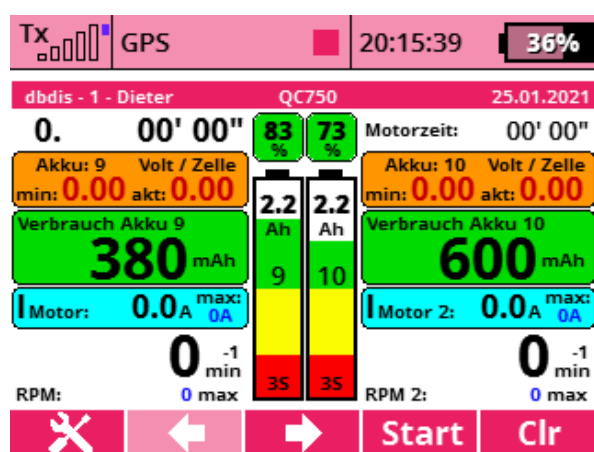
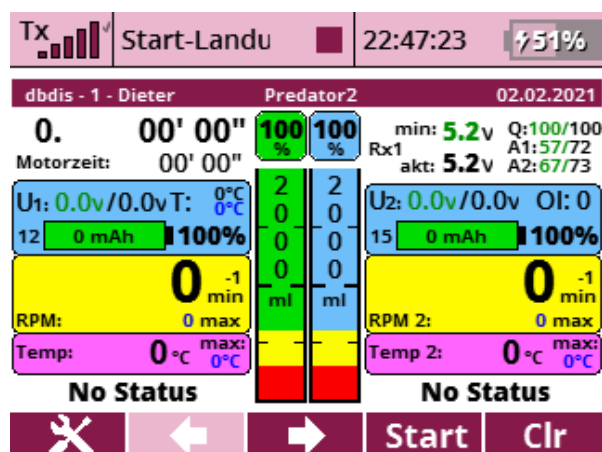
Als generelle Eigenschaft der Anzeigewerte ist der **Maximalwert blau (Himmel)**, der **Minimalwert grün (Wiese)** und Wert der überschritten wurde **rot** dargestellt:

## Quick Tip:

Um schnell zu den restlichen Konfigurationseinstellungen zu gelangen ohne jedes mal lange nach unten scrollen zu müssen einfach Taste 1 betätigen und die „Geräte- und Sensorwahl“ klappt zu.

### 4.1.2 Anzeigeboxen:





Die letzte Anzeige stellt die Farben 1 – 9 dar, welche in der Standardeinstellung der dbdis\_config.json hinterlegt sind. Diesen 9 Farbnummern und auch der Farbnummer 0 kann eine beliebige Farbe zugeordnet werden. Diese Farbnummern sind für alle Modelle gleich. Es kann allerdings für jedes Modell und jede Anzeigebox eine individuelle Farbnummer zugeordnet werden.

Im Beispiel hat „FET-Temp“ die Farbnummer „4“ zugeordnet, in einem anderen Modell kann man „FET-Temp“ die Farbnummer „8“ zuordnen dann wäre die Box violett, oder man ändert die Farbe „4“ auf violett dann werden alle Anzeigeboxen mit der Farbe „4“ violett dargestellt.

Eine Besonderheit stellt die Anzeigebox „UsedCapacity“ (Verbrauch) dar, wird dieser Box eine Farbe zugeordnet dann wird auch entweder der „grüne“ Bereich der Tank- oder der Batterieanzeige in dieser Farbe dargestellt. Wird dieser Box keine Farbe zugeordnet dann bleibt der „grüne“ Bereich auch grün. Speziell bei 2 motorigen Antrieben kann man so eine Unterscheidung der Anzeigen darstellen.

### 4.1.3 Anzeigewerte mit besonderen Eigenschaften:

#### 4.1.3.1 Signalstärke A1 und A2:

Die Signalstärke wird ungefiltert dargestellt so wie sie vom Sender intern verarbeitet wird, und stellt daher eine höhere Auflösung dar, im Endeffekt ist aber wohl Geschmacks- und Gewohnheitssache.

Ab dem Wert 6 wird er rot dargestellt was der Signalstärke 2 im gängigen Anzeigeformat entspricht.

Ist das öfter mal der Fall oder für beide Antennen gleichzeitig sollte man sich mal die log Dateien diesbezüglich ansehen.

#### 4.1.3.2 Akkuspannung:

Um die Akkuspannung leichter „lesbar“ zu machen wird nicht die Gesamtspannung, sondern die Spannung geteilt durch die Zellenanzahl angezeigt. Außerdem wird die Anfangsspannung eines Akkus dazu benutzt um zum einen den Ladezustand des Akkus zu berechnen und zum anderen dazu um zu erkennen ob Akku frisch aufgeladen ist oder bereits verwendet wurde.

Dieser Punkt ist etwas gewöhnungsbedürftig, kann aber auch ausgeschaltet werden, ich möchte den jedenfalls auf keinem Fall mehr missen! So kann es nie mehr passieren dass ein halb entladener Akku zu tief entladen wird. Bei altersschwachen Akkus die den Vollzustand nie mehr ganz erreichen werden so gleich zu Beginn einige mAh Verbrauch angezeigt was dem Alterungszustand des Akkus Rechnung trägt.

Der 2. Punkt funktioniert so: hat der Akku eine Spannung unter den eingestellten 4,08V/Zelle wird er als „gebraucht erkannt“, die App sieht dann in der Akkudatenbank nach wie hoch die zuletzt entnommene Kapazität war und addiert diese zu dem weiteren Verbrauch. Betrug diese allerdings 0mAh, also es wurde nichts hinterlegt, berechnet die App die entnommene Kapazität aufgrund der Akkuspannung. Am Ende des Fluges wird die nun entnommene Kapazität zu der bereits beim letzten Flug entnommenen Kapazität dazu addiert und beim jeweiligen Akku hinterlegt. (Die Zykluszahl wird nicht erhöht) Es wird immer nur die tatsächliche entnommene Kapazität gespeichert, eine aufgrund der Akkuspannung berechnete bereits entnommene Kapazität wird nie gespeichert.

Liegt die Akkuspannung über 4,08V/Zelle so wird der Akku als voll erkannt und die beim letzten Flug entnommene Kapazität auf 0 gesetzt, sobald die Spannung unter die 4,08V fällt und ein Flug gewertet wurde wird der Akku als verwendet erkannt und Zykluszahl um 1 erhöht. Man kann also den Akku beliebig oft An- und Abstecken ohne dass die Zykluszahl erhöht wird.

Die Schwellspannungen können in der dbdis\_config.json eingestellt werden.

#### 4.1.3.3 mAh verbraucht:

Wird dieser Wert zugeordnet werden automatisch das Akkusymbol in Mitte des Anzeigefensters, sowie die Anzeigebox „Verbraucht“ zur Anzeige gebracht. Außer man verwendet die CalCa-Elec, dann muss dieser Wert frei bleiben und die Daten der CalCa App werden übernommen.

Sowohl die Prozentanzeige als auch die Absolutwertanzeige verändern ihre Farbe je nach den eingestellten Grenzwerten und werden gelb oder rot.

Der angezeigte Verbrauch ist wie bereits unter „Akkuspannung“ erwähnt von mehreren Faktoren abhängig, wie ob der Akku voll geladen wurde, und ob ein bereits verwendeter Akku die bereits verbrauchten mAh auch hinterlegt hat. Übrigens sollte man mal vergessen haben den richtigen Akku „einzuschalten“ und kommt erst nach dem Flug drauf, so lässt sich dass immer noch ändern, sofern noch kein Log Eintrag erfolgt ist, sprich der Resetschalter noch nicht betätigt wurde oder der Empfänger noch nicht ausgeschaltet wurde.

Diese Kombination der Anzeige aus gespeicherten Werten, berechneten Werten und Sensorwerten, und die automatische Erkennung des Akkuzustandes wurde bisher in noch keiner Software umgesetzt und ist einzigartig!

#### **4.1.3.4 Füllstand:**

Ist als Pendant zu mAh verbraucht zu sehen, wird dieser Wert zugeordnet erscheint in der Mitte die Füllstandsanzeige, auch dieser Wert muss frei bleiben wenn man die CalCa-Gas verwenden möchte.

Auch hier wechselt die Prozentanzeige je nach Tankinhalt zw. grün, gelb und rot.

#### **4.1.3.5 Status 1 und 2:**

Dienen zur Anzeige eines Telemetriewertes, vorzugsweise für „Zustände“.

In Kombination mit der Tstatus App (T steht in diesem Fall für Telemetrie-Status) kann diese zur Umwandlung des Zahlenwertes in einen sprechenden Text verwendet und angezeigt werden, wie zum Beispiel bei Betriebszuständen und Meldungen von Turbinen.

#### **4.1.3.6 ax, ay, az:**

Da die Momentanwerte hier wenig Sinn machen, werden lediglich die maximalen und minimalen Werte in der entsprechenden Box angezeigt.

#### **4.1.3.8 FET-Temp, Temp und Temp\_2:**

FET-Temp findet man in der Empfänger Kategorie, Temp unter gemischt und Temp\_2 in der Kategorie 2. Motor.

FET-Temp wird zusätzlich noch in der Box „U1\_and\_Temp“ angezeigt

Allen 3 Temperaturanzeigen ist gemeinsam dass der Name der Anzeige in der dbdis\_config.jsn eingestellt werden kann.



#### 4.1.3.9 RPM und RPM 2:

Ähnlich wie die Temperaturanzeigen kann auch bei den Drehzahlboxen der Name in der Anzeigebox mittels der dbdis\_config.json eingestellt werden kann.

Zusätzlich kann für RPM 2 in den Einstellungen auf Seite 1 ein Drehzahlfaktor angegeben werden mit dem der Wert multipliziert wird, um z. Bsp von der Motordrehzahl auf die Haupt- oder Heckrotordrehzahl umzurechnen.

Sobald die Drehzahlanzeige 5 stellig wird ändert sie zur besseren Lesbarkeit ihr Aussehen, dies ist vor allem bei Turbinenantrieben sehr nützlich.

#### 4.1.3.8 schwächste Zelle:

Hier kann man sich entweder die schwächste Zelle eines neueren Muli EX, oder von ein oder mehreren älteren Muli -Sensoren, oder aber auch die schwächste Zelle aus bis zu 6 Einzelspannungen anzeigen lassen. Die Anzeige passt sich den zugeordneten Sensorwerten automatisch an.

Tx Autorot. 12:22:12

**Einstellungen Telemetrie**

Überwachte Zellen: ... ▼

Spannungsdifferenz: ... ▼

Schwächste Zelle 1: ... ▼

U(min) 1: ... ▼

Schwächste Zelle 2: ... ▼

U(min) 2: ... ▼

Schwächste Zelle 3: ... ▼

U(min) 3: ... ▼

Schwächste Zelle 4: ... ▼

U(min) 4: ... ▼

Schwächste Zelle 5: ... ▼

U(min) 5: ... ▼

Schwächste Zelle 6: ... ▼

U(min) 6: ... ▼

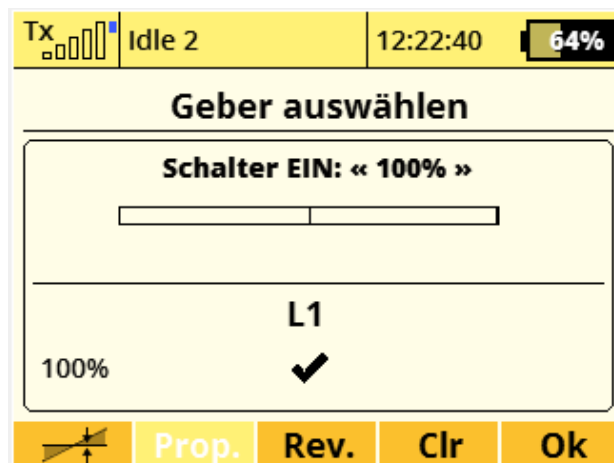
save 2 3 BAT Ok

#### 4.1.4 Speichern der Konfiguration:

Verändert man einen Wert in der Konfiguration erscheint links unten anstelle der 1 ein „save“, drückt man den entsprechenden Taster verschwindet es wieder und im dbdis Ordner am Sender wird eine Datei mit dem Namen : Modellname\_config.json erzeugt. Diese Datei dient lediglich zur Sicherung der Konfiguration, falls man diese mal erneut laden muss oder in einem anderen Modell die gleichen Einstellungen verwenden möchte. Die eingestellten Werte bleiben auch ohne dieses Sichern beim Ausschalten des Senders gespeichert.

In dieser config Datei sind alle Sensorzuordnungen, alle Grenzwerte und vor allem auch alle zugeordneten Schalter und Schalterstellungen gespeichert. Einzige Ausnahme sind Telemetrieschalter, diese werden zwar auch in der config Datei gespeichert (für die Zukunft), können aber nicht wieder hergestellt werden. Jeti stellt da leider noch keine Möglichkeit zur Verfügung. Als Workaround könnte man den Umweg über einen logischen Schalter nehmen. Aber wie gesagt dies betrifft nur die config Datei, ansonsten werden auch die Telemetrieschalter zuverlässig gespeichert.

Eine 2. Ausnahme betrifft den Schaltpunkt Geber Ein:



Auch dieser Wert kann leider noch nicht in einer config Datei gespeichert werden. Ich muss allerdings dazu sagen, dass ich eine Veränderung dieses Wertes noch nie benötigt habe.

Lädt man also eine config Datei von einem anderen Modell muss man diese 2 Punkte berücksichtigen und eventuell nachjustieren.

Um einen Schalterwert zuverlässig in eine config Datei zu speichern ist es erforderlich dass der Schaltwert der „Rev.“ Einstellung entspricht, d. h. entweder den Schalter vor dem Bestätigen mit Ok nicht mehr verändern oder das ganz linke Symbol drücken. Bei federbelasteten Schaltern ist es daher notwendig diese zu halten während mit Ok bestätigt wird.

Grundsätzlich wurde das Speichern der Einstellungen in noch keiner App so konsequent umgesetzt!

#### 4.1.5 Durchgehende Ansage:

Hier werden die Schalter für folgende Ansagen eingestellt:

Die Prozent Ansage sagt entweder den Akku- oder Tankfüllstand in % an.

Die Absolutwert Ansage sagt entweder den Akkufüllstand in mAh oder den Tankfüllstand in ml an.

Die Spannungsansage sagt den Spannungswert pro Zelle in Volt an.

Der Zeitintervall zw. den Ansagen beträgt 10 Sekunden.

Sind 2 Energiequellen vorhanden (2 Akkus oder 2 Tanks) so wird derjenige Wert angesagt, welcher am niedrigsten ist.

The screenshot shows a handheld device interface. At the top, there's a status bar with 'Tx' and signal strength bars, 'Autorot.' with a blue dot, the time '12:22:35', and a battery icon with '64%'. Below this is a menu titled 'Einstellungen Telemetrie'. Under the heading 'Durchgehende Ansage', there are three items: 'Schalter Prozentansage', 'Schalter Absolutwertansage', and 'Schalter Spannungsansage', each followed by a dropdown arrow. Below these is the section 'Sprachdateien wählen' with 'Modellansage:' followed by a dropdown arrow. At the bottom, there are five buttons: '1', '2', '3', 'BAT', and 'Ok'.

#### 4.1.6 Sprachdateien wählen:

Hier könne die Sprachdateien für die jeweiligen Ansagen ausgewählt werden.

Die Modellansage wird 1x beim Einschalten des Senders oder beim Modellwechsel angesagt.

Die Alarm Spannung wird 6x bei Unterschreiten der eingestellten Warnspannung angesagt.

Der Vor-Alarm wird 2x bei unterschreiten des eingestellten %Wertes für den Füllstand bzw. die Rest Akkukapazität angesagt.

Der Haupt-Alarm wird 4x bei unterschreiten des eingestellten %Wertes für den Füllstand bzw. die Rest Akkukapazität angesagt.

Wird keine Sprachdatei angegeben so wird beim Vor- und Hauptalarm der jeweilige aktuelle Prozentsatz angesagt. Bei der Spannung wird nichts angesagt. Die Anzahl der Alarme kann in der dbdis\_config.json eingestellt werden:

This screenshot shows the same handheld device interface as the previous one. The 'Einstellungen Telemetrie' menu is open, and the 'Sprachdateien wählen' section is highlighted. It shows 'Modellansage:', 'Alarm Spannung:', 'Vor-Alarm Kapazität:', and 'Haupt-Alarm Kapazität:', each with a dropdown arrow. Below this section is a new heading 'Batterie / Tank Einstellungen'. The bottom buttons '1', '2', '3', 'BAT', and 'Ok' are still visible. The status bar at the top shows the time as '12:22:56'.

#### 4.1.7 Batterie / Tank Einstellungen:

Für Akku 1 und 2 können hier die jeweiligen ID's eingetragen und mittels Schalter ausgewählt werden. Wird nur eine ID hinterlegt spielt es keine Rolle ob bei Akku1 oder Akku2, diese wird immer ausgewählt. Verwendet man einen Akkusensor so spielen diese Eintragungen ebenfalls keine Rolle, es wird immer die ID vom Sensor verwendet, und sollte der Akku nicht in der Datenbank zu finden sein, so wird er angelegt. Es können nur Akkus ausgewählt werden welche zuvor in der Datenbank angelegt wurden, so kann man nicht versehentlich eine falsche ID angeben.

Verwendet man in seinem Modell 2 Energiequellen so darf der Schalter Akku 2 nicht belegt werden, denn dann weiß die App dass beide Akkus oder Tanks verwendet werden und teilt die Verbrauchsanzeige in 2 unabhängige Anzeigen, sofern auch 2 Verbraucher zugewiesen wurden.

Verwendet man 2 Motoren und weißt den Akku Schalter zu, so werden die beiden Verbräuche, egal ob Strom oder Sprit addiert und angezeigt.

Vor- und Hauptalarm gelten sowohl für die Tankkapazität als auch die Akkukapazität  
Der Spannungsalarm gilt immer nur pro Zelle. Man kann das natürlich auch umgehen, indem man Zellenzahl 1 beim Akku eingibt, dann ist hier aber der Wert entsprechend abzuändern, ansonsten wird der Akku viel zu tief entladen.

Bei 2 Energiequellen wird der Alarm immer dann ausgelöst wenn es vom niedrigsten Wert der beiden Anzeigen unterschritten wurde.

Tx	Autorot.	12:22:33	64%	
<b>Einstellungen Telemetrie</b>				
<b>Batterie / Tank Einstellungen</b>				
Akku 1 ID:	2			
Akku 2 ID:	3			
Tankvolumen 1:	2000 ml			
Tankvolumen 2:	2000 ml			
Schalter für Akku/Tank 2:	Sh			
1	2	3	BAT	Ok

Tx	Autorot.	12:22:40	64%	
<b>Einstellungen Telemetrie</b>				
Tankvolumen 1:	2000 ml			
Tankvolumen 2:	2000 ml			
Schalter für Akku/Tank 2:	Sh			
Vor-Alarm bei (Tank) Kapazität:	30 %			
Haupt-Alarm bei (Tank) Kapazität:	20 %			
Alarm bei Spannung:	3.50 V			
1	2	3	BAT	Ok

#### 4.1.8 Zeitschalter:

Der Schalter für die Flugzeit beginnt nur dann zu laufen, wenn auch ein Empfängersignal anliegt. Der Schalter kann beliebig oft ein- und ausgeschaltet werden. Sobald die Mindestdauer für die Flugzählung erreicht ist, und die Flugzeit gestoppt wird, so wird auch der Flug gezählt und die Flugzeit zur Gesamtflugzeit des Modells hinzu addiert. Der Schalter kann auch nachträglich beliebig oft gestartet werden, erhöht aber nicht mehr die Fluganzahl sondern nur mehr die Flugzeit. Wird der Empfänger ausgeschaltet während die Stoppuhr läuft, und Mindestdauer der Fluges wurde erreicht so wird der Flug trotzdem gespeichert.

Der Schalter für die Motorlaufzeit startet diesen Timer und der Schalter für „Motor AN“ überwacht ob der Motor Schalter betätigt wurde und zeigt dies im Telemetriefenster an. Beide Boxen sind nur dann sichtbar wenn auch Schalter zugewiesen wurden.

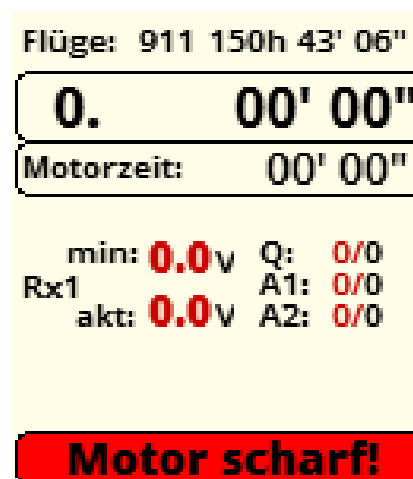
Der Resetschalter hat mehrere Aufgaben. Zum einen bewirkt er ein löschen der minimalen und maximalen Sensorwerte, und liest die Akkudaten neu ein, falls sich diese geändert haben sollten (Akkudaten von einem Sensor sind automatisch immer aktuell). Zum anderen bewirkt er ein Rücksetzen der Stoppuhr auf 0. Wurde die Flugzeit bereits stoppt und die Mindestflugzeit erreicht, so wird der Flug auch abgespeichert und die Logdatei mit allen Flugaten ergänzt. Ein neuerliches Starten der Flugzeit, zählt nun den Flugzähler für einen neuen Flug hoch, und ein neuer Flug kann beginnen.

Wurde die Mindestflugzeit noch nicht erreicht, so wird auch nichts abgespeichert und der Flug wird nicht gezählt.

Wird der Resetschalter betätigt während die Stoppuhr noch läuft und die Mindestflugzeit wurde bereits überschritten, so folgt eine Sicherheitsabfrage ob man den Flug wirklich löschen möchte oder ob er abgespeichert werden soll. (Nur um sicher zu gehen 😊)

Um einen Logeintrag zu generieren gibt es also 2 Möglichkeiten, entweder den Resetschalter betätigen, oder den Empfänger ausschalten, beides funktioniert aber nur wenn die eingestellte Mindestflugdauer erreicht wurde.

Alle Sensordaten bleiben auch nach dem Ausschalten des Empfängers so lange im Telemetriefenster erhalten bis der Reset Schalter betätigt oder der Empfänger wieder eingeschaltet wird, dann werden die Werte zurückgesetzt.



#### 4.1.9 Gyrokanal und RPM Faktor:

Hier kann man sich den Gyrowert für den Heckkreisel, so wie er in der V-Stabi Software steht anzeigen lassen. Dies funktioniert im Prinzip auch für andere Kreiselssysteme, bei Bedarf kann ich da gerne entsprechende Konfigurationseinstellungen vorsehen, ähnlich wie bei meiner **Stabiwertrechner** App.

Der RPM 2 Faktor multipliziert, wie bereits erwähnt, den RPM 2 Wert mit einem bestimmten Faktor und bringt ihn zur Anzeige.

#### 4.1.10 Historie:

Die Anzahl der Flüge und die Gesamtflugzeit können hier einfach und schnell angepasst werden. Die Werte sind einer txt-Datei unter dem jeweiligen Modellnamen im Ordner dbdis dauerhaft gespeichert und können auch dort geändert werden.

Tx Autorot. 12:22:14 64%

### Einstellungen Telemetrie

**Gyrokanal und RPM Faktor**

RPM 2 Faktor: 1.00 ▾

Kanal (17 = kein Gyro) 10 ▾

**Historie**

Flüge 962 ▾

Gesamtflugzeit (min) 9050 min ▾

1 2 3 BAT Ok

#### 4.1.11 Konfig Datei:

Sollten einmal Einstellungen verloren gehen, oder möchte man sich Einstellungen von einem anderen Modell laden so kann man dies indem die entsprechende Datei ausgewählt wird und man Button 1 (Load) betätigt. Wie bereits erwähnt werden alle Sensor Einstellungen, Ansagen, Schalter und deren Stellungen, Grenzwerte, usw. wiederhergestellt, bzw. von einem anderen Modell übernommen.

Alle Einstellungen der aktuellen Konfiguration werden immer beim Betätigen des „save“ Buttons unter dem Namen: Modellname\_config.json im dbdis Ordner gespeichert.

Tx Motor aus 12:22:41 64%

### Einstellungen Telemetrie

**Gyrokanal**

Kanal (17 = kein Gyro) 17 ▾

**Historie**

Flüge 907 ▾

Gesamtflugzeit (min) 9042 min ▾

Konfig. Datei: TDF\_config.json ▾

Load 2 3 BAT Ok

## 4.2 Layoutkonfiguration: (Seite 2 und 3)

Die Layouteinstellungen für die beiden Telemetrieseiten sind ident, mit dem einzigen Unterschied dass die Schaltflächen „save“ und „load“ bei der Seite 2 auf dem 3 Butten erscheinen.

Sobald man etwas ändert erscheint die save Schaltfläche und die Einstellungen können zusätzlich zum Sender internen Speicher auch in einer jsn Datei mit folgender Namenskonvention gespeichert werden:

**Modellname\_d1.jsn** für die 1. Seite und **Modellname\_d2.jsn** für die 2. Seite.

Das Laden der Einstellungen geschieht analog wie bei den Konfigurationseinstellungen, einfach in der obersten Zeile die entsprechende Datei suchen (alphabetisch sortiert) und den „load“ Button drücken. Es spielt dabei keine Rolle ob man die für Seite 1 oder Seite 2 lädt, beide sind untereinander austauschbar.

Man sieht auf den ersten Blick welche Anzeigeboxen bereits Sensorwerte zugewiesen bekommen haben, diese sind in normaler Schrift dargestellt und haben in Klammer die Boxenhöhe in Pixel dabeistehen. Alle anderen Anzeigeboxen sind klein geschrieben.

Unter der Spalte „Sep“ kann man die Stärke des Trennstriches zw. den Boxen einstellen, 0 bedeutet kein Trennstrich, und -1 bedeutet um die Box wird ein Rahmen gezeichnet. Gibt man -2 zwei ein so wird ein Rahmen um diese und auch um die vorhergehende Box gezeichnet und eventuell mit Farbe gefüllt. Bei -3 wird ein Rahmen um 3 Anzeigeboxen gezeichnet, usw.. Dazwischen kann man dann nach wie vor Trennstriche angeben. Es kann also auch über die ganze Spalte ein Rahmen zeichnet und ev. mit Farbe gefüllt werden, und z. Bsp. dazwischen eine Box in einer anderen Farbe, usw.. Man braucht sich auch nicht darum zu kümmern wie die nachfolgende Box aussieht, hat diese z.Bsp einen Rahmen wird der Trennstrich automatisch weggelassen auch wenn er nicht 0 ist. Genau so verhält es sich mit der letzten dargestellten Box auch hier wird der Trennstrich automatisch vom Programm weggelassen, man braucht wirklich nur das Notwendigste einstellen und kann ein und dieselbe Layout Konfiguration für die unterschiedlichsten Modell verwenden.

Die Spalte „Dist“ bestimmt die Distanz zw. den Boxen in Pixel. Möchte man etwas sehr knapp untereinander stehen haben so kann man auch negative Werte eingeben, ist halt Geschmackssache. Wählt man -9, so wird der restliche vorhandene Platz, zwischen allen Boxen mit dieser Einstellung aufgeteilt. Bei der letzten Box bestimmt dieser Wert den Abstand zum untersten Rand. Wie viele Pixel der berechnete Abstand zw. allen Boxen mit -9 hat, sieht man im Klammerwert bei der jeweiligen Spalte. So beträgt im unteren Beispiel der berechnete Abstand bei der linken Spalte 6 Pixel und bei der rechten Spalte 2 Pixel. Somit ist es ein leichtes seine eigenen Vorlieben für die Telemetriefenster umzusetzen.

Die letzte Spalte (Col Colour) bestimmt die Hintergrundfarbe der Anzeigeboxen mit Rand. Es können 10 verschiedene Farbnummern zugeordnet werden. Standardmäßig sind 9 Farben vordefiniert, die jedoch mit beliebigen Farben überschrieben werden können. Die Farbnummer 0 ist standardmäßig keiner Farbe zugeordnet, dies kann aber auch geändert werden, dann werden alle Boxen mit Rand in dieser Farbe dargestellt. Auf Seite 5 sieht man die „Standardfarben“.

Die Reihenfolge der Anzeigeböden kann man sehr einfach mittels der beiden Pfeile rechts unten verändern. Möchte man z.Bsp. die RxB Box in der rechten Spalte haben die sich in der Standardeinstellung unter „nicht verwendet“ befindet, so navigiert man zu dieser Zeile und wählt am einfachsten den nach oben zeigenden Pfeil. Man könnte auch den nach unten weisenden nehmen, nur Springt die Box dann zuerst zur linken Spalte und dann schrittweise nach unten bist man wieder in der rechten Spalte landet. Möchte man eine Zeile nicht dargestellt haben. So kann man diese mittels der Pfeile so lange verschieben bis sie in „nicht verwendet“ landet, oder unter den Konfigurationseinstellungen den zugewiesenen Sensorwert entfernen. Die Böden „FlightTime“ und „EngineTime“ werden, wie im Beispiel, nur angezeigt wenn die entsprechenden Schalter bereits zugewiesen wurden, auch das sieht man sogleich an der entsprechenden Schriftgröße.

Tx Start-Landru 12:22:02 64%

### Ansichtseinstellung Seite 2

Layout Datei: Predator\_d2.jsn

Linke Spalte: (1)	Sep:	Dist:	Col:
C1_and_I1 (16)	0	-9	1
C2_and_I2 (16)	-2	-9	1
U1_and_Temp (16)	-1	-9	2
Vario			

1 2 Load

Tx Start-Landru 12:22:05 64%

### Ansichtseinstellung Seite 2

Vario

U2_and_OI (12)	-1	-9	3
EngineOff			
FET_Temp (17)	-1	-9	4
Temp (17)	-1	-9	5
Temp_2 (17)	-1	-9	6

1 2 Load

Tx Start-Landru 12:22:38 64%

### Ansichtseinstellung Seite 2

Rechte Spalte (2)	Sep:	Dist:	Col:
weakest_Cell (38)	-1	-9	7
Vibes (17)	-1	-9	8
I_BEC			
U1_and_I1 (16)	0	-9	0
U2_and_I2 (16)	-2	-9	9
used Cap1 (16)	0	-9	0

1 2 3

Tx Start-Landru 12:22:38 64%

### Ansichtseinstellung Seite 2

nicht verwendet	Sep:	Dist:	Col:
Pump_voltage			
Current			
PWM			
Status2			
Volt_per_Cell			
Rx1Values (29)	2	-9	0
Speed			

1 2 3

Tx Start-Landru 12:22:12 64%

dbdis - 2 - Dieter Predator 01.03.2012

C1: 0mAh I1: 0.0A	100%	0 von 12 Z. überwacht
C2: 0mAh I2: 0.0A		akt: 0.0v (0) Δ 0.0v
U1: 0.0v/0.0v T: 0°C	5000 mAh	min: 0.0v (0) Δ 0.0v
U2: 0.0v/0.0v OI: 0	5	Vibes: 0 % max: 0%
FET-Temp: 0°C max: 0°C	Quantum	1: 0.0v min 0.0v 0.0A max: 0.0A
Temp: 0°C max: 0°C	12S / 65C	2: 0.0v min 0.0v 0.0A max: 0.0A
Temp 2: 0°C max: 0°C		5: 0 mAh 100%
		10: 0 mAh 100%

Start Clr



## 4.3 Konfiguration der Akkudaten: (Seite 4)

Nur zur letzten Einstellungsseite, den Akkudaten. Alle Einstellungen die hier getroffen werden wirken sich auf alle Modelle aus, da von überall auf die gleiche Datenbank, wieder eine json Datei zugegriffen wird. Zu Beginn wird diese Seite bis auf die Spaltenbezeichnungen leer sein, außer man hat einen Sensor für die Akkuwerte und diesen bereits zugewiesen und in Betrieb genommen, dann werden diese Akkus hier bereits aufscheinen. Ansonsten generiert man durch das + Symbol rechts unten einen neuen Eintrag und befüllt die Werte.

Die erste groß geschriebene Zeile jedes Akkus bestimmt dessen Eigenschaften und diese werden während des Betriebes auch nicht verändert.

Die 2. klein geschriebene Zeile gibt Auskunft über dessen Verwendung, wie die Anzahl der Zyklen, die zuletzt entnommene Kapazität und die gesamt entnommene Kapazität. Diese Werte können selbstverständlich auch angepasst und verändert werden.

Hat man einen Akku nicht länger in Verwendung so kann man diesen durch ändern seine ID auf -1 löschen. Dabei gehen auch die Daten wie die Zyklenzahl verloren.

Einen „save“ Button gibt es hier nicht, da alle Werte sofort in der Akkus.json Datei gespeichert werden.

Im Grunde genügt es völlig wenn ein „Akkusensor“ nur die Akku ID liefert, denn alle anderen Werte werden zuverlässig in der Akkus.json gespeichert und können dort auch mittels Texteditor geändert werden. Das hat auch mehrere Vorteile, z. Bsp. verwende ich die Akkus 4 und 5 in 3 verschiedenen Helis und den 5er zusätzlich in eine EDF Avanti, es ist also ein Ding der Unmöglichkeit, die Sensoren und den ID-Tag so zu platzieren dass er bei jedem Modell passt. Mit Hilfe der dbdis App kann ich die Akkus mit ID-Tags zuflastern, den die Anzahl der Zyklen wird immer nur in der Akkus.json gespeichert, es muss einzig und alleine die ID-Nummer immer die gleiche sein, das genügt. Und in der Avanti verwende ich sowieso immer nur den 5er also ist dort nicht mal ein Sensor notwendig, denn die Akku ID hinterlege ich in den Einstellungen für die Avanti. Somit macht ein Sensor nur dort Sinn, wo ich auch unterschiedliche Akkus verwende.

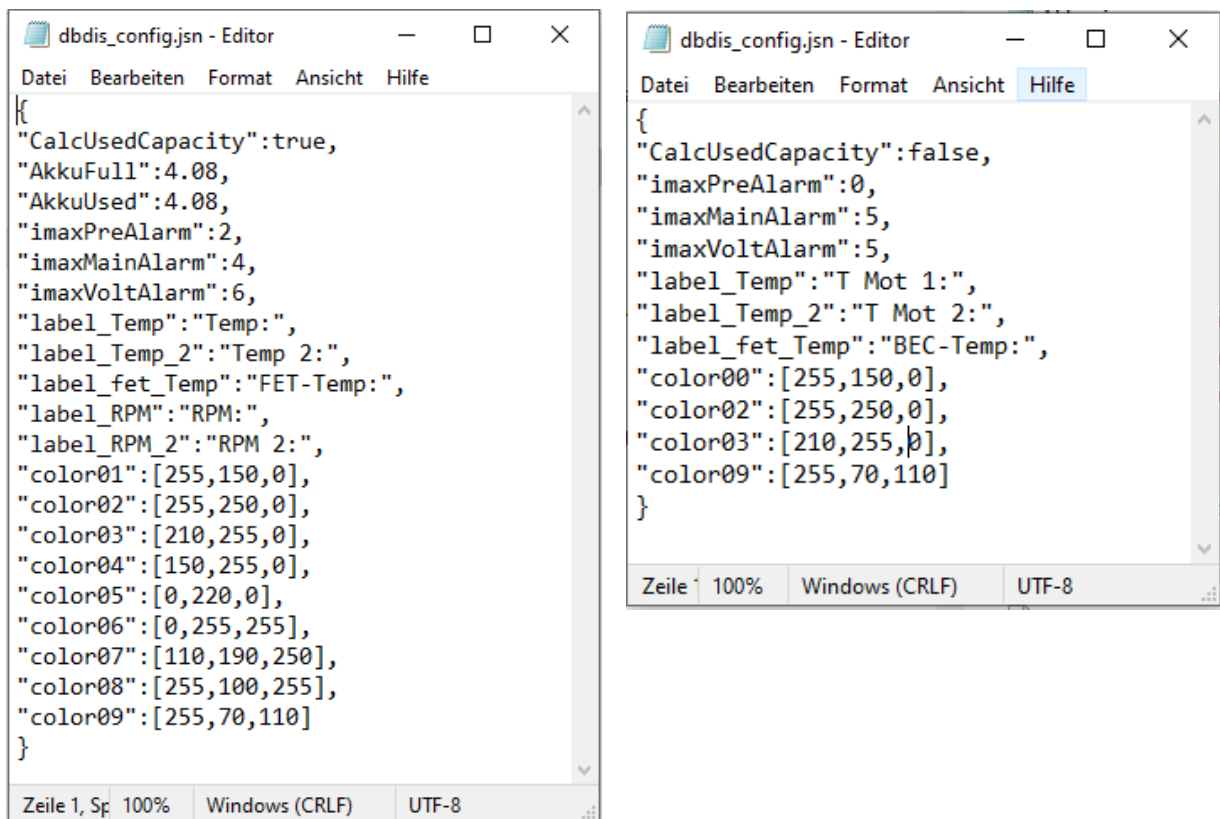
Tx		Motor aus	12:22:55	64%	
ID	Name	S	mAh	C	
Zyklen		Verbrauch zuletzt		Gesamtverbrauch	
4	Quantum	12	5000	40	
682 Zyklen		992 mAh		35 Ah	
5	Quantum	12	5000	65	
218 Zyklen		80 mAh		123 Ah	
9	Wellpower	3	2200	45	
0 Zyklen		500 mAh		0 Ah	
1	2	3	BAT		

Tx		Motor aus	12:22:42	64%
ID	Name	S	mAh	C
9	Wellpower	3	2200	45
0 Zyklen		500 mAh	0 Ah	
10	HV-Hyper	3	2400	5
0 Zyklen		0 mAh	1 Ah	
11	T-Avanti	2	2700	0
0 Zyklen		0 mAh	0 Ah	
ID = -1 löscht den Akku				
1	2	3	BAT	+

## 5. Globale Einstellungen:

Es gibt nun die Möglichkeit auch globale Einstellungen welche für alle Modelle gelten zu treffen. Möchte man dies tun so ist es erforderlich die Datei `dbdis_config.json` entsprechend seinen Bedürfnissen anzupassen und in den Ordner `... Apps\dbdis` am Sender zu kopieren. Möchte man wieder zu den Standardeinstellungen zurückkehren kann man diese Datei einfach wieder löschen. Beim Download von Github befindet sich diese Datei im Hauptverzeichnis und nicht im Unterordner damit man sich seine ev. getroffenen Einstellungen nicht versehentlich überschreibt.

Im Original sieht die Datei so aus und enthält die gleichen Einstellungen wie sie auch in der App hinterlegt sind, rechts daneben eine angepasste Datei:



```
{
  "CalcUsedCapacity":true,
  "AkkuFull":4.08,
  "AkkuUsed":4.08,
  "imaxPreAlarm":2,
  "imaxMainAlarm":4,
  "imaxVoltAlarm":6,
  "label_Temp":"Temp:",
  "label_Temp_2":"Temp 2:",
  "label_fet_Temp":"FET-Temp:",
  "label_RPM":"RPM:",
  "label_RPM_2":"RPM 2:",
  "color01":[255,150,0],
  "color02":[255,250,0],
  "color03":[210,255,0],
  "color04":[150,255,0],
  "color05":[0,220,0],
  "color06":[0,255,255],
  "color07":[110,190,250],
  "color08":[255,100,255],
  "color09":[255,70,110]
}
```

```
{
  "CalcUsedCapacity":false,
  "imaxPreAlarm":0,
  "imaxMainAlarm":5,
  "imaxVoltAlarm":5,
  "label_Temp":"T Mot 1:",
  "label_Temp_2":"T Mot 2:",
  "label_fet_Temp":"BEC-Temp:",
  "color00":[255,150,0],
  "color02":[255,250,0],
  "color03":[210,255,0],
  "color09":[255,70,110]
}
```

Wichtig dabei ist die Syntax nicht zu verändern, also alle Zeilen bis auf die letzte mit einem Beistrich zu beenden und die Bezeichner links vom Doppelpunkt nicht zu verändern. Außerdem dürfen die geschwungenen Klammern nicht gelöscht werden und es darf auch kein Kommentar ergänzt werden.

Man kann aber Zeilen herauslöschen und z.Bsp die Farbnummer „colour00“ hinzufügen, auch die Änderung der Reihenfolge der Zeilen spielt keine Rolle.

Ist man sich nicht sicher, einfach im Forum nachfragen, oder ausprobieren. Die Datei am besten mit dem Editor ändern.

## 6. Bedeutung der Einträge in der dbdis\_config.json:

**"CalcUsedCapacity":true**, ... wenn **true** dann wird der Akkufüllstand aufgrund der initialen Spannung berechnet, bei **false** wird er nicht berechnet.

**"AkkuFull":4.08**, ... bestimmt die Spannung pro Zelle ab wann der Akku als voll erkannt wird, über diesem Wert wird der in der Akkudatenbank hinterlegte Verbrauch ignoriert

**"AkkuUsed":4.08**, ... bestimmt die Spannung pro Zelle ab wann der Akku als gebraucht erkannt wird, unter diesem Wert wird die entnommene Kapazität in der Akkudatenbank gespeichert

**"imaxPreAlarm":2**, ... maximale Wiederholungszahl des Voralarms

**"imaxMainAlarm":4**, ... maximale Wiederholungszahl des Hauptalarms

**"imaxVoltAlarm":6**, ... maximale Wiederholungszahl des Spannungsalarms

**"label\_Temp":"Temp"**, ... Bezeichner in der Temp Box

**"label\_Temp\_2":"Temp 2"**, ... Bezeichner in der Temp\_2 Box

**"label\_fet\_Temp":"FET-Temp"**, ... Bezeichner in der FET\_Temp Box

**"label\_RPM":"RPM"**, ... Bezeichner in der Drehzahl Box

**"label\_RPM\_2":"RPM 2"**, ... Bezeichner in der Drehzahl 2 Box

**"color01":[255,150,0]**, ... Farbzordnung für Farbnummer 1

**"color02":[255,250,0]**, ... Farbzordnung für Farbnummer 2

.

.

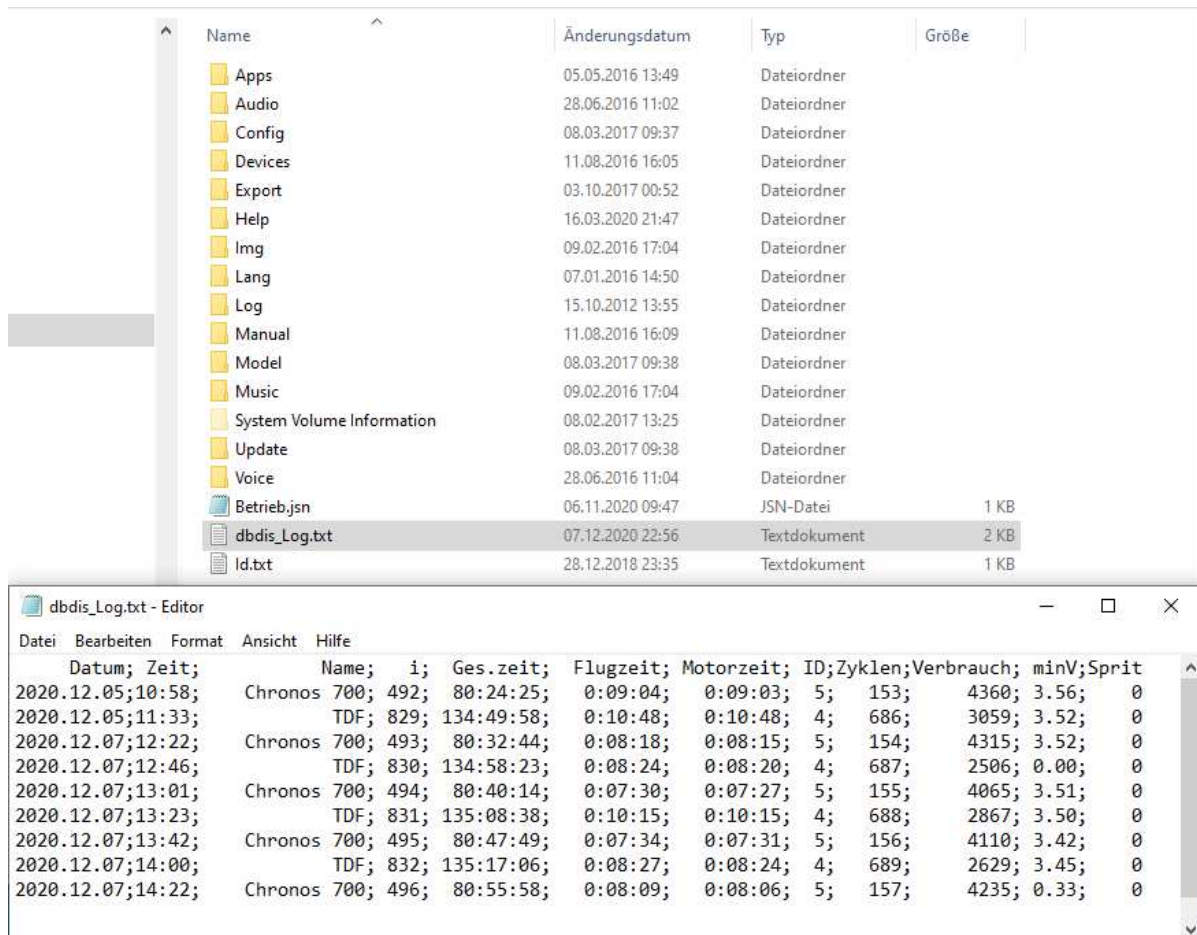
.

**"color09":[255,70,110]** ... Farbzordnung für Farbnummer 9

## 7. Die Log Datei:

Wie bereits erwähnt wird bei jedem Reset oder beim Ausschalten des Empfängers, sofern die eingestellte Flugdauer erreicht wurde ein Logeintrag generiert und in der „dbdis\_Log.txt“ gespeichert.

Diese Datei befindet sich im Root Verzeichnis am Sender und man sich diese auch sofort dort ansehen, wenn man möchte:



Name	Änderungsdatum	Typ	Größe
Apps	05.05.2016 13:49	Dateiordner	
Audio	28.06.2016 11:02	Dateiordner	
Config	08.03.2017 09:37	Dateiordner	
Devices	11.08.2016 16:05	Dateiordner	
Export	03.10.2017 00:52	Dateiordner	
Help	16.03.2020 21:47	Dateiordner	
Img	09.02.2016 17:04	Dateiordner	
Lang	07.01.2016 14:50	Dateiordner	
Log	15.10.2012 13:55	Dateiordner	
Manual	11.08.2016 16:09	Dateiordner	
Model	08.03.2017 09:38	Dateiordner	
Music	09.02.2016 17:04	Dateiordner	
System Volume Information	08.02.2017 13:25	Dateiordner	
Update	08.03.2017 09:38	Dateiordner	
Voice	28.06.2016 11:04	Dateiordner	
Betrieb.json	06.11.2020 09:47	JSON-Datei	1 KB
dbdis_Log.txt	07.12.2020 22:56	Textdokument	2 KB
Id.txt	28.12.2018 23:35	Textdokument	1 KB

Datum; Zeit;	Name;	i;	Ges.zeit;	Flugzeit;	Motorzeit;	ID;Zyklen;	Verbrauch;	minV;	Sprit
2020.12.05;10:58;	Chronos	700; 492;	80:24:25;	0:09:04;	0:09:03;	5; 153;	4360;	3.56;	0
2020.12.05;11:33;	TDF;	829;	134:49:58;	0:10:48;	0:10:48;	4; 686;	3059;	3.52;	0
2020.12.07;12:22;	Chronos	700; 493;	80:32:44;	0:08:18;	0:08:15;	5; 154;	4315;	3.52;	0
2020.12.07;12:46;	TDF;	830;	134:58:23;	0:08:24;	0:08:20;	4; 687;	2506;	0.00;	0
2020.12.07;13:01;	Chronos	700; 494;	80:40:14;	0:07:30;	0:07:27;	5; 155;	4065;	3.51;	0
2020.12.07;13:23;	TDF;	831;	135:08:38;	0:10:15;	0:10:15;	4; 688;	2867;	3.50;	0
2020.12.07;13:42;	Chronos	700; 495;	80:47:49;	0:07:34;	0:07:31;	5; 156;	4110;	3.42;	0
2020.12.07;14:00;	TDF;	832;	135:17:06;	0:08:27;	0:08:24;	4; 689;	2629;	3.45;	0
2020.12.07;14:22;	Chronos	700; 496;	80:55:58;	0:08:09;	0:08:06;	5; 157;	4235;	0.33;	0

Dies ist aber auf Dauer nach hunderten von Flügen nicht sehr übersichtlich, also habe ich ein kleines Makro für den einfachen Import in eine Excel Datei erstellt. Dazu die gewünschten Einträge in der txt Datei markieren und mit Strg + C in die Zwischenablage kopieren. Anschließend die **dbdis\_Log\_de.xlsm** Datei öffnen und zum ersten freien Feld in der A-Spalte navigieren. Danach braucht man nur noch das Makro „Datenimport“ anklicken und schon hat man alle Werte übersichtlich in Excel importiert.

Ein Video dazu gibt es hier: <https://youtu.be/opMr2ESBsgg>

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
	Datum	Zeit	Name	Flüge	Ges.zeit	Flugzeit	Motorzeit	ID 1	Zyklen	Verbrauch	minV	maxA	Sprit	ID 2	Zyklen 2	Verbrauch 2	minV 2	maxA 2	Sprit 2
93	2020.12.30	15:10	T-Rex 700	694	90:02:20	00:07:21	00:07:16	5	179	4307	3,56								
94	2020.12.30	16:30	Oxy 3	4	00:22:32	00:05:37	00:05:34	14	2	1550	0,00								
95	2020.12.31	14:31	Oxy 3	5	00:27:59	00:05:27	00:05:24	13	3	1550	0,00								
96	2020.12.31	14:41	Oxy 3	6	00:33:48	00:05:48	00:05:45	14	3	1550	0,00								
97	2020.12.31	16:06	Oxy 3	7	00:39:22	00:05:33	00:05:30	13	4	1550	0,00								
98	2020.12.31	16:17	Oxy 3	8	00:45:09	00:05:46	00:05:44	14	4	1550	0,00								
99	2021.01.02	14:15	Oxy 3	9	00:52:04	00:06:55	00:06:50	13	5	1550	0,00								
100	2021.01.02	14:29	Oxy 3	10	00:58:58	00:06:54	00:06:50	14	5	1511	0,00								
101	2021.01.02	15:53	Oxy 3	11	01:04:55	00:05:56	00:05:52	13	6	1384	0,00								
102	2021.01.02	16:03	Oxy 3	12	01:11:09	00:06:13	00:06:10	14	6	1451	0,00								
103	2021.01.03	13:40	Oxy 3	13	01:17:09	00:06:00	00:05:58	13	7	1409	0,00								
104	2021.01.03	13:48	Oxy 3	14	01:23:10	00:06:00	00:05:53	14	7	1390	0,00								
105	2021.01.03	14:19	T-Rex 700	695	90:09:29	00:07:09	00:07:05	5	180	3753	3,56								
106	2021.01.03	14:33	TDF	849	137:32:38	00:08:00	00:07:50	4	706	2363	3,46								
107	2021.01.03	14:51	T-Rex 700	696	90:15:54	00:06:24	00:06:20	5	181	3961	3,56								
108	2021.01.03	15:08	TDF	850	137:40:36	00:07:58	00:07:54	4	707	2352	3,49								
109	2021.01.05	15:46	Oxy 3	15	01:29:17	00:06:07	00:06:05	13	8	1429	0,00								
110	2021.01.05	15:56	Oxy 3	16	01:35:05	00:05:47	00:05:44	14	8	1435	0,00								
111	2021.01.25	15:18	T-Rex 700	697	90:23:19	00:07:25	00:07:08	5	182	4043	3,59	78							
112	2021.01.25	15:31	TDF	851	137:46:28	00:05:51	00:05:21	4	708	1743	3,53	45							
113	2021.01.25	15:52	T-Rex 700	698	90:30:45	00:07:25	00:07:12	5	183	3971	3,56	65							
114	2021.01.25	16:07	TDF	852	137:53:22	00:06:54	00:06:35	4	709	2171	3,43	43							
115																			
116																			

## 8. Zu guter Letzt:

Ich möchte mich vor allem bei db\_nichtgedacht bedanken, der die Vorlage zu dieser App lieferte und mich vor allem zu Beginn des Projekts mit zahlreichen Tipps unterstützte. Die Originalversion seiner App findet man hier: <https://github.com/nichtgedacht/JLog-Heli>

Mein Dank gilt auch denjenigen Piloten welche die App getestet, mir Anregungen gegeben und auf Fehler aufmerksam gemacht haben!

Wem die App gefällt und etwas zu seiner Unterstützung beitragen möchte, so kann er das gerne machen, indem er einer wohltätigen Organisation seines Vertrauens einen Beitrag leistet. Ich würde es ja sowieso nur in fliegendes Spielzeug umsetzen 😊, und so hätte die ganze Mühe und der viele Zeitaufwand auch etwas Gutes.

**Und das Wichtigste: Viel Spaß mit der App!!!**

Dieter