

Werte aus Victron Cerbo GX in Homematic CCU übertragen

Diese Anleitung beschreibt wie man Daten aus Victron Cerbo GX (oder anderen Victron GX devices) in die Homematic CCU bekommt. Scripting oder Linux Kenntnis ist hierzu nicht nötig. Sobald man die Daten in der Homematic hat kann man z.B. die Werte visualisieren (ich nutze Mediola), oder Schaltvorgänge auslösen (z.B. Boiler mit Solarstrom betreiben, wenn die Batterien voll sind und man PV-Überschuss hat).

Für die Funktion sind 2 Programme nötig:

- „Node-Red“ sowie „Mqtt Explorer“. Letzterer dient nur dazu die entsprechenden Verzeichnisse der benötigten Daten im Cerbo GX herauszufinden, danach wird dieser nicht mehr benötigt.

Folgende Schritte sind nötig:

1. Auf der Cerbo GX muss man das Mqtt Protokoll aktivieren unter „Dienste“, mehr ist hier nicht nötig:



2. nun installiert man sich den „Mqtt Explorer“ (gibt es für viele Betriebssysteme) <http://mqtt-explorer.com/>. Bei IP-Adresse hinterlegt man die der Cerbo GX mit folgenden Einstellungen, danach empfängt man die Mqtt Daten der Cerbo:

Connections

- cerbo cx**
mqtt://192.168.0.242:1883/
- test.mosquitto.org**
mqtt://test.mosquitto.org:1883/

MQTT Connection

mqtt://192.168.0.242:1883/

Name: cerbo cx

Validate certificate: ☒

Encryption (tls): ☐

Protocol: mqtt:// Host: 192.168.0.242 Port: 1883

Username: _____ Password: _____

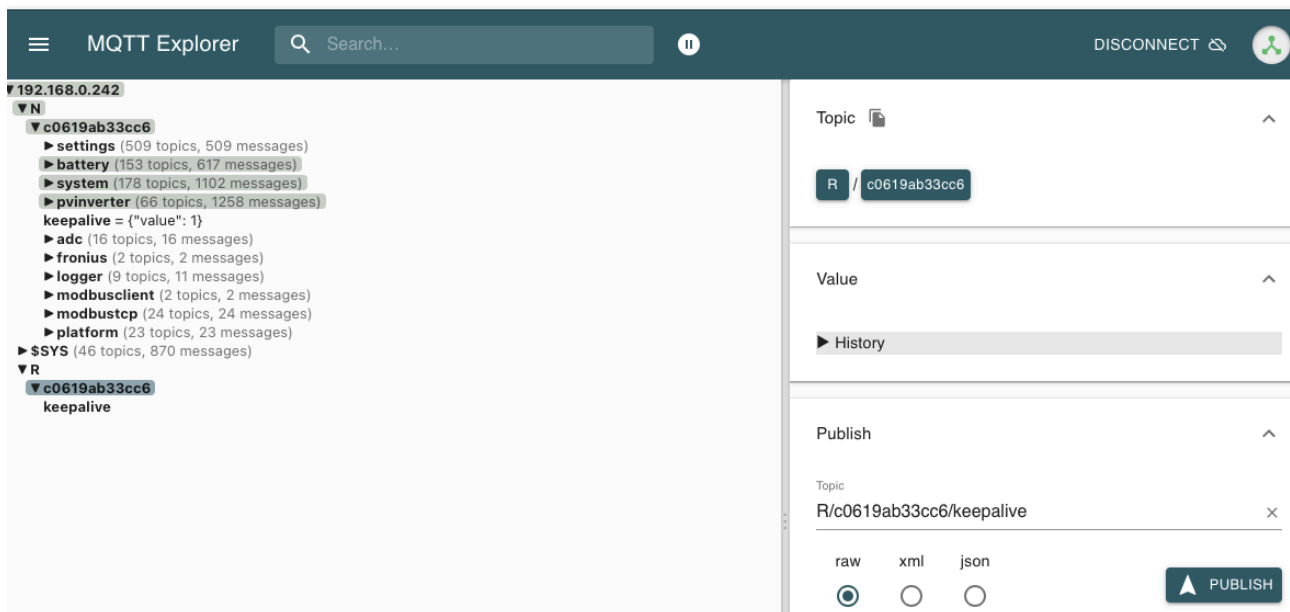
DELETE
 ADVANCED
 SAVE
 CONNECT

Nach „Connect“ erhält man die Mqtt Daten des Cerbo GX. Beispielsweise im Unterordner „R“ und der VRM Instanz-Nummer von Victron hat man das Verzeichnis an den die „keepalive“ Pakete hingeschickt werden können (siehe Punkt 4), damit der Cerbo nicht aufhört Daten zu Übertragen.

Kann man auch manuell testen indem man das keepalive Paket mit der Schaltfläche „publish“ verschickt, damit füllt sich z.B. der Explorer mit Informationen (siehe nachfolgender Screenshot)

In dem Verzeichnisbaum des Mqtt-Exploreres findet man alle Daten des Cerbo GX. Sucht man z.B. die Batteriespannung, kann man einfach unter „Search“ den Wert eingeben und dann danach suchen. Beispiel: Die aktuelle Spannung ist 53,44 Volt, man sucht einfach nach „53“.

Die Verzeichnis mit den gewünschten Daten kann man einfach mit Schaltfläche „Topic“ kopieren und in Node-Red dann einfügen (siehe später Abschnitt).



3. „Node-Red“, eine Zusatzsoftware, die einfach auf die CCU unter Systemsteuerung —> Zusatzsoftware installiert wird. Die Software kann man unter <https://github.com/rdmtdc/RedMatic#readme> downloaden. Nach Installation und reboot der CCU kann man die RedMatic unter <http://192.168.0.110/addons/red> erreichen (IP-Adresse ist die der CCU, Benutzername Kennwort entspricht dem der Homematic). Nach Aufruf sieht man vorinstallierte „Flows“, die kann man auch löschen. Man legt dann einen neuen Flow an:
4. Erster Flow ist eine Routine die den Cerbo GX ständig abfragt. Hintergrund ist das der Cerbo GX sonst die Datenübertragung einstellt und man keine Werte erhält. das geht ganz einfach:



wobei das Inject-Node wie folgt konfiguriert wird:

Inject Node bearbeiten

Löschen Abbrechen Fertig

Properties

Name Inject_alle_30s

msg. payload = {}

+ hinzufügen

☒ Einmal nach der Injektion 0.9 Sekunden, dann

Wiederholen Intervall

alle 30 Sekunden

Das andere, rosafarbene Node „Mqtt out“ wird wie folgt konfiguriert: Der Wert unter „Topic“ ist das Verzeichnis den man vorher im Mqtt-Explorer gesucht hat, zusätzlich „keepalive“ (das komplette Verzeichnis erscheint nicht immer im Explorer) angehängt. Es wird somit alle 30 Sekunden an den Cerbo eine Nachricht geschickt sodass dieser anfängt die Werte zurück zu senden.

Wobei bei „Server“ die IP-Adresse des Cerbo GX hinterlegt wird. Das muss man nur einmalig machen, bei allen weiteren Konfigurationen kann man dann den Server einfach wieder auswählen:

U

mqtt out Node bearbeiten

Löschen

Abbrechen

Fertig

⚙ Properties

⚙ 📄 🖨

🌐 Server

Cerbo_GX

✎

📄 Topic

R/c0619ab33cc6/keepalive

⚙ QoS

🔄 Retain

📄 Name

Leere Nachricht senden

Tipp: Behalten Sie das Topic "Artikel", "qos" oder "retain" bei, wenn Sie diese über die Eigenschaft "msg" festlegen

mqtt out Node bearbeiten > mqtt-broker Node bearbeiten

Löschen

Abbrechen

Aktualisieren

⚙ Properties

⚙ 📄

📄 Name

Cerbo_GX

Verbindung

Sicherheit

Nachrichten

🌐 Server

192.168.0.242

Port

1883

☐ Sichere Verbindung (SSL/TLS) aktivieren

📄 Client-ID

Leerer Wert für automatische Generierung

🕒 Keepalive-Zeit (en)

60

☒ Bereinigte Sitzung verwenden

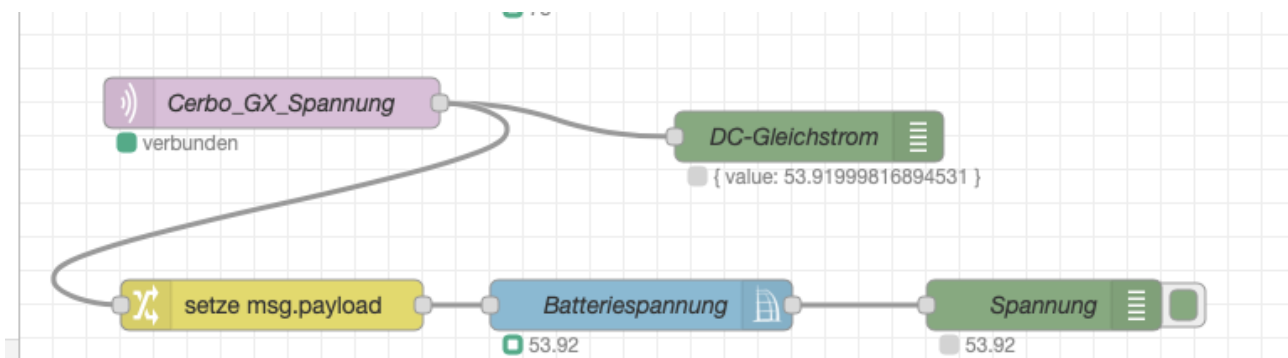
☐ Traditionelle MQTT 3.1-Unterstützung verwenden

Mit der Schaltfläche „deploy“ in Node-Red überträgt man immer die Konfiguration zur CCU und das Programm arbeitet.

5. Nun noch alle Systemvariablen in der CCU anlegen, die man Auswertung möchte. Als Beispiel Batteriespannung:

Startseite Status und Bedienung Programme und Verknüpfungen Einstellungen Geräte anlernen Hilfe							
Name	Beschreibung	Variablentyp	Werte	Maßeinheit	Kanal-zuordnung	Aktion	Verknüpfung
Filter		Filter		Filter	Filter		
Batteriespannung		Zahl	Minimalwert: 0 Maximalwert: 100			<button>Löschen</button> <input checked="" type="checkbox"/> sichtbar <button>Bearbeiten</button> <input checked="" type="checkbox"/> protokolliert <input type="checkbox"/> systemintern	<button>Programme</button>

Hier der zugehörige Flow in Node-Red, wobei die Grünen Debug-Nodes auch weggelassen werden können, das sind reine Informationsfelder die die Werte in Node-Red zusätzlich ausgeben:



Das rosafarbene „mqtt In“ Node wird wie folgt konfiguriert, unter Topic das Verzeichnis das man im Mqtt Explorer kopiert hat und den Wert der Spannung enthält:

mqtt in Node bearbeiten

Löschen

Abbrechen

Fertig

⚙️ Properties

⚙️ 📄 🖨️

🌐 Server

Cerbo_GX

✎

📋 Topic

N/c0619ab33cc6/system/0/Dc/Battery/Voltage

⚙️ QoS

2

➡️ Output

a parsed JSON object

🏷️ Name

Cerbo_GX_Spannung

Das Gelbe Change Node sieht wie folgt aus, hier wird der Wert gleich gerundet:

change Node bearbeiten

Löschen

Abbrechen

Fertig

⚙️ Properties

⚙️ 📄 🖨️

🏷️ Name

Name

📋 Regeln

⋮

Setze

▼ msg. payload

auf

▼

⌘: \$round(payload.value,2)

⋮

✕

Das blaue sysvar Node sieht wie folgt aus, die vorher angelegte Systemvariable in der CCU wird im dropdown Feld „Name“ ausgewählt:

sysvar Node bearbeiten

Löschen

Abbrechen

Fertig

Properties

CCU

CCU

Topic

ReGaHSS/\${Name}

Name

Batteriespannung

☒ Nur geänderte Werte ausgeben

☒ Beim Start aktuellen Wert ausgeben

Wird keine Variable ausgewählt kann der Variablen-Name über msg.topic übergeben werden.

Im Feld „CCU“ unter „bearbeiten“ hinterlegt man die IP-Adresse der CCU, die, meisten anderen Werte werden nicht geändert. Muss auch nur einmal je CCU gemacht werden. Alle anderen Werte werden Analog zur Batteriespannung konfiguriert:

sysvar Node bearbeiten > ccu-connection Node bearbeiten

Löschen

Abbrechen

Aktualisieren

⚙ Properties

⚙

CCU name

CCU

CCU address

localhost

Listen address

192.168.0.110

Init address

192.168.0.110

BINRPC
listening port

2047

XMLRPC
listening port

2048

Interfaces

☒ ReGaHSS

☐ BidCos-RF

☐ BidCos-Wired

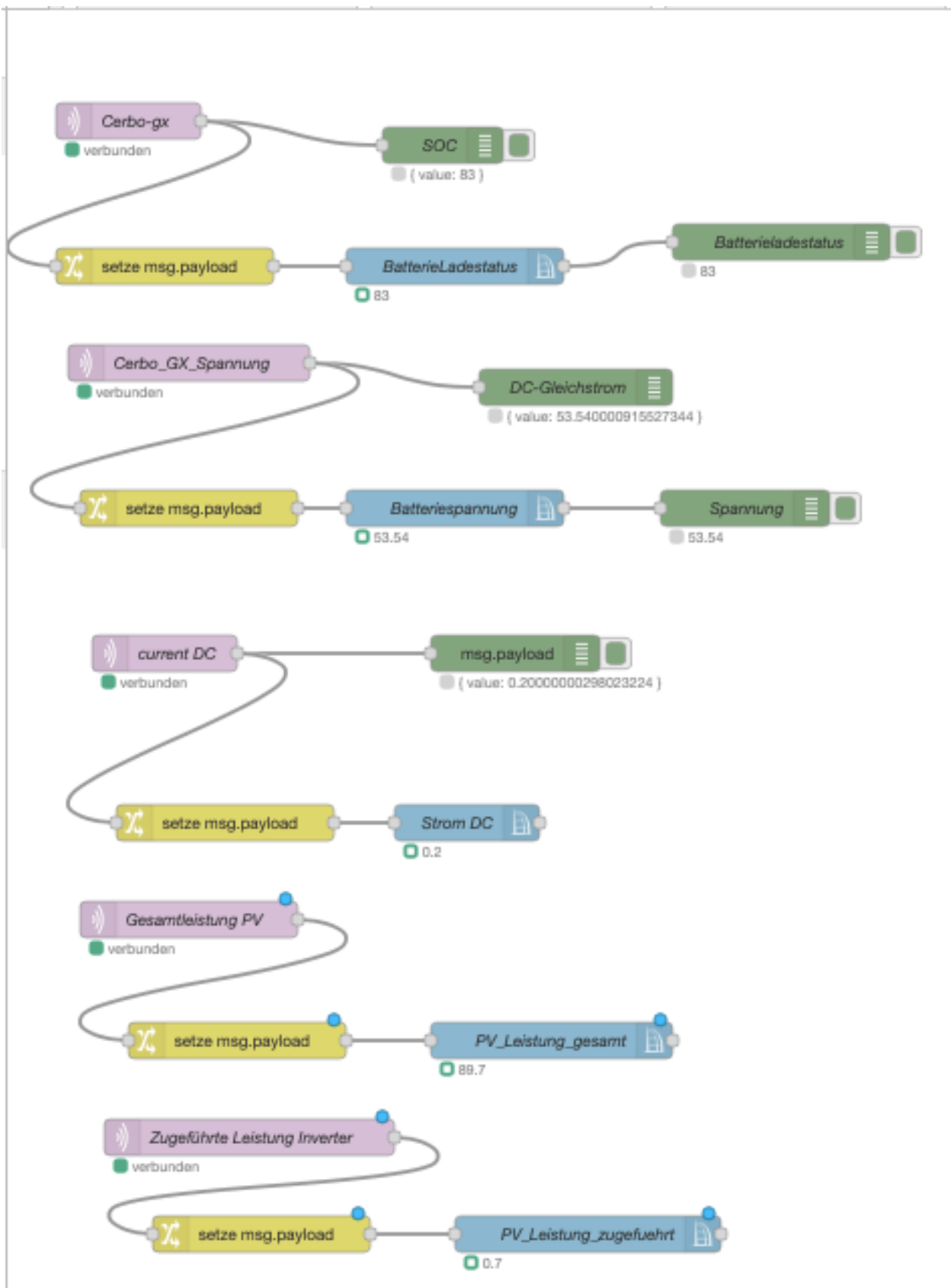
☐ HmIP-RF

☐ VirtualDevices

☐ Aktiviert

15 -Nodes verwenden diese Konfiguration

Alle anderen gewünschten Werte werden dazu analog konfiguriert. Insgesamt sieht das dann, bei 5 Systemvariablen, wie folgt aus:



alle

Die Werte können dann in der Homematic weiter verarbeitet werden oder z.B. auf Anzeigetableaus etc. ausgegeben werden, wie hier z.B. bei Mediola:

