Projektzusammenfassung: EV Truck Charging Phase 1

Frequenz EaaS

2. September 2025

Inhaltsverzeichnis

1 Einleitung **2**

2 Systemarchitektur und Design **2**
2.1 Sensor- & Zähler-Integration 3
2.2 Implementierung der Edge-Steuerung 3
2.3 Visualisierung des Energieflusses 4

3 Tests und Validierung **5**

4 Geschäftlicher Mehrwert **5**

5 Fazit und nächste Schritte **5**

1 Einleitung

Der Übergang zu Flotten von Elektrofahrzeugen (EV), insbesondere bei schweren Lastkraftwagen, stellt eine erhebliche Herausforderung für die Energieinfrastruktur dar. Der hohe Leistungsbedarf für das Laden kann die lokalen Netze belasten, was zu Instabilität und erhöhten Kosten führt. Um diesem Problem zu begegnen, haben Siemens Energy und Frequenz EaaS das Projekt „EV Truck Charging“ initiiert, um eine Lösung für ein intelligentes On-Site-Energiemanagement zu entwickeln.

Phase 1 dieses Projekts, das Thema dieses Berichts, konzentrierte sich auf das Design und die Entwicklung einer grundlegenden Softwareanwendung. Die Hauptziele für diese Phase waren:

- **High-Level-Anwendungsdesign:** Schaffung einer erweiterbaren und robusten Softwarearchitektur zur Optimierung der Energieflüsse innerhalb eines Microgrids.
- **Fokus auf On-Site-Messungen:** Integration mit simulierter On-Site-Hardware, einschließlich Truck-Charger, Batterien, PV Installation und dem öffentlichen Netz, um Echtzeit-Energiedaten zu sammeln.
- **Edge-Steuerungslogik:** Implementierung eines lokalen Steuerungssystems, das in der Lage ist, Echtzeitentscheidungen zu treffen, um einen zuverlässigen und effizienten Betrieb zu gewährleisten.
- **Systemtransparenz:** Bereitstellung eines Live-Dashboards zur Überwachung der Energieflüsse und des Systemstatus.
- **Wissenstransfer:** Angebot von fortlaufenden SDK-Schulungen für Mitarbeiter von Siemens Energy, um den langfristigen Projekterfolg und die Wartbarkeit zu sichern.

Dieser Bericht beschreibt die Architektur, Implementierung und Ergebnisse der Phase 1 und zeigt die erfolgreiche Erreichung dieser Ziele auf.

2 Systemarchitektur und Design

Die Anwendung basiert auf dem Frequenz SDK, das ein robustes Framework für die Entwicklung von Energiemanagement-Anwendungen bietet. Die Architektur basiert auf dem Aktor-Modell, einem Paradigma, das die Entwicklung von nebenläufigen und verteilten Systemen erleichtert. Jede Komponente des Systems ist ein „Aktor“, der unabhängig läuft und mit anderen Aktoren über asynchronen Nachrichtenaustausch kommuniziert. Dieses Design gewährleistet eine hohe Verfügbarkeit und Skalierbarkeit.

Das Herzstück der Anwendung ist der `TruckChargingActor`, der als zentraler Orchestrator dient. Er ist verantwortlich für:

- Das Abonnieren von Echtzeit-Datenströmen von verschiedenen Microgrid-Komponenten.
- Die Verarbeitung der eingehenden Daten und die Aufrechterhaltung des Systemzustands.
- Das Aufrufen der Steuerungslogik, um Entscheidungen auf der Grundlage des aktuellen Zustands zu treffen.
- Das optionale Melden von Schlüsselmetriken an eine Zeitreihendatenbank zur Überwachung und Visualisierung.

Diese modulare, aktorbasierte Architektur macht das System hochgradig erweiterbar, sodass neue Funktionalitäten und Steuerungsstrategien mit einfachen Änderungen am bestehenden Code hinzugefügt werden können.

2.1 Sensor- & Zähler-Integration

Ein zentrales Ergebnis der Phase 1 war die erfolgreiche Integration mit simulierten On-Site-Energiemessgeräten. Die Anwendung nutzt das Frequenz SDK, um Verbindungen zu verschiedenen Microgrid-Komponenten herzustellen und deren Echtzeit-Datenströme zu abonnieren.

Der `TruckChargingActor` initialisiert und verwaltet Abonnements für die folgenden Datenquellen:

- **Netzleistung:** Die Hauptleistungszufuhr aus dem öffentlichen Netz, bezogen über `microgrid.grid().power`.
- **Batterie-Pool:** Die aggregierte Leistung und der Ladezustand (SoC) der On-Site-Batterien, verwaltet durch `microgrid.new_battery_pool()`.
- **EV-Ladegerät-Pool:** Der kombinierte Stromverbrauch aller Truck-Charger, zugänglich über `microgrid.new_ev_charger_pool()`.
- **Erzeugerleistung:** Die von On-Site-Anlagen wie Solarmodulen erzeugte Leistung, verfügbar über `microgrid.producer().power`.

Der Aktor verwendet eine asynchrone `select`-Anweisung, um effizient auf alle diese Datenströme gleichzeitig zu lauschen. Wenn eine neue Messung empfangen wird, aktualisiert der Aktor seinen internen Zustand und stellt so sicher, dass die Steuerungslogik immer mit den aktuellsten Daten arbeitet. Diese Echtzeit-Datenerfassung ist die Grundlage für die intelligenten Entscheidungsfähigkeiten des Systems.

2.2 Implementierung der Edge-Steuerung

Die Kernintelligenz der Anwendung liegt in der Klasse `TcControlLogic`, welche die Edge-Steuerungslogik implementiert. Diese Komponente ist dafür verantwortlich, Echtzeitentscheidungen zu treffen, um die Netzstabilität zu gewährleisten und den Energieverbrauch zu optimieren. Die Steuerungslogik wird immer dann ausgeführt, wenn eine neue Netzleistungsmessung empfangen wird.

Das Hauptziel der Steuerungslogik ist es, den Stromverbrauch aus dem Netz unter einem vordefinierten Zielwert (z.B. 2.5 MW) zu halten. Die Logik folgt diesen Schritten:

1. **Prüfung auf Überverbrauch:** Wenn die aktuelle Netzleistung den Zielwert überschreitet, ergreift das System korrigierende Maßnahmen.
   - Wenn die Batterie ausreichend geladen ist (SoC $> 10\%$), wird die Batterie angewiesen, zu entladen, um den überschüssigen Leistungsbedarf zu decken.
   - Wenn die Batterie erschöpft oder nicht verfügbar ist, wird die Stromversorgung der Truck-Charger gedrosselt, um die Netzleistung wieder in den Zielbereich zu bringen.
2. **Prüfung auf überschüssige Leistung:** Wenn die Netzleistung negativ ist (d.h. Strom wird exportiert), deutet dies auf einen Überschuss an On-Site-Erzeugung hin. In diesem Fall wird die Batterie angewiesen, mit diesem überschüssigen Strom zu laden.
3. **Keine Aktion:** Wenn sich die Netzleistung im Zielbereich befindet, wird keine Steuerungsaktion durchgeführt und Truck Charging wird nicht gedrosselt.

Der folgende Code-Ausschnitt aus der Methode `TcControlLogic.perform` veranschaulicht die Logik zur Behandlung im Fall des überschreitens der vorab definierten Netzleistungsgrenze:

```
if latest_grid_power > self._target_power:
    if latest_battery_soc > self._min_soc:
        restrict_power = latest_grid_power - self._target_power
        battery_discharge_power = max(
            -restrict_power, latest_battery_max_discharge_power
        )

        ev_restriction_power = max(
            restrict_power + latest_battery_max_discharge_power, Power.zero()
        )
        ev_charge_power = max(
            latest_ev_charger_max_power - ev_restriction_power, Power.zero()
        )

        # Discharge Battery & Restrict EV
        await self._battery_pool.propose_power(battery_discharge_power)
        await self._ev_charger_pool.propose_power(ev_charge_power)
    else:
        # Restrict EV Power only
        ev_restriction_power = max(
            latest_ev_charger_max_power
            - (latest_grid_power - self._target_power),
            Power.zero(),
        )
        await self._ev_charger_pool.propose_power(ev_restriction_power)
```

Listing 1: Steuerungslogik zur Behandlung von Netzüberlastung.

Diese Logik stellt sicher, dass das Microgrid effizient arbeitet und hohe Kosten für Spitzenlastbezug aus dem Netz vermieden werden.

2.3 Visualisierung des Energieflusses

Um Transparenz zu schaffen und eine Echtzeit-Überwachung der Leistung des Microgrids zu ermöglichen, enthält die Anwendung eine Komponente zur Visualisierung des Energieflusses. Dies wird durch die Integration von InfluxDB, einer Zeitreihendatenbank, und Grafana, einem beliebten Datenvisualisierungstool, erreicht.

Die Klasse `InfluxReporter` ist dafür verantwortlich, Schlüsselmetriken aus der Anwendung zu sammeln und an eine InfluxDB-Instanz zu melden. Die folgenden Metriken werden gemeldet:

- Netzleistung (Watt)
- Batterieleistung (Watt)
- Batterieladezustand (SoC)
- EV-Ladegerätleistung (Watt)
- Erzeugerleistung (Watt)

Diese Metriken werden mit einem Zeitstempel versehen und in der Datenbank gespeichert, wodurch ein historischer Verlauf des Systembetriebs erstellt wird. Ein vorkonfiguriertes Dashboard verbindet sich mit dieser Datenbank, um Live-Visualisierungen dieser Metriken bereitzustellen. Dieses Dashboard ermöglicht es den Betreibern, den Zustand des Systems zu überwachen, Energieflussmuster zu verstehen und zu überprüfen, ob die Steuerungslogik wie erwartet funktioniert.

Die `README.md`-Datei des Repositorys enthält detaillierte Anweisungen zum Einrichten der Grafana- und InfluxDB-Dienste, was eine einfache Bereitstellung der Überwachungslösung ermöglicht.

3 Tests und Validierung

Der Projektumfang für Phase 1 betonte die Entwicklung einer erweiterbaren Anwendung mit einer Grundlage für robuste Software-Unit- und Integrationstests. Die modulare Architektur mit ihrer klaren Trennung der Verantwortlichkeiten zwischen dem Haupt-Aktor, der Steuerungslogik und der Datenberichterstattung ist so konzipiert, dass sie leicht testbar ist.

Umfassende Unit-Tests für die `TcControlLogic` wurden implementiert und befinden sich in `tests/test_ev_charging_main.py`. Diese Tests decken eine Vielzahl von Szenarien ab, darunter:

- Situationen, in denen keine Steuerungsaktion erforderlich ist, da die Netzleistung im Zielbereich liegt.
- Fälle, in denen die Batterie entladen und das Laden von E-Fahrzeugen aufgrund hoher Netzleistung eingeschränkt wird.
- Szenarien, in denen nur das Laden von E-Fahrzeugen eingeschränkt wird, da die Netzleistung hoch ist, der Ladezustand der Batterie jedoch zu niedrig ist.
- Fälle, in denen die Batterie bei negativer Netzleistung mit überschüssiger Energie geladen wird.

Diese Tests stellen die Korrektheit der Steuerungslogik sicher und validieren ihr Verhalten unter verschiedenen Bedingungen.

4 Geschäftlicher Mehrwert

Die in Phase 1 gelieferte Anwendung für das EV Truck Charging bietet einen erheblichen geschäftlichen Mehrwert, indem sie zentrale betriebliche und finanzielle Herausforderungen im Zusammenhang mit dem groß angelegten Laden von Elektrofahrzeugen adressiert.

Die primären geschäftlichen Vorteile umfassen:

- **Verbesserte betriebliche Effizienz:** Die automatisierte Steuerungslogik stellt sicher, dass das Microgrid optimal ohne manuelle Eingriffe arbeitet. Das System gleicht intelligent Stromangebot und -nachfrage aus und stellt sicher, dass die Truck-Charger bei Bedarf verfügbar sind, während das lokale Netz vor übermäßiger Belastung geschützt wird.
- **Reduzierte Energiekosten:** Durch die aktive Steuerung des Energieflusses hilft die Anwendung, die Energiekosten zu senken. Sie minimiert den Verbrauch aus dem Netz während der Spitzenlastzeiten durch die Nutzung von On-Site-Batteriespeichern und kann hohe Lastspitzenentgelte vermeiden, indem sie bei Bedarf nicht wesentliche Lasten drosselt.
- **Erhöhte Zuverlässigkeit und Stabilität:** Die Edge-Steuerungslogik bietet eine schnell reagierende, lokalisierte Antwort auf sich ändernde Bedingungen, was die Zuverlässigkeit und Stabilität der Ladeinfrastruktur erhöht. Dies ist entscheidend, um sicherzustellen, dass die EV-LKW-Flotte betriebsbereit bleibt.
- **Zukunftssichere, skalierbare Lösung:** Die erweiterbare Architektur ermöglicht die Integration neuer Anlagen, komplexerer Steuerungsstrategien und die, für Phase dieses Projekts geplante, Teilnahme an Energiemärkten , was sie zu einer wertvollen langfristigen Investition macht.

5 Fazit und nächste Schritte

Phase 1 des Projekts „EV Truck Charging“ hat erfolgreich eine grundlegende Anwendung zur On-Site-Energieoptimierung geliefert. Das Projekt hat alle seine Hauptziele erreicht, einschließlich der Integration von On-Site-Sensoren, der Implementierung eines Echtzeit-Edge-Steuerungssystems und

der Bereitstellung eines Live-Visualisierungs-Dashboards. Die resultierende Anwendung ist robust, erweiterbar und bietet durch die Verbesserung der Effizienz und die Senkung der Kosten einen sofortigen geschäftlichen Mehrwert.

Die folgenden nächsten Schritte werden für zukünftige Projektphasen empfohlen:

- **Verbesserung der Steuerungsstrategien:** Implementierung anspruchsvollerer intelligenter Steuerungsalgorithmen, die Faktoren wie Energiepreise, Demand-Response-Signale und Wettervorhersagen berücksichtigen können, um den Energieverbrauch weiter zu optimieren.
- **Bereitstellung und Feldtests:** Einsatz der Anwendung in einer Live-Produktionsumgebung, um reale Leistungsdaten zu sammeln und die Steuerungslogik basierend auf betrieblichen Erfahrungen zu verfeinern.

Der erfolgreiche Abschluss der Phase 1 hat eine starke Grundlage für die weitere Entwicklung und den Einsatz dieser innovativen Energiemanagementlösung gelegt.