Kolloquium

Ansteuerungseinheit für Ladeboxen für E-Autos

VON FELIX SEIBT

Inhaltsverzeichnis

Problemstellung

- Ansteuerungseinheit entwickeln
- Oft verwendetes Protokoll
- Relevante Daten lesen und schreiben
- einfachen Benutzeroberfläche

Wahl des Protokolls

- Auswahlmöglichkeiten
- EEBUS, Modbus, TCP, MQTT, RTU(Grundlage Modbus)

- OCPP 1.6 (8 / 10 Fälle)
- Unterschied Soap und JSON
- OCPP 1.6 J auf 2.0.1J upgraden

Protokoll (OCPP)

- OCPP Server wartet auf Verbindungsanfrage
- Empfängt regelmäßig Nachrichten (Heartbeat)
- Statusupdate, Start / Stopp Transaktion
- Kann Befehle senden (Start Stopp)
- Status Nachrichten triggern

Wahl der Wallbox

- Unterstützung des OCPP 1.6 - Protokolls
- Ladeleistung 11kW (möglich zwischen 3,7 kW und 22 kW)

- DC-Fehlerstromerkennung
- optional, Autorisierungsmöglichkeit

Wahl der Wallbox

- WB24 der EC Serie
- Modulare Bauweise
- einfache Erweiterung der Funktionalität
- Ladestecker mitinbegriffen (Typ2)
- Bei Bedarf einzelne Bauteile zum ersetzen

Wahl der Wallbox

- ABB Hauptschalter
- 12 V Netzteil
- Messtromwandler
- Laderegler mit Schnittstelle

Wahl der Ansteuerungseinheit

- Einplatinencomputer
- Große Auswahl
- Unihiker, AUSUS Thinker Borad, Banana Pi, Rockpi(Win10)
- Entscheidung für Raspberry Pi 4B → zugehöriges Display
- Keine größere Vielzahl an ungenutzten Ein- und Ausgängen

Programmierung

- Programmiersprache Python gewählt
- Einfache Syntax
- Umfangreiche Bibliotheken
- Dynamische Typisierung

- Alternativen Java, C#, C++,
  Webserver mit HTML,CSS und JavaSkript

Programmierung

- 2 Python Skripte
- Webinterface
- OCPP-Webserver

- Verbindung zwischen beiden Skripten durch RabbitMQ
- Webserver Flexibilität

Programmierung

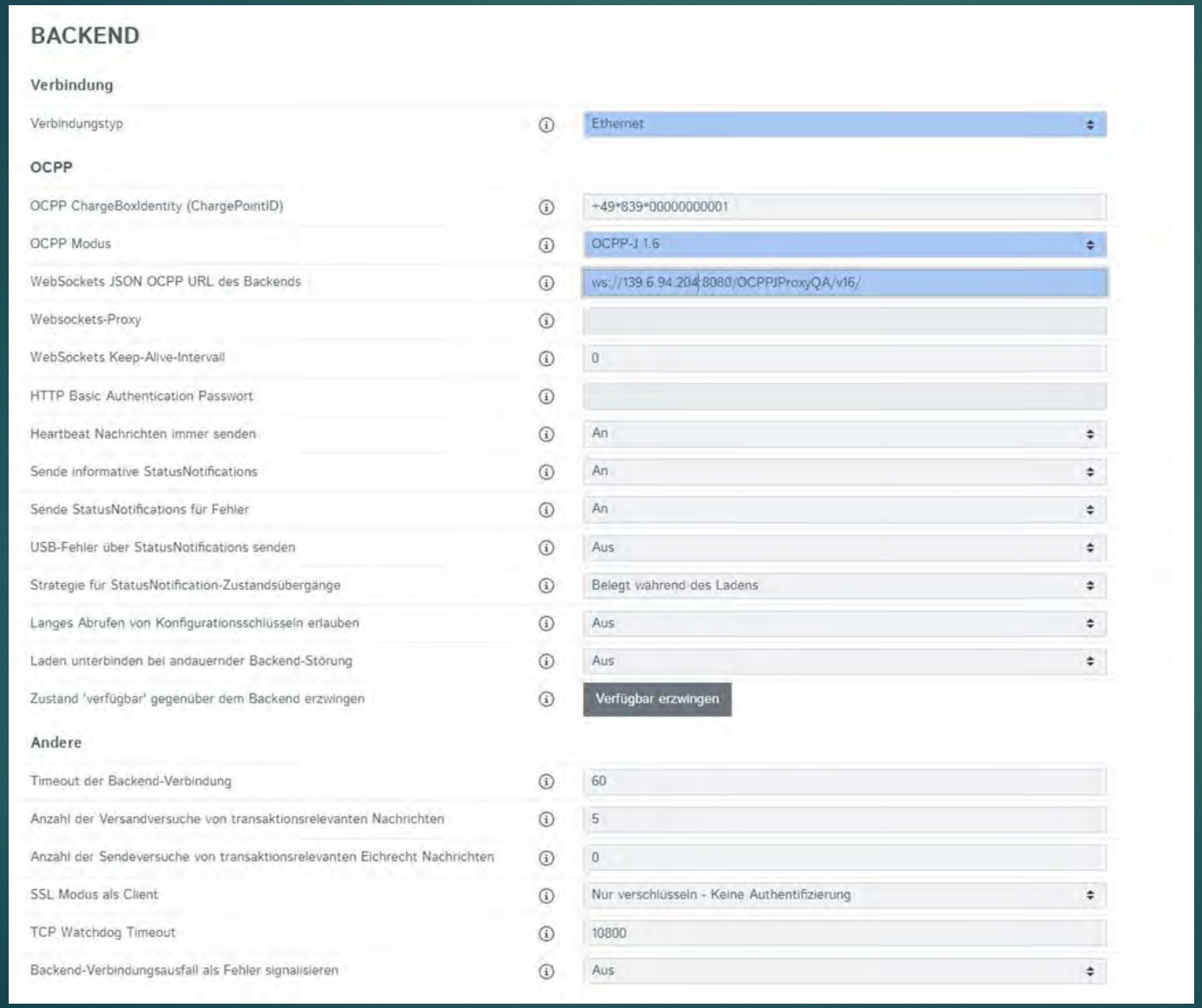
Programmierung

- Parallel Ausführung der Prozesse
- OCPP (Nachrichten Verarbeitung und OCPP –Part)
- Web (Nachrichten und Webinterface zur Verfügung stellen

Konfiguration der Wallbox

- Verbindung zur Wallbox (über Ethernet, IP-Adresse oder Hostname)
- Anmelden
  Standard Anmeldedaten:
  User: operator,
  Password: yellow_zone)

Konfiguration der Wallbox



BACKEND

Verbindung

Verbindungstyp	Ethernet

OCPP

OCPP ChargeBoxIdentity (ChargePointID)	+49*839*00000000001
OCPP Modus	OCPP-J 1.6
WebSockets JSON OCPP URL des Backends	ws://139.6.94.204:8080/OCPPJProxyQA/v16/
Websockets-Proxy	
WebSockets Keep-Alive-Intervall	0
HTTP Basic Authentication Passwort	
Heartbeat Nachrichten immer senden	An
Sende informative StatusNotifications	An
Sende StatusNotifications für Fehler	An
USB-Fehler über StatusNotifications senden	Aus
Strategie für StatusNotification-Zustandsübergänge	Belegt während des Ladens
Langes Abrufen von Konfigurationsschlüsseln erlauben	Aus
Laden unterbinden bei andauernder Backend-Störung	Aus
Zustand 'verfügbar' gegenüber dem Backend erzwingen	Verfügbar erzwingen

Andere

Timeout der Backend-Verbindung	60
Anzahl der Versandversuche von transaktionsrelevanten Nachrichten	5
Anzahl der Sendeversuche von transaktionsrelevanten Eichrecht Nachrichten	0
SSL Modus als Client	Nur verschlüsseln - Keine Authentifizierung
TCP Watchdog Timeout	10800
Backend-Verbindungsausfall als Fehler signalisieren	Aus

Ergebnis

- Software der Ansteuerungseinheit auf dem Raspberry Pi
- Webinterface zum Steuern der Wallbox
- Webinterface zeigt aktuelle Daten der Wallbox an
- Wallbox mit 11 kW Leistung und OCPP 1.6 verwendet
- Demo im Labor und der Praxis

Fazit

- viele Wallboxen und Ansteuerungseinheiten
- Protokolle für Wallboxen herstellerübergreifende Nutzung
- Einbindung in ein dezentrales Lastmanagement
- Steuern je nach Modell nur ein / aus
- Leistung drosseln

Danke für ihre Aufmerksamkeit!