

# Projektzusammenfassung: EV-LKW-Laden Phase 2

Frequenz EaaS

12. September 2025

## Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>2</b>
<b>2</b>	<b>Systemarchitektur und Design</b>	<b>2</b>
2.1	Cloud-basierter Handels-Akteur . . . . .	2
2.2	Dynamische Reaktion vor Ort . . . . .	3
<b>3</b>	<b>Geschäftliche Bewertung</b>	<b>4</b>
<b>4</b>	<b>Fazit und nächste Schritte</b>	<b>4</b>

## 1 Einleitung

Aufbauend auf der in Phase 1 geschaffenen Grundlage befasst sich die zweite Phase des Projekts “EV-LKW-Laden” mit den wirtschaftlichen Aspekten des Energieverbrauchs. Während sich Phase 1 auf die Gewährleistung der Netzstabilität und Betriebssicherheit konzentrierte, führt Phase 2 eine Ebene der wirtschaftlichen Optimierung durch die Integration von Echtzeit-Marktdaten in die Steuerungslogik ein.

Die Hauptziele für diese Phase waren:

- **Entwurf einer übergeordneten Cloud-Anwendung:** Entwicklung einer Cloud-basierten Anwendung zur Verwaltung der Strombeschaffung am Spotmarkt.
- **Optimierungsparameter:** Definition und Implementierung von Optimierungsparametern wie Ladeschwellen für E-Fahrzeuge und Zielen zur Spitzenlastkappung auf Basis von Marktpreisen.
- **Implementierung einer Handelsstrategie:** Implementierung einer Handelsstrategie, die Intraday oder Day-Ahead-Marktdaten nutzt, um den Energieverbrauch vor Ort zu steuern.
- **Dynamische Interaktion mit dem Energiemarkt:** Schaffung eines Systems, das dynamisch mit dem Energiemarkt interagieren kann, um Kosten zu optimieren.

Dieser Bericht beschreibt den Entwurf und die Implementierung des Cloud-basierten Handels-Akteurs und seine Integration in das lokale Edge-Control-System und zeigt damit einen bedeutenden Schritt in Richtung einer vollständig autonomen und wirtschaftlich optimierten Energiemanagementlösung.

## 2 Systemarchitektur und Design

### 2.1 Cloud-basierter Handels-Akteur

Phase 2 führt eine neue Cloud-basierte Komponente ein, den **PowerCurtailmentActor**, der für die Überwachung des Energiemarktes und das Treffen strategischer Entscheidungen verantwortlich ist. Dieser Akteur läuft in der Cloud und kommuniziert mit den Systemen vor Ort über die Frequenz Dispatch API.

Die Hauptverantwortlichkeiten des **PowerCurtailmentActor** sind:

- **Marktdaten abrufen:** Er ruft regelmäßig Day-Ahead-Strompreise von der öffentlichen API von ENTSO-E (Verband Europäischer Übertragungsnetzbetreiber) ab.
- **Handelsstrategie bewerten:** Er vergleicht die abgerufenen Preise mit einem konfigurierbaren Schwellenwert.
- **Dispatches senden:** Wenn der Preis den Schwellenwert überschreitet, sendet er einen Dispatch an das Microgrid vor Ort und weist es an, den Stromverbrauch zu reduzieren.

Dieses Design trennt die übergeordnete strategische Entscheidungsfindung (die in der Cloud erfolgen kann) von der untergeordneten Echtzeitsteuerung (die am Edge erfolgen muss).

Der folgende Codeausschnitt aus dem **PowerCurtailmentActor** zeigt die Logik zum Abrufen von Preisen und zum Senden von Dispatches:

```
1 async def _process_available_prices(  
2     self, da_prices: pd.Series, contracts: list[pd.Timestamp]  
3 ) -> None:  
4     """Process contracts that have available price data."""  
5     limited_price_count = 0  
6  
7     for timestamp in contracts:
```

```

8         if timestamp in da_prices.index:
9             price = da_prices[timestamp]
10
11         if price > self._config.price_limit:
12             dispatch_success = await self._send_dispatch(timestamp)
13             if dispatch_success:
14                 limited_price_count += 1
15             else:
16                 # Don't mark as processed if dispatch failed
17                 continue
18
19         self._active_contracts[timestamp] = True
20
21     if limited_price_count > 0:
22         _logger.info(
23             "Found %d prices above limit %.2f, sent charge curtailment dispatch
24             contracts",
25             limited_price_count,
26             self._config.price_limit,
27         )

```

Listing 1: Logik zum Abrufen von Preisen und zum Versenden von Aktionen.

## 2.2 Dynamische Reaktion vor Ort

Der `TruckChargingActor` aus Phase 1 wurde erweitert, um Dispatches aus der Cloud zu abonnieren. Wenn ein Dispatch empfangen wird, passt der Akteur seine Betriebsparameter an, um die Anforderungen der Handelsstrategie zu erfüllen.

Die wichtigsten Änderungen am `TruckChargingActor` umfassen:

- **Dispatch-Abonnement:** Der Akteur abonniert jetzt die Frequenz Dispatch API, um Nachrichten vom Typ `SE_TRUCK_CHARGING` zu empfangen.
- **Parameteranpassung:** Nach Erhalt eines Dispatches analysiert er die Nutzdaten, um die erforderliche Leistungsreduzierung zu bestimmen.
- **Dynamische Steuerung:** Anschließend passt er seine Netzziel-Leistung an, was wiederum das Verhalten der `TcControlLogic` beeinflusst, um das Laden von E-Fahrzeugen zu drosseln und/oder die Batterie zu entladen.

Dies ermöglicht es dem System vor Ort, dynamisch auf Marktbedingungen zu reagieren, den Energieverbrauch bei hohen Preisen zu senken und so die Betriebskosten zu senken.

Der folgende Codeausschnitt aus dem `TruckChargingActor` zeigt, wie er eingehende Dispatches behandelt:

```

1 def _apply_dispatch_info(self, dispatch_info: DispatchInfo) -> None:
2     """Apply a dispatch info."""
3     _logger.info("Received new dispatch info: %s", dispatch_info)
4     if reduction_w := dispatch_info.options.get("power_reduction_w"):
5         try:
6             power_reduction = Power.from_watts(float(reduction_w))
7             self._target_power = self._config.target_power - power_reduction
8             _logger.info(
9                 "Dispatch: Reducing target power by %s. New target power: %s",
10                 power_reduction,
11                 self._target_power,
12             )
13         except (ValueError, TypeError) as exc:
14             _logger.error("Dispatch: Failed to set target power reduction: %s", exc)
15     else:
16         self._target_power = self._config.target_power

```

```
17     _logger.info(  
18         "Dispatch: No power reduction specified. Resetting to configured target  
power: %s",  
19         self._target_power,  
20     )  
21  
22     if self._tc_control_logic:  
23         self._tc_control_logic.set_target_power(self._target_power)
```

Listing 2: Behandlung von Dispatches im TruckChargingActor.

### 3 Geschäftliche Bewertung

Die Einführung eines Cloud-basierten Handels-Akteurs in Phase 2 erschließt einen erheblichen neuen Geschäftswert, indem sie dem System ermöglicht, aktiv am Energiemarkt teilzunehmen. Dies verwandelt das Microgrid in einen aktiven, wirtschaftlich bewussten Teilnehmer des Energiesystems.

Die primären geschäftlichen Vorteile dieser Phase umfassen:

- **Direkte Kosteneinsparungen:** Durch die automatische Reduzierung des Verbrauchs bei hohen Strompreisen senkt das System direkt die Energiekosten. Dies ist besonders wirkungsvoll bei energieintensiven Vorgängen wie dem Laden von Lastwagen.
- **Neue Einnahmequellen:** Die Architektur legt den Grundstein für fortschrittlichere Handelstrategien. Zum Beispiel könnte das System erweitert werden, um Netzdienstleistungen wie Frequenzregulierung oder Demand Response anzubieten und so neue Einnahmequellen zu schaffen.
- **Gesteigerte Rentabilität:** Durch die Optimierung des Energieverbrauchs auf der Grundlage von Marktsignalen steigert das System die Gesamtrentabilität des Betriebs. Es stellt sicher, dass Energie auf die wirtschaftlich vorteilhafteste Weise genutzt wird.
- **Verbesserte Nachhaltigkeit:** Indem das System in Spitzenzeiten (die oft mit der Stromerzeugung aus fossilen Brennstoffen verbunden sind) weniger Energie verbraucht, kann es zu einem nachhaltigeren Energiesystem beitragen.

Diese Phase demonstriert das Potenzial für erhebliche finanzielle Gewinne durch die Nutzung von Marktchancen und die Optimierung des Energieverbrauchs in Echtzeit.

### 4 Fazit und nächste Schritte

Phase 2 des Projekts "EV-LKW-Laden" hat das Energiemanagementsystem vor Ort erfolgreich um eine Cloud-basierte Handelsfunktion erweitert. Durch die Integration von Echtzeit-Marktdaten kann das System nun wirtschaftlich optimierte Entscheidungen treffen und seinen Geschäftswert erheblich steigern. Das Projekt hat erfolgreich eine übergeordnete Cloud-Anwendung zur Verwaltung der Energiebeschaffung am Spotmarkt und einen dynamischen Reaktionsmechanismus vor Ort demonstriert.

Der erfolgreiche Abschluss dieser Phase eröffnet mehrere Wege für die zukünftige Entwicklung:

- **Fortschrittliche Handelsalgorithmen:** Implementierung komplexerer Handelsalgorithmen, die Markttrends vorhersagen, Lade-/Entladezyklen von Batterien gewinnorientiert optimieren und an anderen Energiemärkten (z. B. Intraday, Regelenergie) teilnehmen können.
- **Integration mit anderen Datenquellen:** Einbindung anderer Datenquellen wie Wettervorhersagen (zur Vorhersage der Solarstromerzeugung) und Flottenpläne, um die Optimierungslogik weiter zu verfeinern.

- **Vollständige Bereitstellung und Validierung:** Einsatz des Systems in einer Live-Produktionsumgebung, um seine Leistung zu validieren und die wirtschaftlichen Vorteile über einen längeren Zeitraum zu quantifizieren.
- **Benutzeroberfläche für die Handelsstrategie:** Entwicklung einer Benutzeroberfläche, die es den Betreibern ermöglicht, die Handelsstrategien zu konfigurieren und zu überwachen, Risikoparameter festzulegen und Leistungsberichte einzusehen.

Dieses Projekt hat eine robuste Grundlage für eine umfassende, intelligente und wirtschaftlich ausgerichtete Energiemanagementlösung für E-LKW-Flotten gelegt.