

Auswertung JK-BMS (und andere) und Kommunikation zu Victron

Ich habe mir letztes Jahr eine 24,8 kWp Anlage mit 5 Packs a 16s (EVE LF280K) gebaut. Wir haben einen 24h Verbrauch zwischen 40 – 45 kWh (4 Wohnungen). Jedes Pack hat ein JK-BMS mit 2A Balancer und 200A Dauerbelastbarkeit (aus heutiger Sicht rausgeschmissenes Geld. Ich benötige niemals 1000A für 3 MP2 5000. Selbst wenn 2 Packs gleichzeitig ausfallen, sind 600A nicht benötigt).

Bei der Integration der BMS ins Venus-OS habe ich mit dem dbus-Treiber von Louisvdw angefangen. Was anfangs sehr gut aussah. Es wurde alles schön in der VRM_Konsole angezeigt. Jede einzelne Zelle, die Minimum Zelle, die Maximum Zelle usw.. Dann habe ich noch die Zusammenfassung der Packs von Dr-Gigavolt [hier](#) gefunden und dachte: Klasse damit ist das Ziel erreicht. Leider war dem nicht so. Ich habe festgestellt, Wenn die Verbindung zum Venus-OS gekappt wird, Venus-OS einfach die letzten Werte einfriert. Sprich, alles ist gut. In meinen Augen ein NO-GO.

Dann habe ich mich mit dem Projekt von [Scotty89](#) beschäftigt, in der Hoffnung das der Online-Status und der für das Laden und Entladen aktuell sind beschäftigt (Funktioniert zu 100% so weit). Für jedes Pack habe ich ein ESP32 im Einsatz. Der ESP liefert die Spannungen der einzelnen Zellen, das Delta, den Strom, den Balancestrom, ob das Laden und Entladen aktiv sind und ob das BMS online ist. Leider nicht die Minimum-Spannung mit der zugehörigen Zelle und die Maximum-Spannung mit der zugehörigen Zelle (die Berechnung in ioBroker ist sehr ressourcenhungrig. Besonders bei einem Sendeinterfall von 1 Sek. Ich habe das Sendeinterfall auf 3 Sek angehoben, was mir nicht wirklich gefällt). Aus den gelieferten Daten habe ich für jedes Pack ein Dashboard in Grafana erstellt. Das Dashboard liefert mir die entsprechenden Informationen aufgrund der Datenpunkte in ioBroker.

Die folgenden Informationen erhalte ich:

Pack ist online oder offline, das Laden ist on oder off, das Entladen ist on oder off, die Minimum-Spannung der Zellen vom Pack, die Zelle welche die Minimum-Spannung hat, die Maximum-Spannung der Zelle vom Pack, die Zelle welche die Maximum-Spannung hat, das Delta der Zellen, die Pack-Spannung, den Strom der fließt, die Leistung in Watt, den SOC vom Pack (was auch immer jikong meint), den Balance-Strom und ein Balkendiagramm mit jeweils einem Balken pro Zelle der die entsprechende Spannung wiedergibt.

Nachdem ich mich von der Lösung mit dem Treiber von Louis abgewandt habe, habe ich mich gefragt, wie bekomme ich die Daten ins Venus-OS. Die Farge die zuerst gestellt werden muss ist: Was kann die Kommunikation / Datenvorgabe leisten?

Alle Packs hängen an einem Busbar. Venus-OS ist es nicht möglich auf jedes BMS individuell zu reagieren. Venus-OS kann nur den Ladestrom und den Entladestrom auf dem Bus begrenzen. Mit anderen Worten kann Venus-OS (oder jedes andere System) nur den Ladestrom oder Entladestrom für alle Packs anpassen. Sprich wir haben nur 2 Parameter.

Ich berechne in ioBroker anhand der Daten für jedes Pack den aktuellen Ladestrom und Entladestrom. Dies mache ich unter den Bedingungen wie in Louis-Treiber.

SOC (State Of Charge) from the BMS

- CCCM_SOC_ENABLE = True/False
- DCCM_SOC_ENABLE = True/False

CCCM limits the charge/discharge current depending on the SOC

- between 99% - 100% => 5A charge
- between 95% - 98% => 1/4 Max charge
- between 91% - 95% => 1/2 Max charge
- 30% - 91% => Max charge and Max discharge
- between 20% - 30% => 1/2 Max discharge
- between 10% - 20% => 1/4 Max discharge
- below <= 10% => 5A

Cell Voltage

- CCCM_CV_ENABLE = True/False
- DCCM_CV_ENABLE = True/False

CCCM limits the charge/discharge current depending on the highest/lowest cell voltages

- between 3.50V - 3.55V => 2A charge
- between 3.45V - 3.50V => 30A charge
- between 3.30V - 3.45V => 60A
- 3.30V - 3.10V => Max charge and Max discharge (60A)
- between 2.90V - 3.10V => 30A discharge
- between 2.8V - 2.9V => 5A discharge
- below <= 2.70V => 0A discharge

Temprature

- CCCM_T_ENABLE = True/False
- DCCM_T_ENABLE = True/False

CCCM limits the charge/discharge current depending on the highest/lowest temperature sensor values Charging will be 0A if below 0°C or above 55°C
Discharging will be 0A if below -20°C or above 55°C

CCCM werte ich anhand von der Maximum-Spannung einer Zelle, SOC (auch wenn dieser vom JK-BMS nicht gut ist) und der Temperatursensoren aus (ich könnte auch noch den inneren Widerstand hinzufügen). Ich schreibe das niedrigste Ergebnis der Auswertung in den Datenpunkt CCCM_Aktuell wenn das BMS „online“ ist und das Laden aktiv ist, ansonsten „0“.

DCCM werte ich anhand von der Minimum-Spannung einer Zelle, SOC (auch wenn dieser vom JK-BMS nicht gut ist) und der Temperatursensoren aus (ich könnte auch noch den inneren Widerstand hinzufügen). Ich schreibe das niedrigste Ergebnis der Auswertung in den Datenpunkt DCCM_Aktuell wenn das BMS „online“ ist und das Entladen aktiv ist, ansonsten „0“.

Final schreibe ich die Summe aller CCCM_Aktuell in einen Datenpunkt: Packs_CCCM_Aktuell und die Summe aller DCCM_Aktuell in einen Datenpunkt Packs_DCCM_Aktuell. Das sind die Werte die ich ins Venus-OS übertrage.

Meine Datenstruktur in ioBroker.

ID	Name	Typ	Rolle	Raum	Funktion	Wert	Einstellun...
0_userdata	Stammordner für Benutzer...	meta					
0							
PV	PV	folder					
Pack_1	Pack_1	folder					
Pack_2	Pack_2	folder					
Pack_3	Pack_3	folder					
Pack_4	Pack_4	folder					
Pack_5	Pack_5	folder					
Pack_Alle	Pack_Alle	folder					

ID	Name	Typ	Rolle	Raum	Funktion	Wert	Einstellun...
PV	PV	folder					
Pack_1	Pack_1	folder					
P1_CCCM_CV	P1_CCCM_CV	state	state			60	
P1_CCCM_SOC	P1_CCCM_SOC	state	state			60	
P1_CCCM_T	P1_CCCM_T	state	state			60	
P1_Charge_Aktuell	P1_Charge_Aktuell	state	state			60	
P1_DCCM_CV	P1_DCCM_CV	state	state			70	
P1_DCCM_SOC	P1_DCCM_SOC	state	state			70	
P1_DCCM_T	P1_DCCM_T	state	state			70	
P1_Discharge_Aktuell	P1_Discharge_Aktuell	state	state			70	
P1_Maximum	P1_Maximum	state	state			3,319	
P1_Minimum	P1_Minimum	state	state			3,316	
P1_Name_Max	P1_Name_Max	state	state			P1_Z02	
P1_Name_Min	P1_Name_Min	state	state			P1_Z09	
P1_Status_Charge	P1_Status_Charge	state	state			1	
P1_Status_Discharge	P1_Status_Discharge	state	state			1	

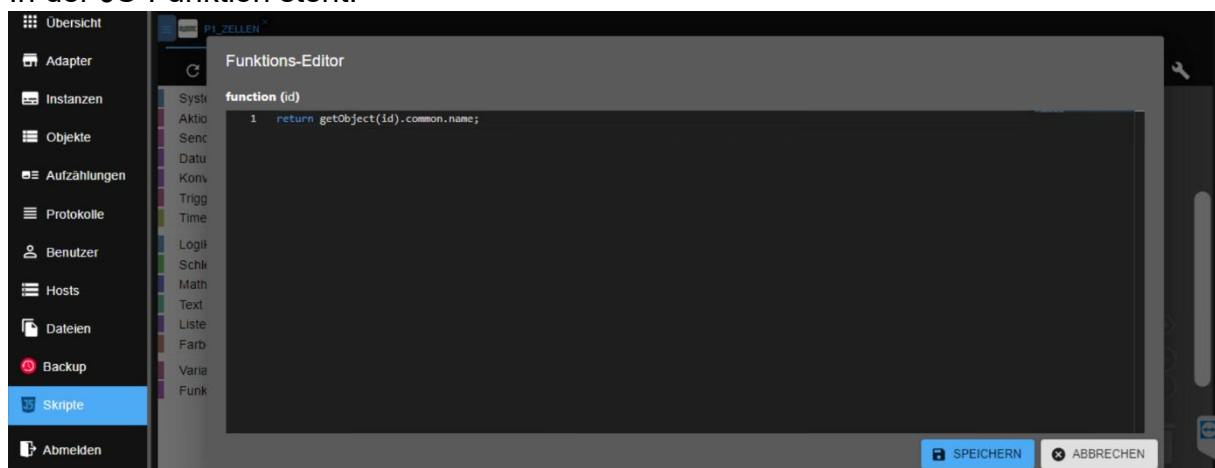
ID	Name	Typ	Rolle	Raum	Funktion	Wert	Einstellun...
Pack_Alle	Pack_Alle	folder					
Grundwert_Charge	Grundwert_Charge	state	value			60 A	
Grundwert_Discharge	Grundwert_Discharge	state	value			70 A	
Pack_Alle_CCCM_Aktuell	Pack_Alle_CCCM_Aktuell	state	state			300	
Pack_Alle_Charger	Pack_Alle_Charger	state	state			5	
Pack_Alle_DCCM_Aktuell	Pack_Alle_DCCM_Aktuell	state	state			350	
Pack_Alle_Delta	Pack_Alle_Delta	state	state			0,019	
Pack_Alle_Discharge	Pack_Alle_Discharge	state	state			5	
Pack_Alle_Max	Pack_Alle_Max	state	state			3,398	
Pack_Alle_Min	Pack_Alle_Min	state	state			3,379	
Pack_Alle_Name_Max	Pack_Alle_Name_Max	state	state			P1_Z09	
Pack_Alle_Name_Min	Pack_Alle_Name_Min	state	state			P4_Z01	
Pack_Alle_Status	Pack_Alle_Status	state	state			5	

Grundwert_Charge und Grundwert_Discharge sind die Grundwerte für jedes einzelne Pack wenn keine Beschränkung vorliegt. In meinem Fall mit 5 Packs, max. 300A Ladestrom und max. 350A Entladestrom.

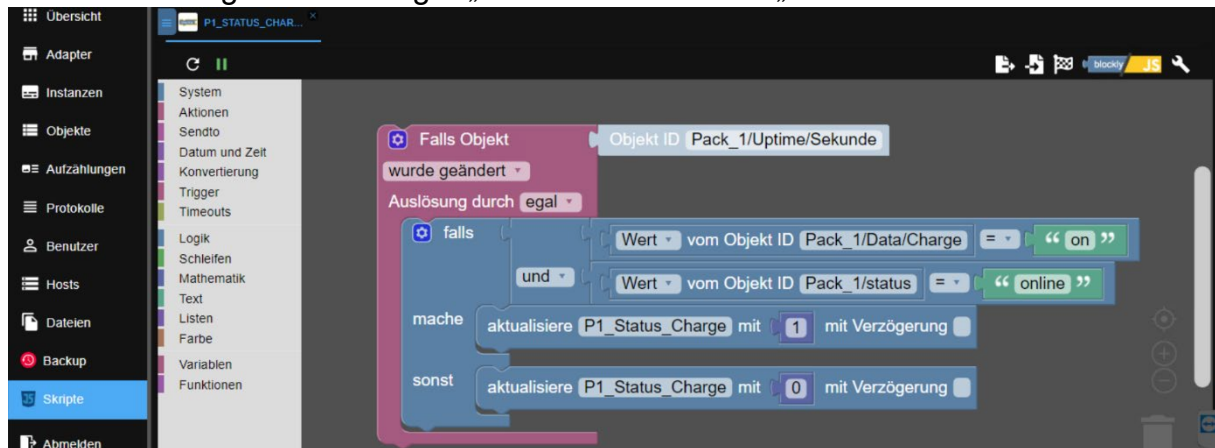
Die Bestimmung der Zelle mit der min. Spannung und welche es ist, sowie die Bestimmung der Zelle mit der max. Spannung und welche es.



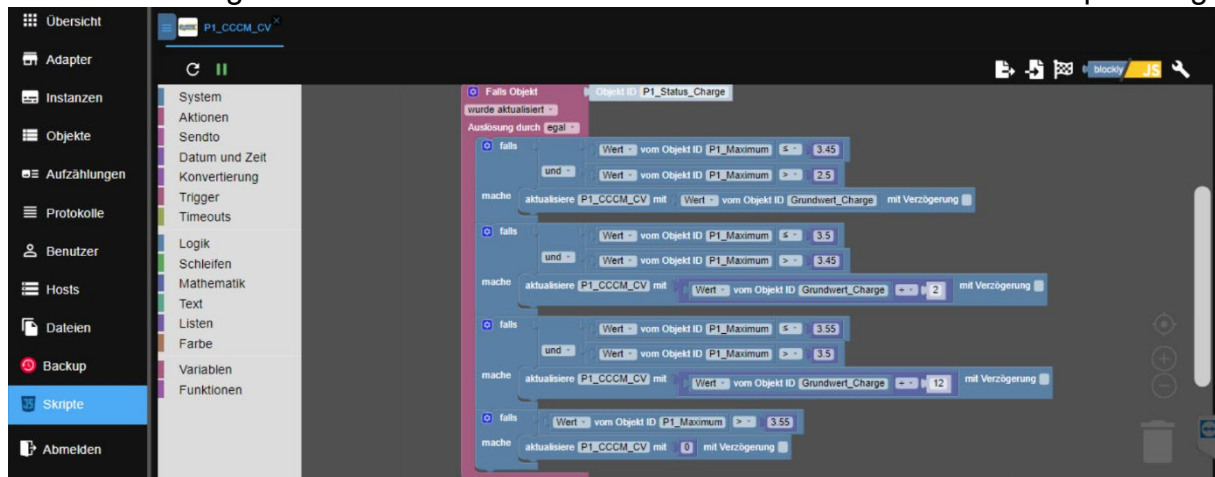
In der JS-Funktion steht:



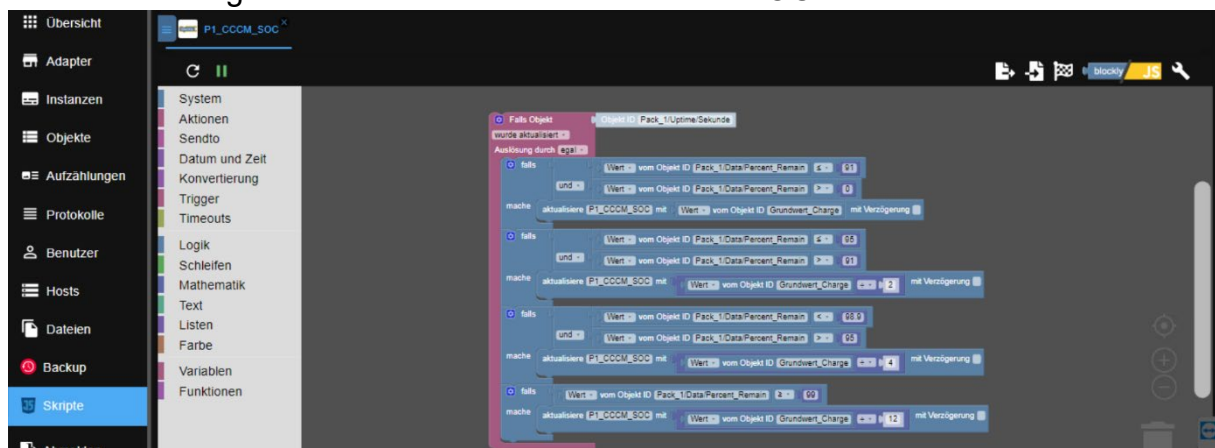
Die Feststellung ist der Charger „on“ und ist das BMS „online“.



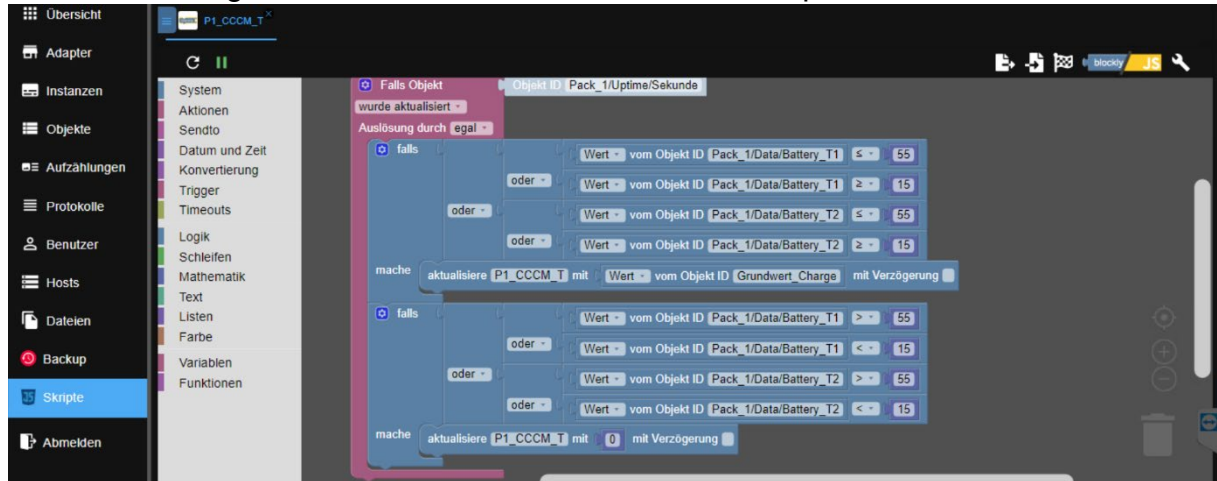
Die Bestimmung des max. Ladestroms anhand der Zelle mit der höchsten Spannung.



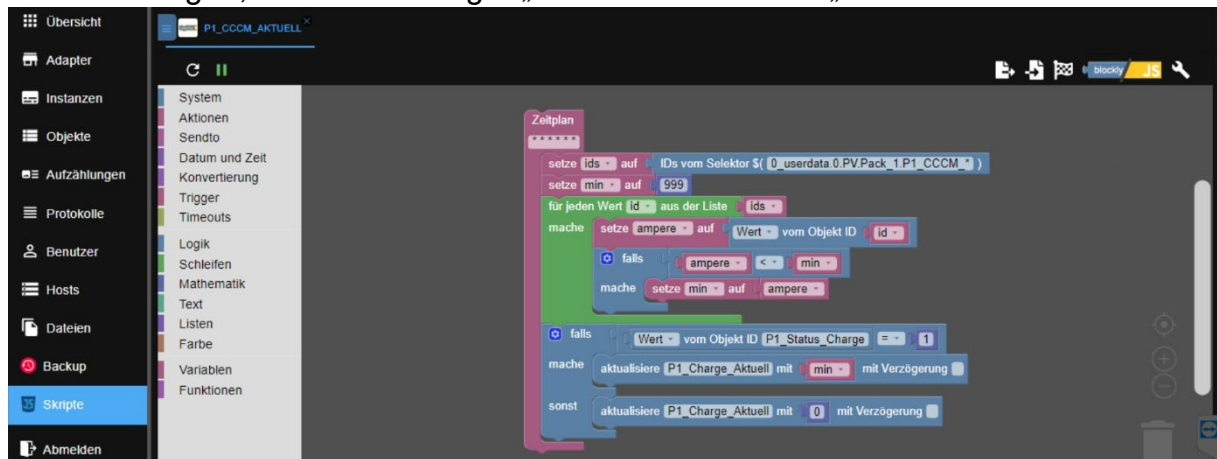
Die Bestimmung des max. Ladestroms anhand des SOC.



Die Bestimmung des max. Ladestroms anhand der Temperaturen.



Setzen des aktuellen max. Ladestroms für das Pack_1 mit dem niedrigsten Ergebnis der 3 Prüfungen, wenn der Charger „on“ ist und das BMS „online“ ist.

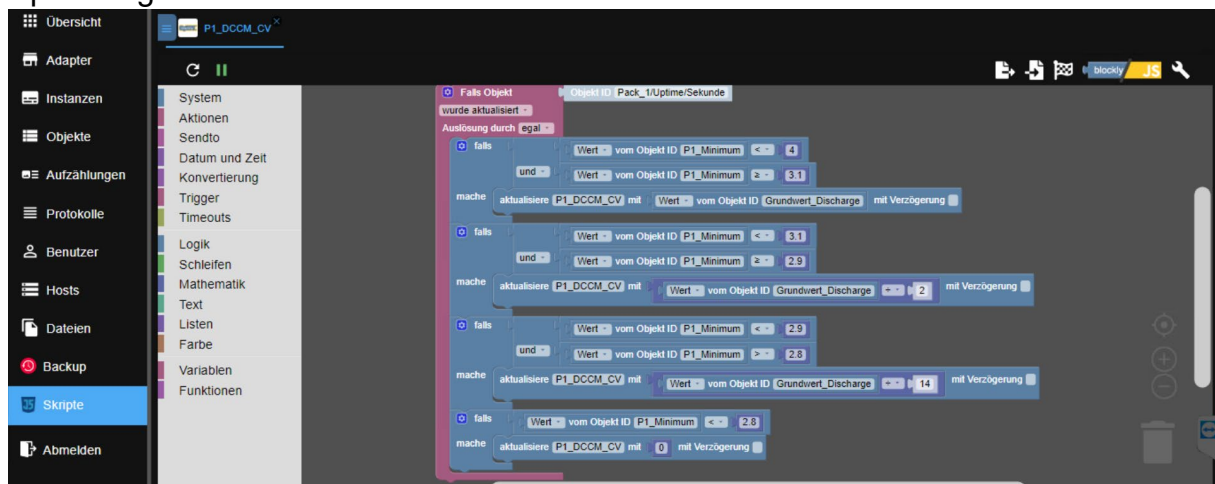


Und nochmals ähnlich für das Entladen.

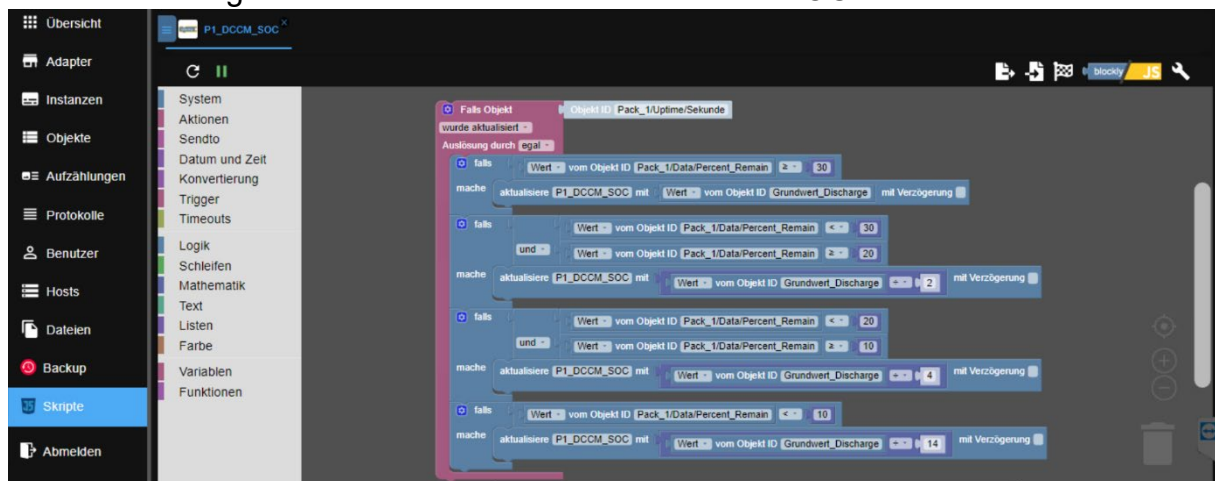
Die Feststellung ist der Discharger „on“ und ist das BMS „online“.



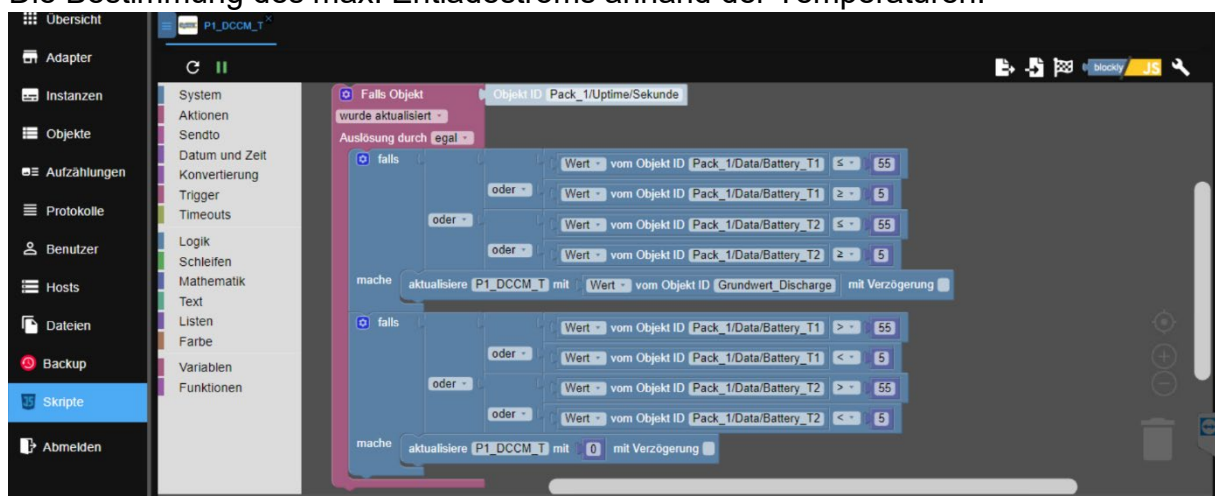
Die Bestimmung des max. Entladestroms anhand der Zelle mit der niedrigsten Spannung.



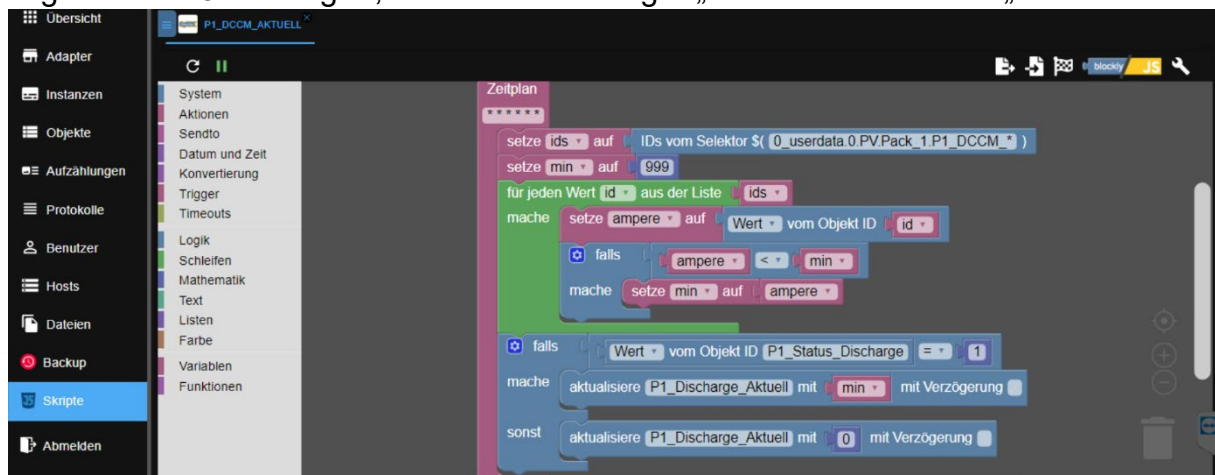
Die Bestimmung des max. Entladestroms anhand des SOC.



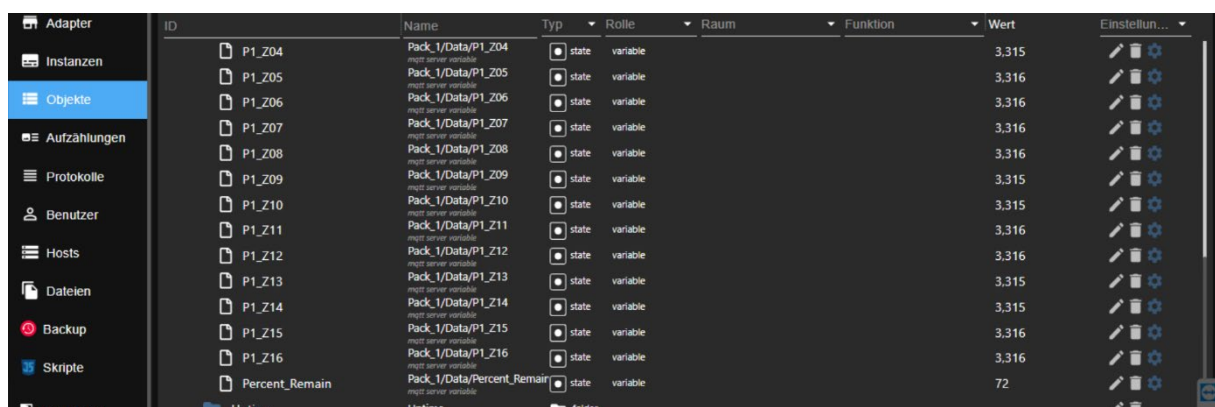
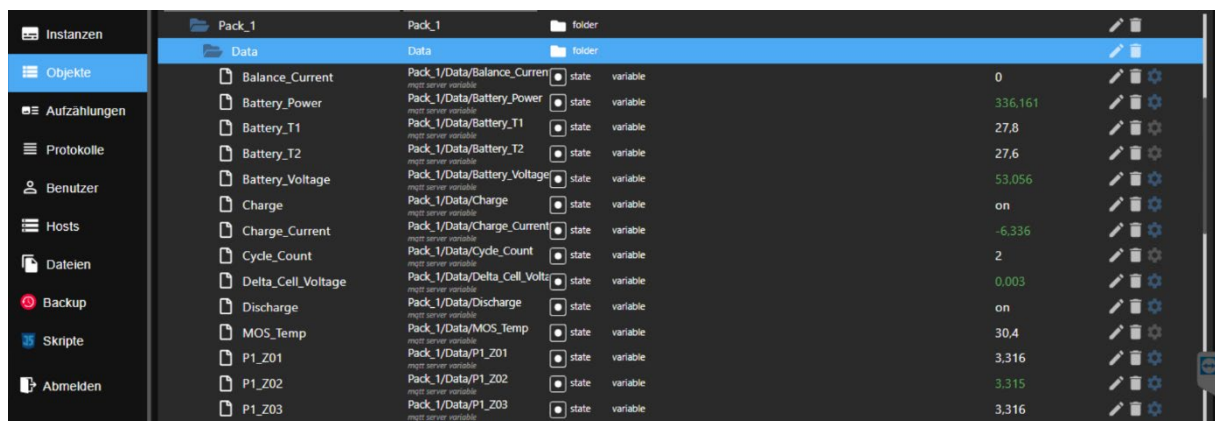
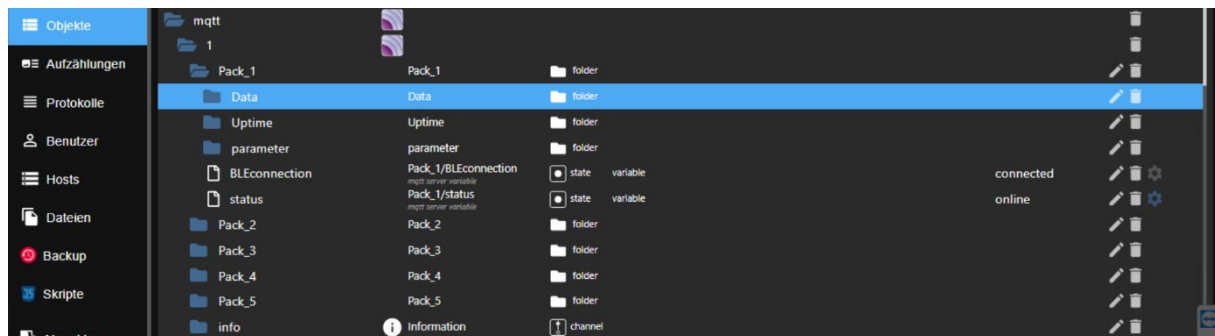
Die Bestimmung des max. Entladestroms anhand der Temperaturen.

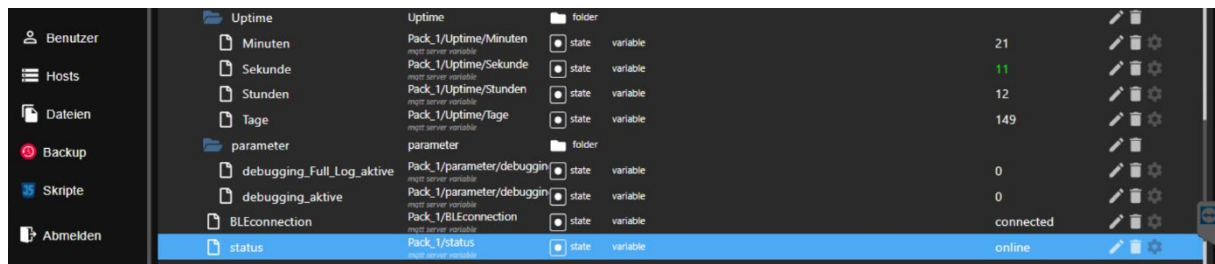


Setzen des aktuellen max. Entladestroms für das Pack_1 mit dem niedrigsten Ergebnis der 3 Prüfungen, wenn der Discharger „on“ ist und das BMS „online“ ist.

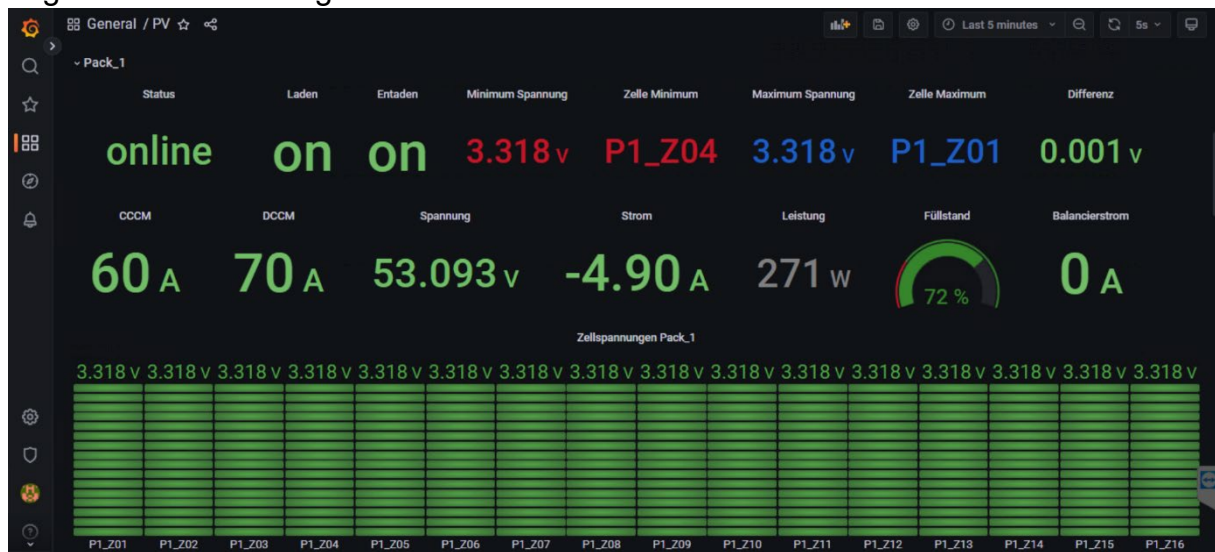


Anhand der Daten die ich vom ESP32 bekomme





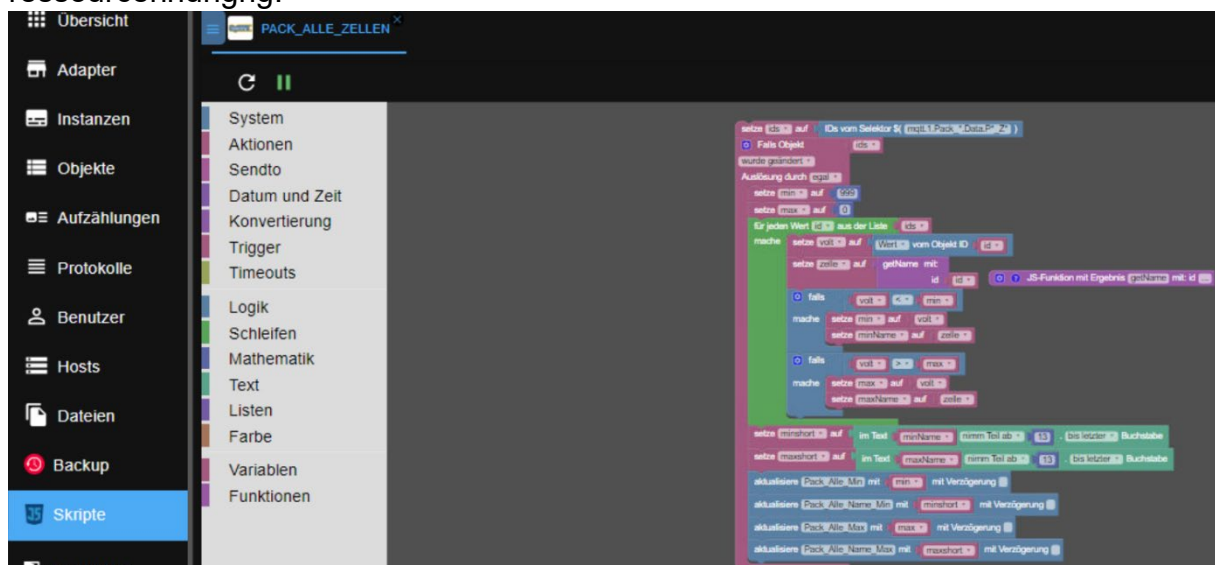
und der anschließenden Berechnung der vorherigen Datenpunkte habe ich mir folgendes in Grafana gebastelt.



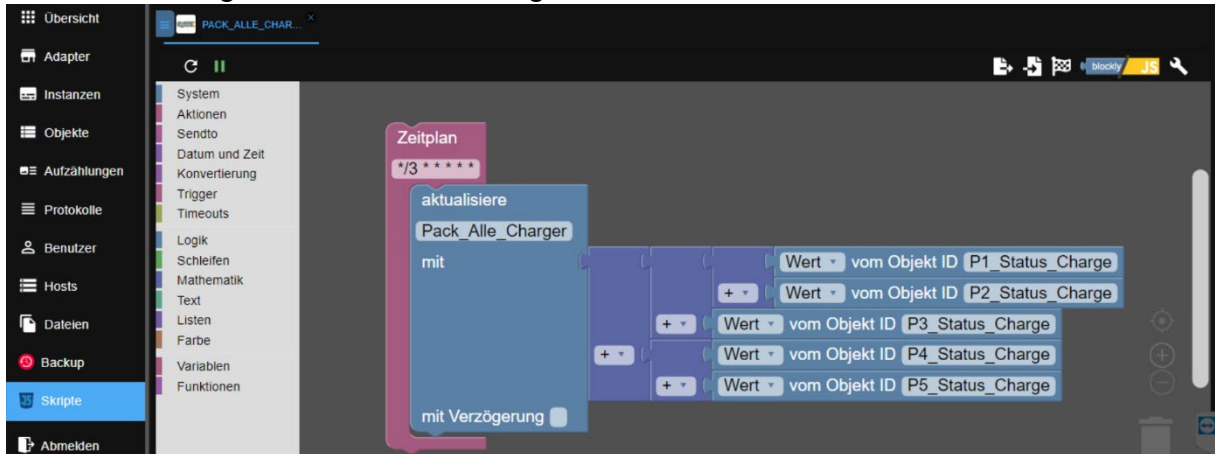
Das Ganze habe ich dann für die restlichen 4 Packs wiederholt (hält schon ein paar Stunden auf).

Dann habe ich mit der Zusammenfassung aller Packs begonnen.

Die Bestimmung der Zelle mit der min. Spannung und welche es ist, sowie die Bestimmung der Zelle mit der max. Spannung und welche es. Sehr ressourcenhungrig.



Die Feststellung wie viele der 5 Charger „on“ sind.



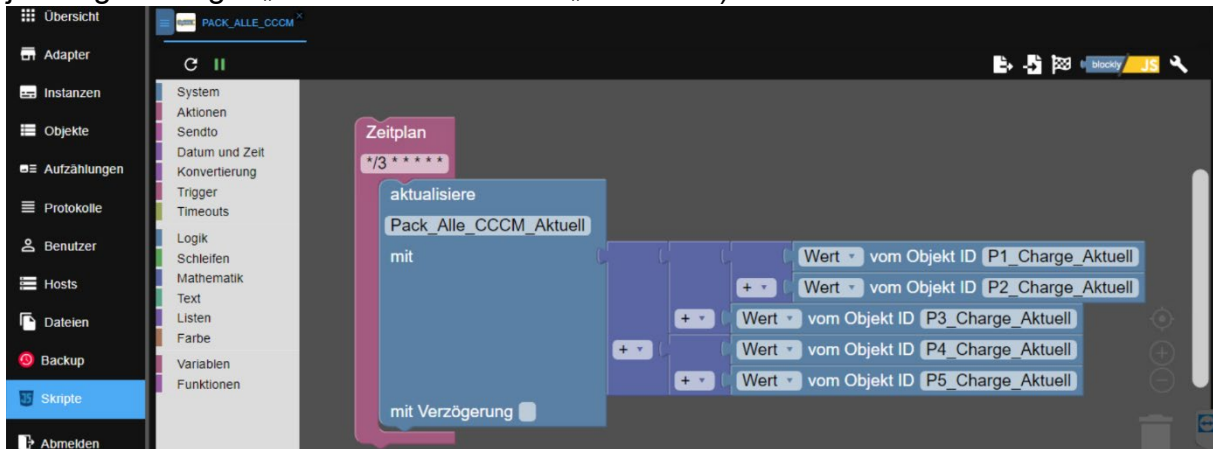
Die Feststellung wie viele der 5 Discharger „on“ sind.



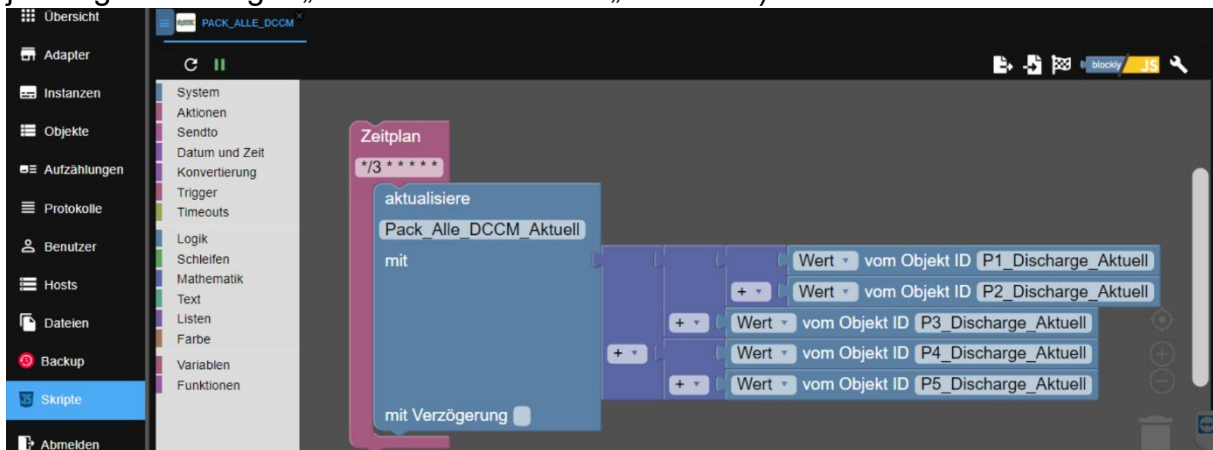
Die Feststellung wie viele der 5 BMS „online“ sind.



Die Berechnung des max. Ladestroms anhand der Werte von den 5 Packs (zur Erinnerung, die Werte der einzelnen Packs werden nur geschrieben wenn der jeweilige Charger „on“ ist und das BMS „online“ ist).

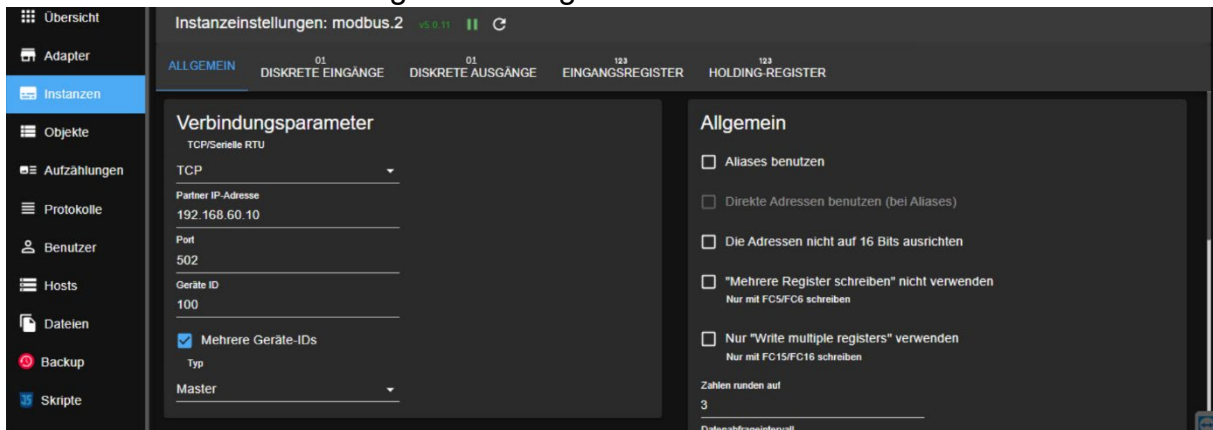


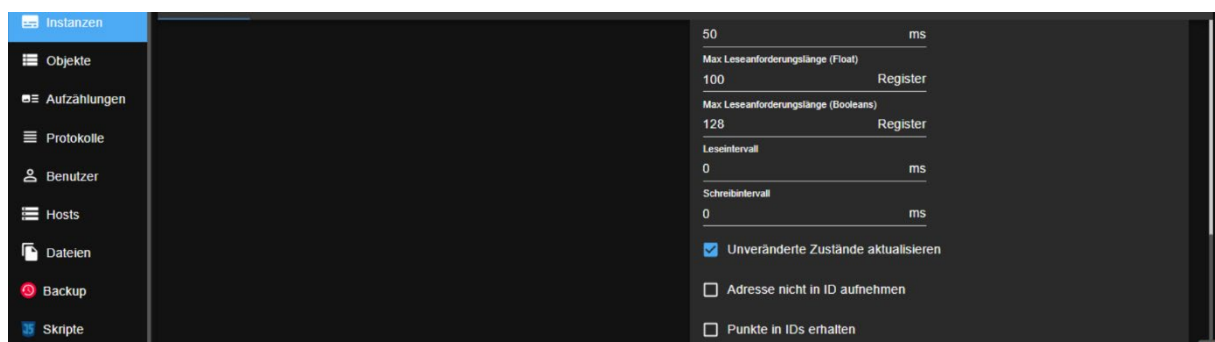
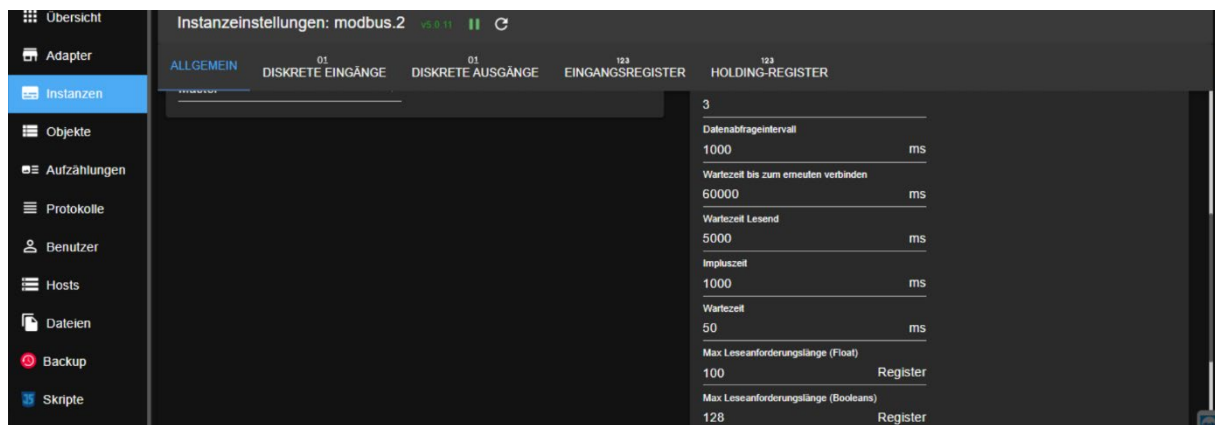
Die Berechnung des max. Entladestroms anhand der Werte von den 5 Packs (zur Erinnerung, die Werte der einzelnen Packs werden nur geschrieben wenn der jeweilige Discharger „on“ ist und das BMS „online“ ist).



Die Werte vom Victron SmartShunt und den aktuellen Lade.- und Entladestrom hole ich mir via Modbus vom Cerbo.

Dies sind meine Verbindungseinstellungen:





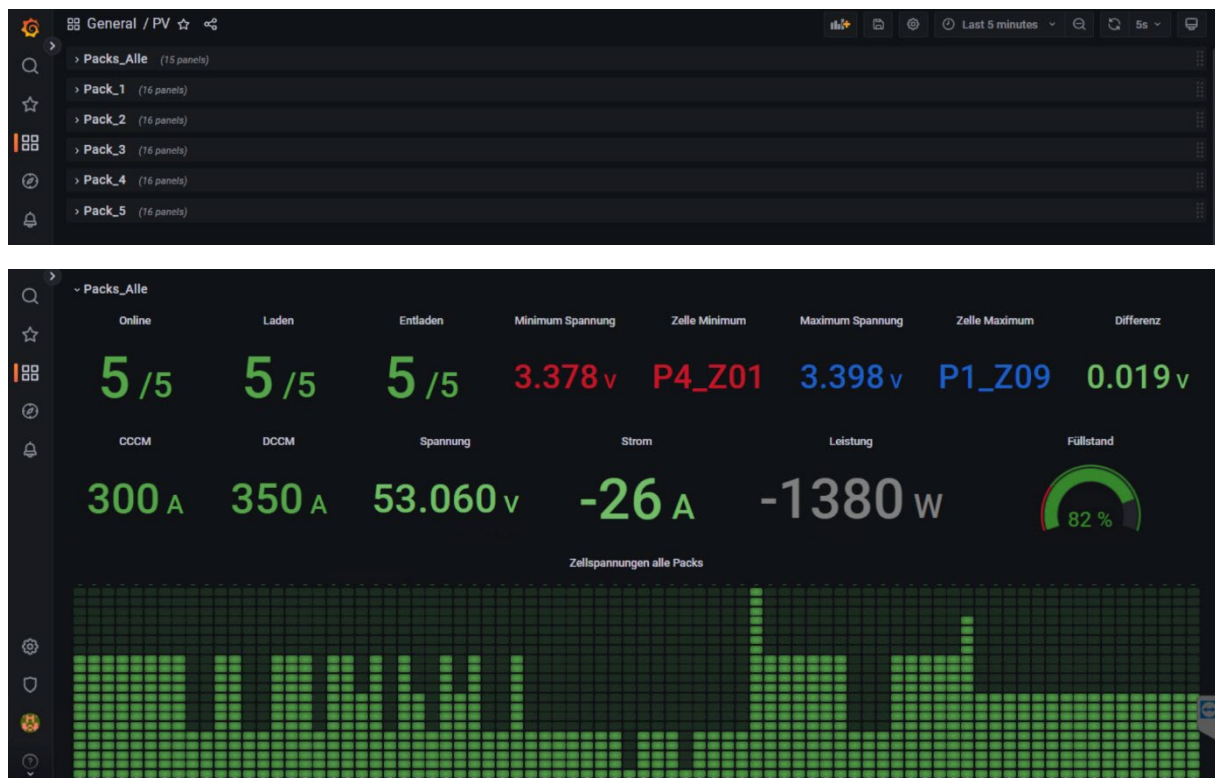
Mit diesen Einstellungen hole ich mir diese Eingangsregister:

Adresse	Slave-ID ↑	Name	Beschreibung	Einheit	Typ	Länge	Rolle	Raum	☐ CW	
842	100	Power	Power	W	Signed 16 bit (Big Endian)	1	value		☐	
843	100	SOC	SOC	%	Unsigned 16 bit (Big Endian)	1	value		☐	
2704	100	max Discharge	max Discharge	A	Unsigned 16 bit (Big Endian)	1	value		☐	
2705	100	max Charge	max Charge	A	Unsigned 16 bit (Big Endian)	1	value		☐	
259	226	Battery voltage	Battery voltage	V	Unsigned 16 bit (Big Endian)	1	value		☐	
261	226	Current	Current	A	Signed 16 bit (Big Endian)	1	value		☐	

Daraus erhalte ich die folgenden Datenpunkte:

Objekt	Name	Beschreibung	state	value	Einheit	Wert	Icon
modbus							
0							
1							
2							
holdingRegisters		Holding registers					
info		info					
inputRegisters		Input registers					
100							
2704_max_Discharge	max Discharge		state	value	350 A		
2705_max_Charge	max Charge		state	value	300 A		
842_Power	Power		state	value	-1258 W		
843_SOC	SOC		state	value	82 %		
226							
259_Battery_voltage	Battery voltage		state	value	53.07 V		
261_Current	Current		state	value	-23.8 A		

Mit den Daten für alle 5 Packs und den Daten vom Cerbo, habe ich mein Grafana Dashboard erweitert.



Das wovor ich die größte Angst hatte (das integrieren der Werte ins Venus-OS) hat sich als die einfachste Übung herausgestellt.

In der Modbus Instanz gibt es auch Holdingregister.

Instanzeinstellungen: modbus.2 v5.0.11

ALLGEMEIN DISKRETE EINGÄNGE DISKRETE AUSGÄNGE EINGANGSREGISTER **HOLDING-REGISTER**

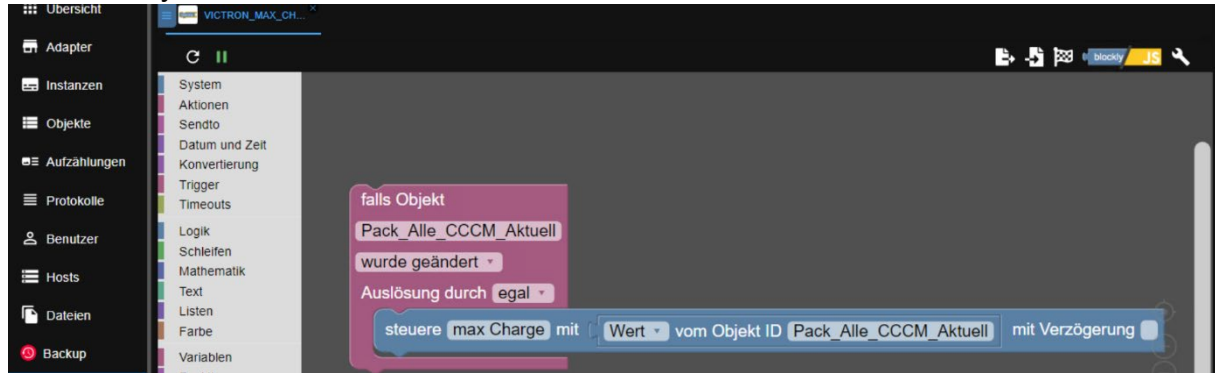
Adresse	Slave-ID	Name	Beschreibung	Einheit	Typ	Länge	Rolle	Raum	Abfrage	CW	
2704	100	max Discharge	max Discharge	A	Unsigned 16 bit (Big Endian)	1	value		☐	☐	☐
2705	100	max Charge	max Charge	A	Unsigned 16 bit (Big Endian)	1	value		☐	☐	☐

Diese Register werden auf den Cerbo geschrieben und dann sofort geleert.

Gesetzt werden diese Datenpunkte:

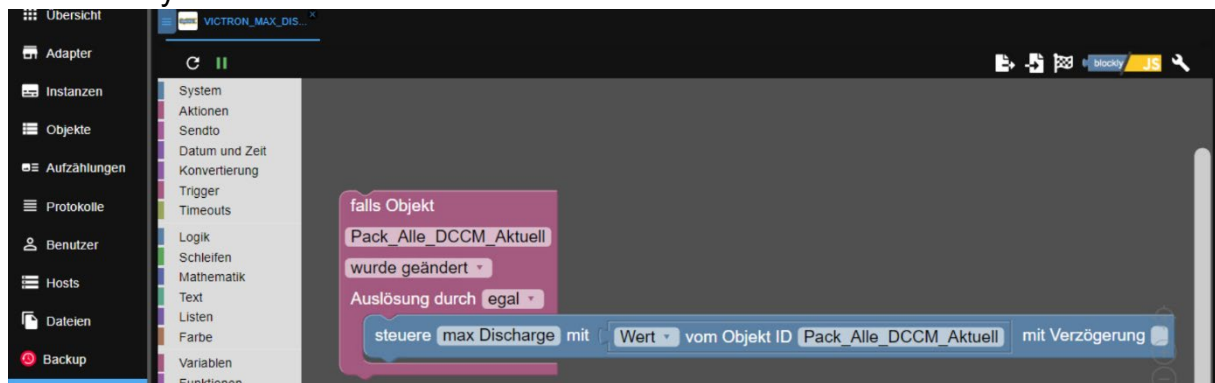
ID	Name	Typ	Rolle	Raum	Funktion	Wert	Einstellun...
modbus							
0		meta					
1		meta					
2		meta					
holdingRegisters	Holding registers	channel					
100							
2704_max_Discharge	max Discharge	state	value			(null) A	
2705_max_Charge	max Charge	state	value			(null) A	
info	info	channel					
inputRegisters	Input registers	channel					

Das Blockly für das Laden:



Sobald Pack_Alle_CCCM_Aktuell (die Summe aus den Packs) geändert wurde, wird max_Charge (das Holdinregister) auf den Wert von Pack_Alle_CCCM_Aktuell gesetzt. Im nächsten Zyklus wird es ins Cerbo geschrieben und das Holdingregister geleert (Achtung! Auf den Trigger achten „wurde geändert“, wenn dieser auf „wurde aktualisiert“ steht, wird das Holdingregister permanent beschrieben).

Das Blockly für das Entladen:



Sobald Pack_Alle_DCCM_Aktuell (die Summe aus den Packs) geändert wurde, wird max_Discharge (das Holdinregister) auf den Wert von Pack_Alle_DCCM_Aktuell gesetzt. Im nächsten Zyklus wird es ins Cerbo geschrieben und das Holdingregister geleert (Achtung! Auf den Trigger achten „wurde geändert“, wenn dieser auf „wurde aktualisiert“ steht, wird das Holdingregister permanent beschrieben).

Ich hoffe es hilft dem Einem oder Anderen, besonders mit mehreren Packs.

In meinen Augen ist die Lösung auch mit nur einem Pack besser als das Pack direkt mit dem Cerbo zu verbinden, da der Onlinestatus beachtet wird.

VG

Harald