Programmierung BMS Controller
(07.04.2023, Hartmut Buschke)

Das Batterie-Management-System, das ich benutze, habe ich von

https://github.com/stuartpittaway/diyBMSv4

übernommen. Zugeben muss ich allerdings, dass ich damit ohne Hilfe der Programmierprofis aus der Familie überfordert gewesen wäre und auch jetzt noch nicht alles nachvollziehen kann.

Ein Detail des Systems, die Vorbereitung und Inbetriebnahme des Controllers, beschreibe ich hier, weil ich mich damit nach einem selbst verschuldeten Fehlerfall intensiv befassen musste.

1. Die Firmware Datei herunterladen

In der oben angeführten Quelle findet man das komplette Softwarepaket in einer ZIP-Datei:

Compiled_Firmware_2022-05-24-08-14.zip

Nach dem entpacken interessiert uns für die Programmierung des Controllers nur die Datei

diybms_controller_firmware_espressif8266_esp8266_d1mini.bin

Weitere Dateien werden auf dem Controller nicht installiert, das war wohl bei älteren Softwareversionen noch erforderlich.
Als Controller verwende ich den ESP 8266 Wemos D1 mini v3
Das Modul hat einen USB Anschluss und mein Rechner ein Ubuntu Betriebssystem.

2. Datei installieren

Zur Kommunikation mit meinem ESP habe ich in der Konsole meines Ubuntu Rechners ein Programm installiert:

```
sudo apt install esptool
```

Ob die Verbindung zum ESP funktioniert, lässt sich, nachdem geprüft wurde, ob sich der USB Port als „ttyUSB0“ angemeldet hat, mit folgendem Befehl leicht kontrollieren:

```
esptool --port /dev/ttyUSB0 chip_id
```

Der Befehl holt mehrere Informationen aus dem ESP, darunter z.B. die MAC Adresse. Wenn das funktioniert, kann die Firmwaredatei überspielt werden, die natürlich im benutzten Ordner vorhanden sein muss.

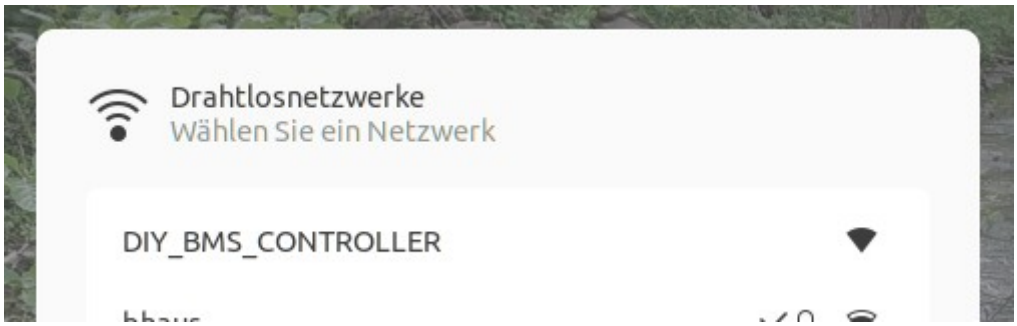
```
esptool --port /dev/ttyUSB0 write_flash 0x0
diybms_controller_firmware_espressif8266_esp8266_d1mini.bin
```

Nach der Installation startet das Programm automatisch und der ESP macht ein eigenes WLAN auf.

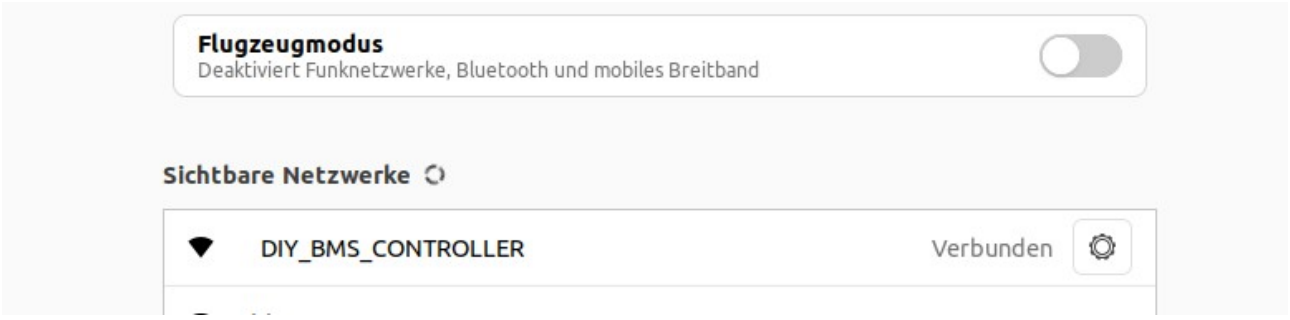
3. WLAN Zugang anmelden

Das eigene WLAN des Controllers ist nur hilfsweise erforderlich, um die Zugangsdaten des hauseigenen WLAN eintragen zu können, damit der Controller im LAN erreichbar ist und er auch selbst dort Daten verschicken kann.

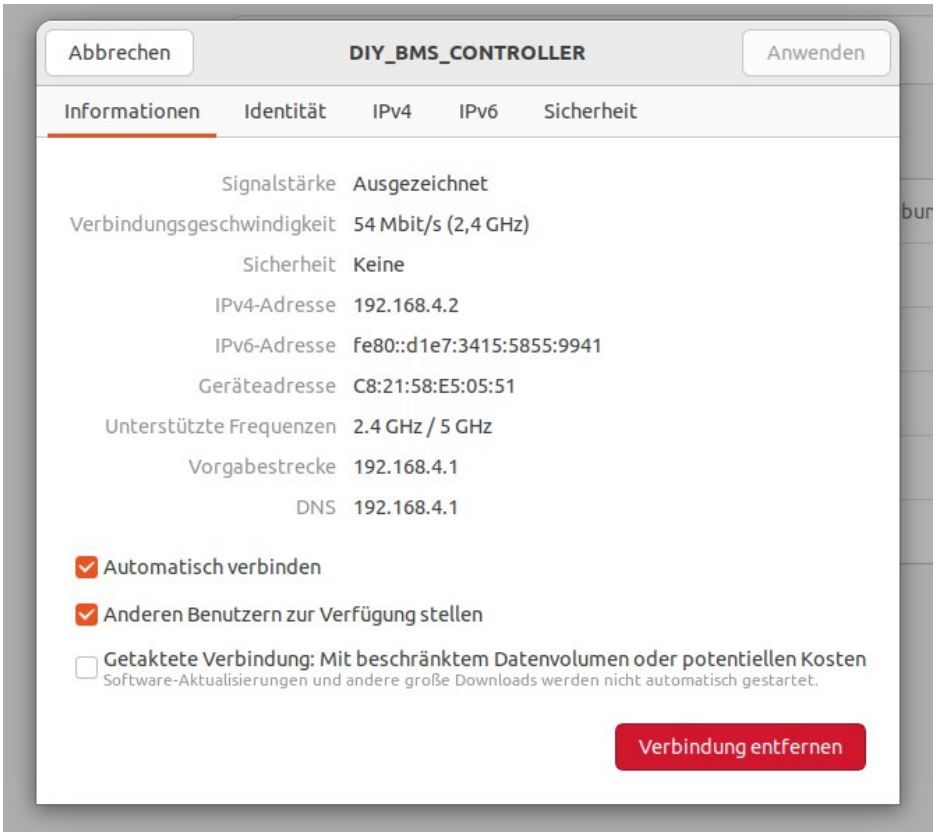
Zunächst suchen wir also den WLAN Sender des ESP mit einem WLAN fähigen Gerät, z.B. Laptop oder Smartphone:



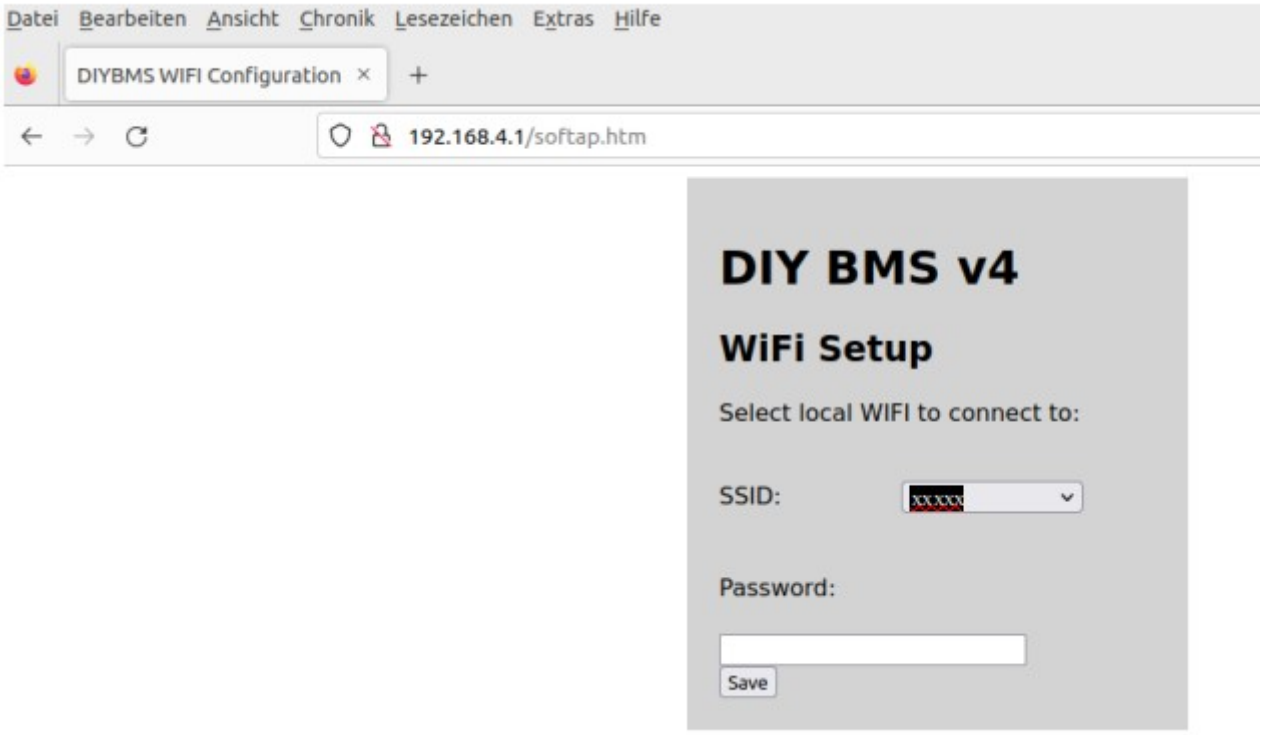
Der programmierte ESP meldet sich mit der SSID „DIY_BMS_CONTROLLER“. Die Verbindung ist nicht Passwortgeschützt, kann also sofort hergestellt werden.



Steht die Verbindung, kann in den Einstellungen des Rechners die IP ermittelt werden, mit der die Verbindung zum WLAN des Controllers hergestellt wurde:



Das wird so sein, wie in meinem Beispiel, der ESP will mit 192.168.4.1 angesprochen werden und hat meinem Rechner die 192.168.4.2 vergeben.
Die so ermittelte IP Adresse wird in einen Internetbrowser eingetragen.



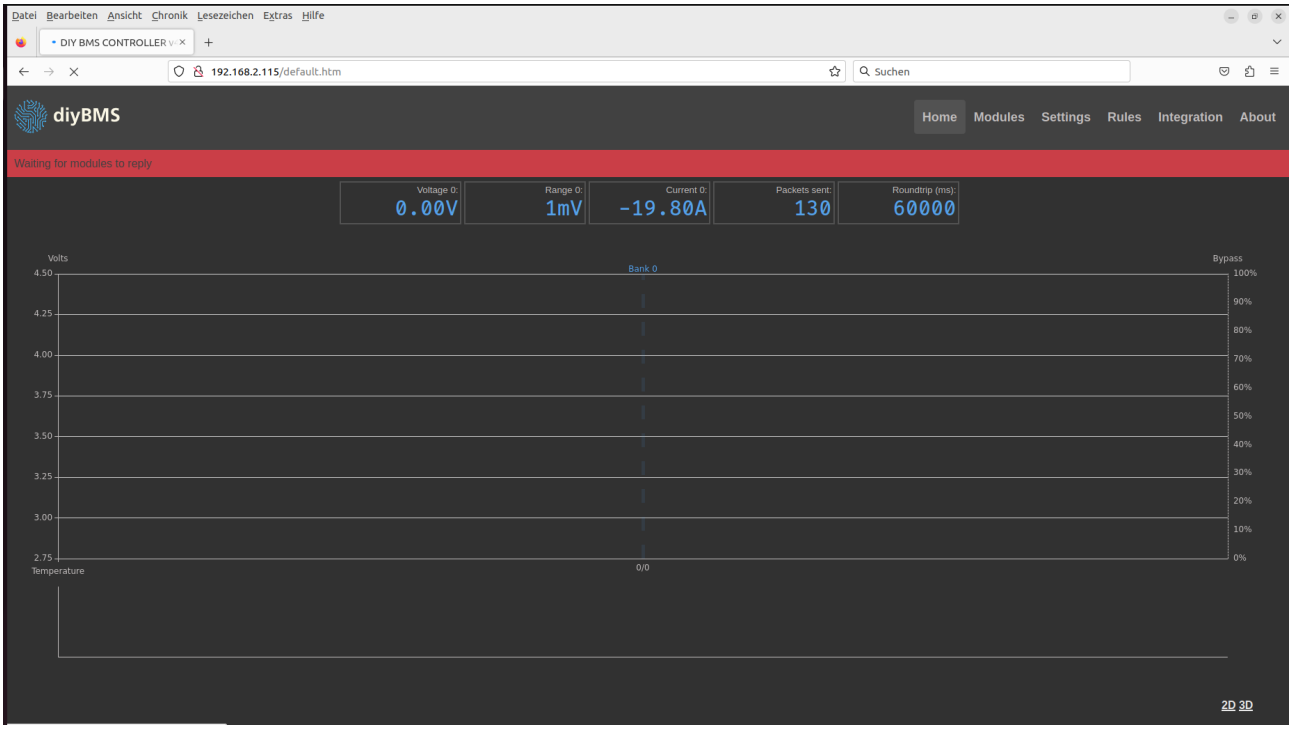
Der ESP liefert damit die Datei „softap.htm“ aus. In der Maske kann die SSID des eigenen Funknetzes ausgewählt werden. Voraussetzung ist natürlich, dass der Controller dieses Netz erreichen konnte. Hier ist anschließend auch der passende Schlüssel einzutragen und alles abzuspeichern.

Ist der Vorgang erfolgreich und der ESP bekommt eine Verbindung zum hauseigenen Router, was einige Minuten dauern kann, löscht er diese Routine und ist künftig über das LAN erreichbar.

4. Einstellungen

Zunächst wird im Router ermittelt, welche IP der DHCP an den ESP vergeben hat. Diese IP wird wieder mit einem Internetbrowser aufgerufen. In meiner Fritz-Box habe ich gleich auch einen Gerätenamen eingetragen, den ich mir merken kann, und die Box angewiesen die vergebene IP für die MAC Adresse dieses Gerätes zu reservieren.

Wenn der Browser folgendes Bild ausgibt, ist schon viel erreicht:



****

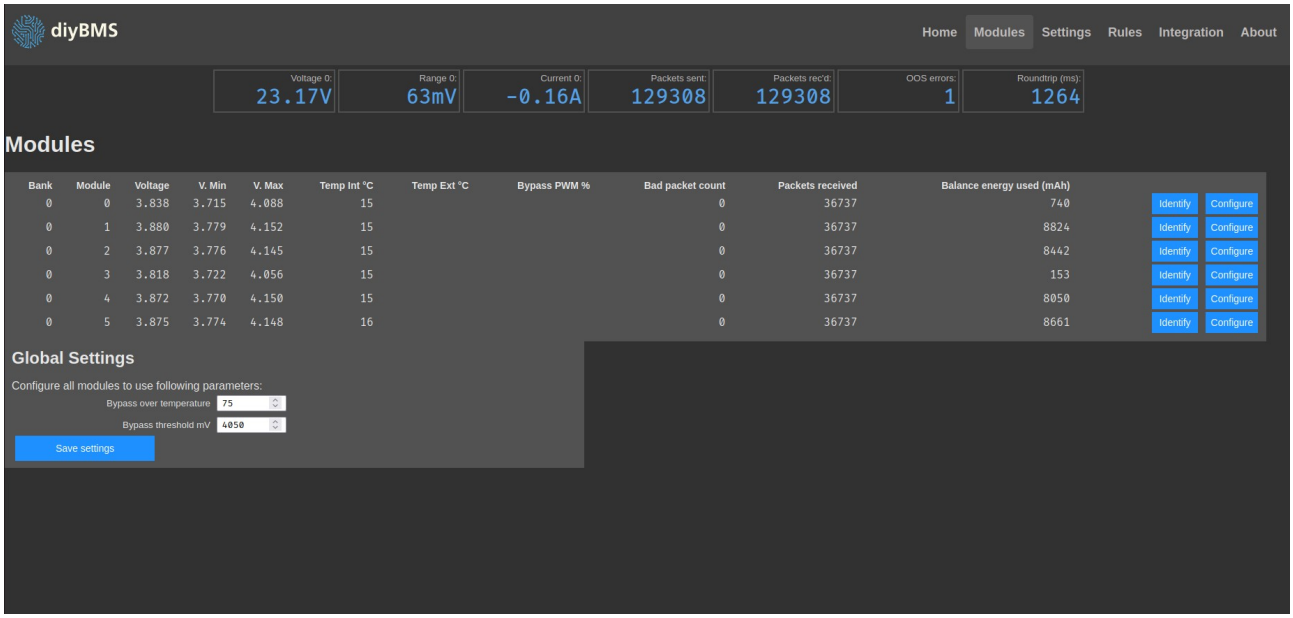
An dieser Stelle ist eine Lücke in meiner Dokumentation, weil ich beschreiben müsste, wie die gesamte Hardware aufgebaut wird und die Zellsensoren zu programmieren sind.

****

Vorausgesetzt die Zellsensoren, bei mir sind es sechs, sind am Controller angeschlossen, baut der Controller eine Verbindung zu den Sensoren auf und ließt die Spannungswerte ein.

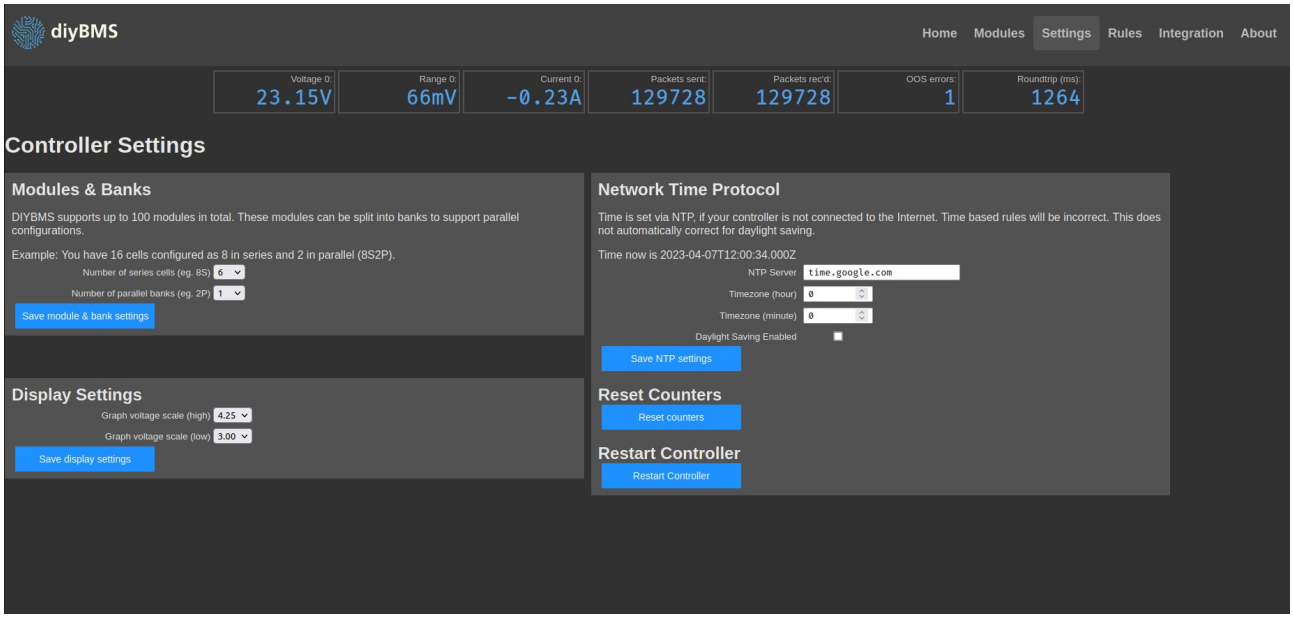
Mit der vollständig aufgebauten Hardware und sechs Zellsensoren sieht die Ausgabe so aus:

Die Spannungswerte der einzelnen Zellblöcke können mit einem Multimeter genau ermittelt und als Konfigurationswert erfasst werden:

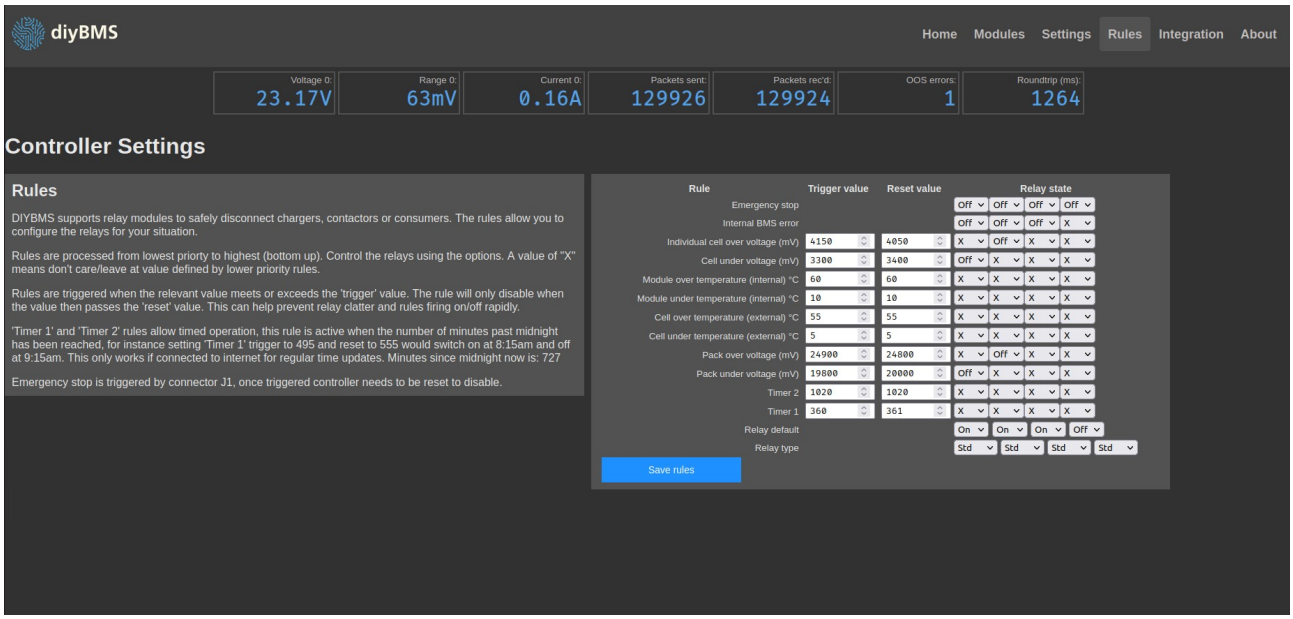


In dieser Eingabemaske wird auch der Startwert eingetragen, ab dem das System beginnt, Zellunterschiede auszugleichen. Ich fand es sinnvoll, den Wert 100 mV unterhalb der eingestellten Ladeschlussspannung anzusetzen.

Hier wird die Anzahl der Zellblöcke erfasst und Einstellungen für die Anzeige gemacht. Die Zeiterfassung ist notwendig, weil das System auch zeitgesteuerte Ausgaben machen kann:



Im folgenden Menü werden die Bedingungen für die vier möglichen Schaltvorgänge eingestellt, wobei ich gegenwärtig nur die Endabschaltung bei erreichen der Ladeschlussspannung und die Endabschaltung zur Verhinderung der Tiefentladung nutze:



Theoretisch könnte die Ladeschlussspannung auch auf 4,2 Volt eingestellt werden und man könnte die Zellen auch tiefer entladen. Ich habe mich für diese Werte entschieden und hoffe, dass die Zellen damit weniger Stress haben und länger halten.