

dbdis App

Version 3.26

08.12.2020

© 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, usw....



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

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

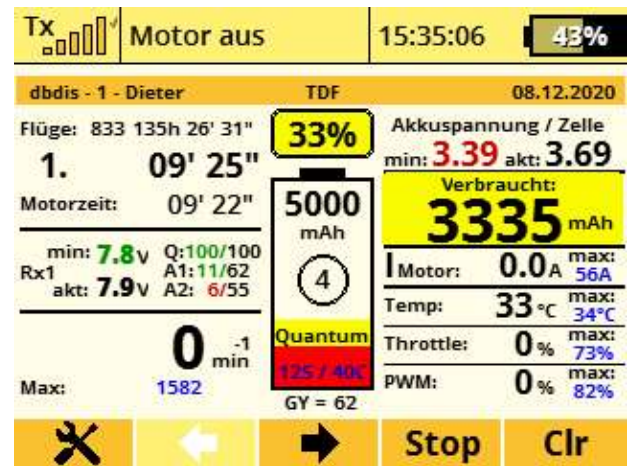
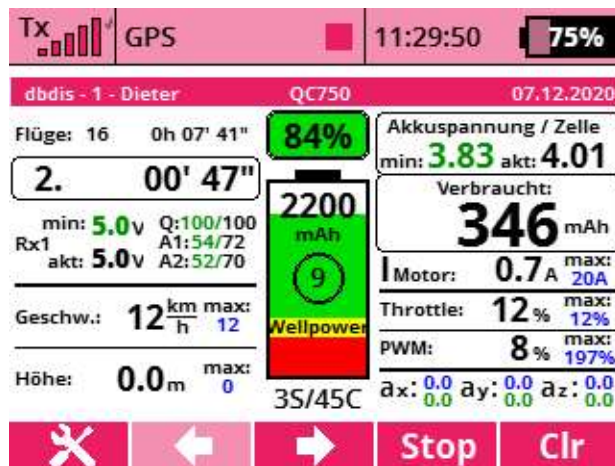
Kategorie dient dazu etwas Ordnung in unzähligen Anzeigewerte zu bringen, wer sich aber nicht sicher ist wo sein Anzeigewert sein könnte wählt am besten „Alle“:

Tx	Autorot.	12:22:17	64%
Option wählen			
Alle			
Elektr. getrieben			
Sprit getrieben			
Empfänger			
gemischt			
Esc			Ok

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

Tx	Autorot.	12:22:10	64%
Einstellungen Telemetrie			
Gerät- und Sensorwahl			
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:



4.1.3 Anzeigewerte mit besonderen Eigenschaften:

4.1.3.1 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.

Der erste Punkt ist etwas gewöhnungsbedürftig, 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 Zykuszahl 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 Screen.lua eingestellt werden:

```
local AkkuFull = 4.08 --bei einer Spannung über diesen Wert wird der Akku als voll erkannt
local AkkuUsed = 4.08 --bei einer Spannung unter diesem Wert wird der Akku als verwendet erkannt
```

4.1.3.2 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!

4.1.3.3 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.

4.1.3.4 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.

4.1.3.5 ax, ay, az:

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

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 Schalterpunkt 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.

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.

Tx 
Autorot. 
12:22:35  64%

Einstellungen Telemetrie

Durchgehende Ansage

Schalter Prozentansage ... 

Schalter Absolutwertansage ... 

Schalter Spannungsansage ... 

Sprachdateien wählen

Modellansage: ... 

1 2 3 BAT 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 Screen.lua eingestellt werden:

```

local imaxPreAlarm = 2  -- maximale Anzahl Voralarm
local imaxMainAlarm = 4 -- maximale Anzahl Hauptalarm
local imaxVoltAlarm = 6 -- maximale Anzahl voltage Alarm

```

Tx 
Autorot. 
12:22:56  64%

Einstellungen Telemetrie

Sprachdateien wählen

Modellansage: ... 

Alarm Spannung: ... 

Vor-Alarm Kapazität: ... 

Haupt-Alarm Kapazität: ... 

Batterie / Tank Einstellungen

1 2 3 BAT Ok

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 sofern in der Akkudatenbank der Akku existiert. 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.

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.

Tx Motor aus 12:22:49 64%					Tx Autorot. 12:22:56 64%				
Einstellungen Telemetrie					Einstellungen Telemetrie				
Batterie / Tank Einstellungen					Schalter für Akku 2: Sh <input checked="" type="checkbox"/>				
Akku 1 ID: 5					Tankvolumen: 2000 ml				
Akku 2 ID: 4					Vor-Alarm bei (Tank) Kapazität: 30 %				
Schalter für Akku 2: Sg <input checked="" type="checkbox"/>					Haupt-Alarm bei (Tank) Kapazität: 20 %				
Tankvolumen: 2000 ml					Alarm bei Spannung: 3.50 V				
Vor-Alarm bei (Tank) Kapazität: 30 %					Flugzeit				
save	2	3	BAT	Ok	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 Minstdauer 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 Minstdauer 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.

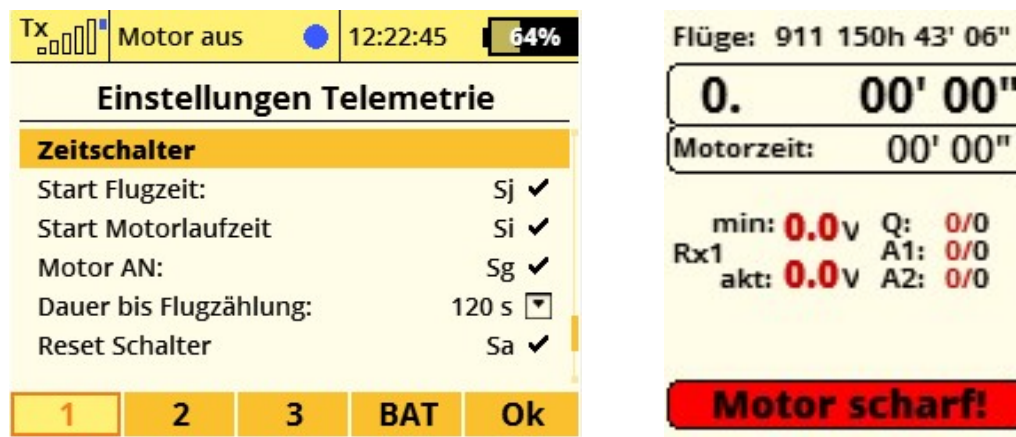
Der Resetschalter hat mehrere Aufgaben. Zum einen bewirkt er ein löschen der minimalen und maximalen Sensorwerte, und liest die Akkudaten neu ein, falls sich diese geändert haben sollten (Akkudaten von einem Sensor sind automatisch immer aktuell). Zum anderen bewirkt er ein Rücksetzen der Stoppuhr auf 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:

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

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 	Motor aus 	12:22:21	 64%
Einstellungen Telemetrie			
Gyrokanal			
Kanal (17 = kein Gyro)		17	
Historie			
Flüge		907	
Gesamtflugzeit (min)		9042 min	
Konfig. Datei:			
save	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.

Alle Einstellungen der aktuellen Konfiguration werden immer beim Betätigen des „save“ Buttons unter dem Namen: Modellname_config.jsn 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.jsn	
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. 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 Reihenfolge der Anzeigeboxen 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 Boxen „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 Motor aus 12:22:03 64%

Ansichtseinstellung Seite 1

Layout Datei: TDF_d1.jsn

Linke Spalte: (6)	Sep.:	Dist.:
TotalCount (9)	0	-9
FlightTime (17)	0	-9
EngineTime (12)	2	-9
Rx1Values (29)	2	-9

1 Load 3 ↑ ↓

Tx Motor aus 12:22:40 64%

Ansichtseinstellung Seite 1

RPM (37) 2 -9
 Altitude
 Vario
 Status

Rechte Spalte (2)	Sep.:	Dist.:
Volt_per_Cell (27)	2	-9

1 2 3 ↑ ↓

Tx Motor aus 12:22:36 64%

Ansichtseinstellung Seite 1

UsedCapacity (35)	-1	-9
Current (17)	2	-9
Pump_voltage I_BEC		
Temp (17)	1	-9
Throttle (17)	1	-9
PWM (17)	1	-9

1 2 3 ↑ ↓

Tx Motor aus 12:22:54 64%

Ansichtseinstellung Seite 1

PWM (17) 1 -9
 C1_and_I1
 C2_and_I2
 U1_and_Temp
 U2_and_OI
 Status2

nicht verwendet	Sep.:	Dist.:

1 2 3 ↑ ↓

Tx Motor aus 12:22:24 64%

Ansichtseinstellung Seite 1

nicht verwendet	Sep.:	Dist.:
Speed		
Rx2Values (29)	2	-9
RxBValues (29)	2	-9
ax_ay_az		

Wenn Dist. = -9 dann wird der Abstand berechnet
 Wenn Sep. = -1 dann wird das Feld umrandet

1 save 3 ↑ ↓

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.

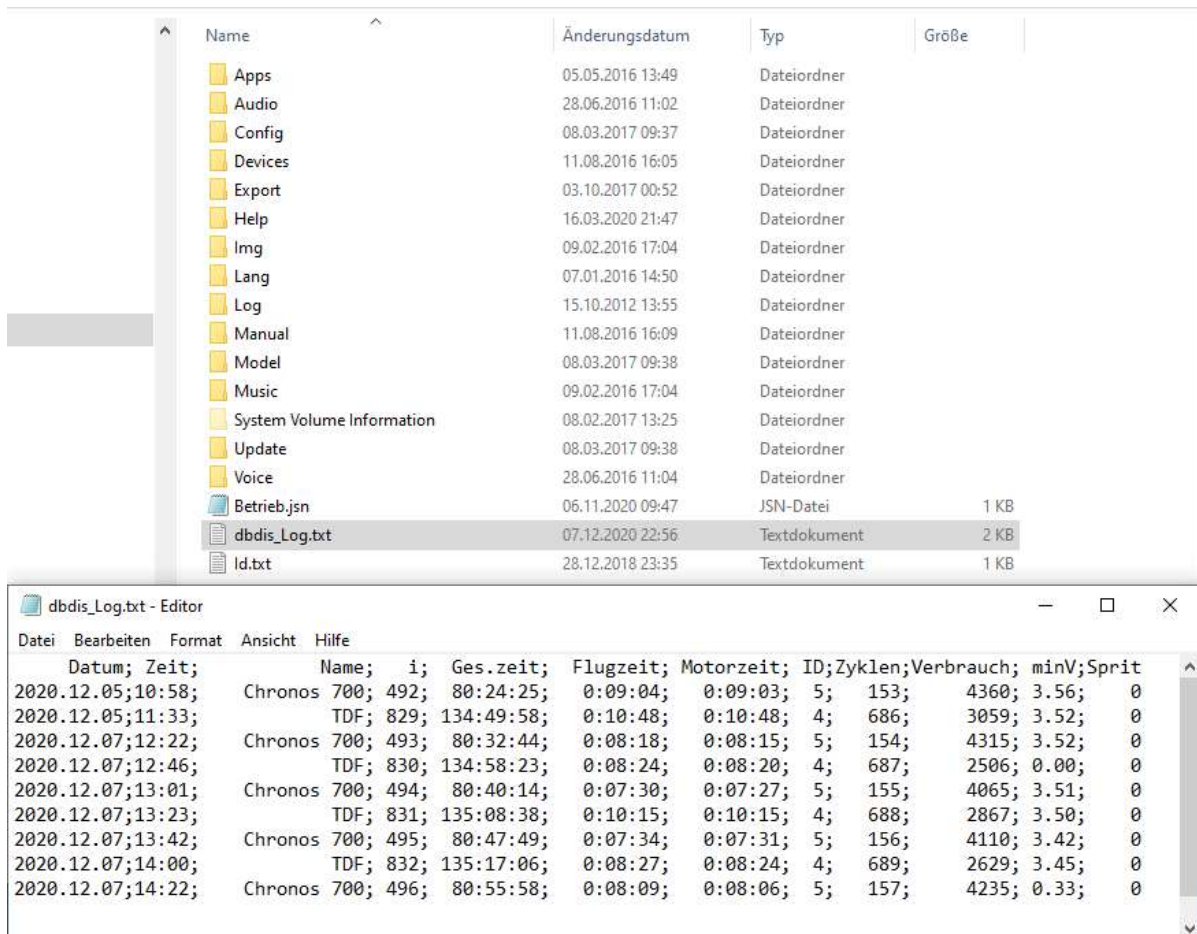
Tx		Motor aus	12:22:55		64%
ID	Name	S	mAh	C	
Zyklen		Verbrauch zuletzt		Gesamtverbrauch	
4	Quantum	12	5000	40	
682 Zyklen		992 mAh		35 Ah	
5	Quantum	12	5000	65	
218 Zyklen		80 mAh		123 Ah	
9	Wellpower	3	2200	45	
0 Zyklen		500 mAh		0 Ah	
1	2	3	BAT		

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. 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.jsn	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 aller Werte übersichtlich in Excel importiert.

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

dbdis_Log_de.xlsm - Excel Dieter Birkbauer

Start Einfügen Seitenlayout Formeln Daten Überprüfen Ansicht Hilfe Was möchten Sie tun?

Einfügen Zwischenablage

Schriftart Ausrichtung Zahl Formatvorlagen Zellen Bearbeiten Makros

A53

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Datum	Zeit	Name	Flüge	Ges.zeit	Flugzeit	Motorzeit	ID	Zyklen	Verbrauch	minV	Sprit		
37	2020.11.22	15:11	TDF	826	134:18:11	00:08:52	00:08:52	4	683	2772	3,15	0		
38	2020.11.22	15:37	Chronos 700	488	79:53:57	00:09:08	00:09:05	5	149	5000	2,52	0		
39	2020.11.24	15:43	Chronos 700	489	79:55:07	00:09:09	00:09:09	5	150	4400	2,56	0		
40	2020.11.24	16:01	TDF	827	134:28:59	00:10:48	00:10:47	4	684	2926	3,50	0		
41	2020.11.24	16:21	Chronos 700	490	80:05:11	00:10:04	00:10:04	5	151	4790	2,02	0		
42	2020.11.30	15:32	Chronos 700	491	80:15:21	00:10:10	00:10:09	5	152	4835	3,55	0		
43	2020.11.30	15:57	TDF	828	134:39:09	00:10:09	00:10:08	4	685	2888	3,40	0		
44	2020.12.05	10:58	Chronos 700	492	80:24:25	00:09:04	00:09:03	5	153	4360	3,56	0		
45	2020.12.05	11:33	TDF	829	134:49:58	00:10:48	00:10:48	4	686	3059	3,52	0		
46	2020.12.07	12:22	Chronos 700	493	80:32:44	00:08:18	00:08:15	5	154	4315	3,52	0		
47	2020.12.07	12:46	TDF	830	134:58:23	00:08:24	00:08:20	4	687	2506	0,00	0		
48	2020.12.07	13:01	Chronos 700	494	80:40:14	00:07:30	00:07:27	5	155	4065	3,51	0		
49	2020.12.07	13:23	TDF	831	135:08:38	00:10:15	00:10:15	4	688	2867	3,50	0		
50	2020.12.07	13:42	Chronos 700	495	80:47:49	00:07:34	00:07:31	5	156	4110	3,42	0		
51	2020.12.07	14:00	TDF	832	135:17:06	00:08:27	00:08:24	4	689	2629	3,45	0		
52	2020.12.07	14:22	Chronos 700	496	80:55:58	00:08:09	00:08:06	5	157	4235	0,33	0		
53														
54														

6. 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>

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