Projekt *Smart Energy*

HB Solutions GbR

27. April 2021

1 Einführung

In diesem Dokument werden die Anforderungen für das Projekt *Smart Energy*, in Auftrag gegeben durch die Rüdensche Windenergie AG (RWE) und durchgeführt von der HB Solutions GbR, aufgestellt. *Smart Energy* sieht vor, eine zentrale Kraftwerkssteuerung für Versorger und Verbraucher in einem Stromnetz zu erstellen. Die Versorger reichen von einer Vielzahl an unterschiedlichen Kraftwerken wie etwa für Solar-, Wind- und Biomasseenergie. Der Kunde RWE bedient sowohl Haushalte als auch Unternehmen als Verbraucher. Stromspeicher sieht der Kunde nicht in der Kraftwerkssteuerung vor. Die Kraftwerke werden zunächst simuliert, damit die Software zunächst vollständig fertiggestellt und getestet werden kann, bevor sie in einer Realumgebung eingesetzt wird.

2 Anforderungsanalyse

Das Projekt setzt sich aus mehreren Komponenten zusammen. Dabei steht im Zentrum die Kraftwerkssteuerung. Weitere Komponenten sind Erzeuger und Verbraucher. Ferner können weitere Komponenten wie Browser oder Energieversorger über definierte Schnittstellen mit dem System interagieren (z.B. historische Daten abfragen).

2.1 Kraftwerkssteuerung

Die zentrale Kraftwerkssteuerung koordiniert Erzeuger und Verbraucher. Zusätzlich soll die zentrale Kraftwerkssteuerung in der Lage sein, per TCP/HTTP mit jedem beliebigen Webbrowser zu kommunizieren. Dafür stellt die Kraftwerkssteuerung einen HTTP-Server mit einer REST-API zur Verfügung, die mindestens `GET`-Anfragen unterstützt. Über diese REST-API können dann externe Komponenten wie ein Webbrowser auf die Steuerung zugreifen. Alternativ dazu, soll die Kraftwerkssteuerung selbst auch per RPC-Schnittstelle konfigurierbar sein, die externe Clients wie Energieversorger nutzen können. Informationen der Erzeuger und Verbraucher werden sowohl vorerst per UDP, als auch später per MQTT empfangen.

2.2 Erzeuger / Verbraucher

Die Erzeuger und Verbraucher teilen ihre Informationen mit der Kraftwerkssteuerung vorerst per UDP, im späteren Verlauf des Projekts aber auch via MQTT. Die Konfiguration der Komponenten durch die Kraftwerkssteuerung soll jedoch per RPC geschehen. Dabei ist darauf zu achten, dass Versorger und Verbraucher klar identifizierbar sind, die Art des Teilnehmers bekannt ist, der Stromverbrauch/-erzeugung in kW angegeben ist und er per UDP mit der Zentralen Kraftwerkssteuerung kommuniziert. Der Kunde sieht für die Simulation mindestens vier Verbraucher/Erzeuger vor, darunter mindestens ein Verbraucher. Zum verbesserten Testen soll der Verbrauch bzw. die erzeugte Menge künstlich variieren können.

2.3 Funktionale Anforderungen

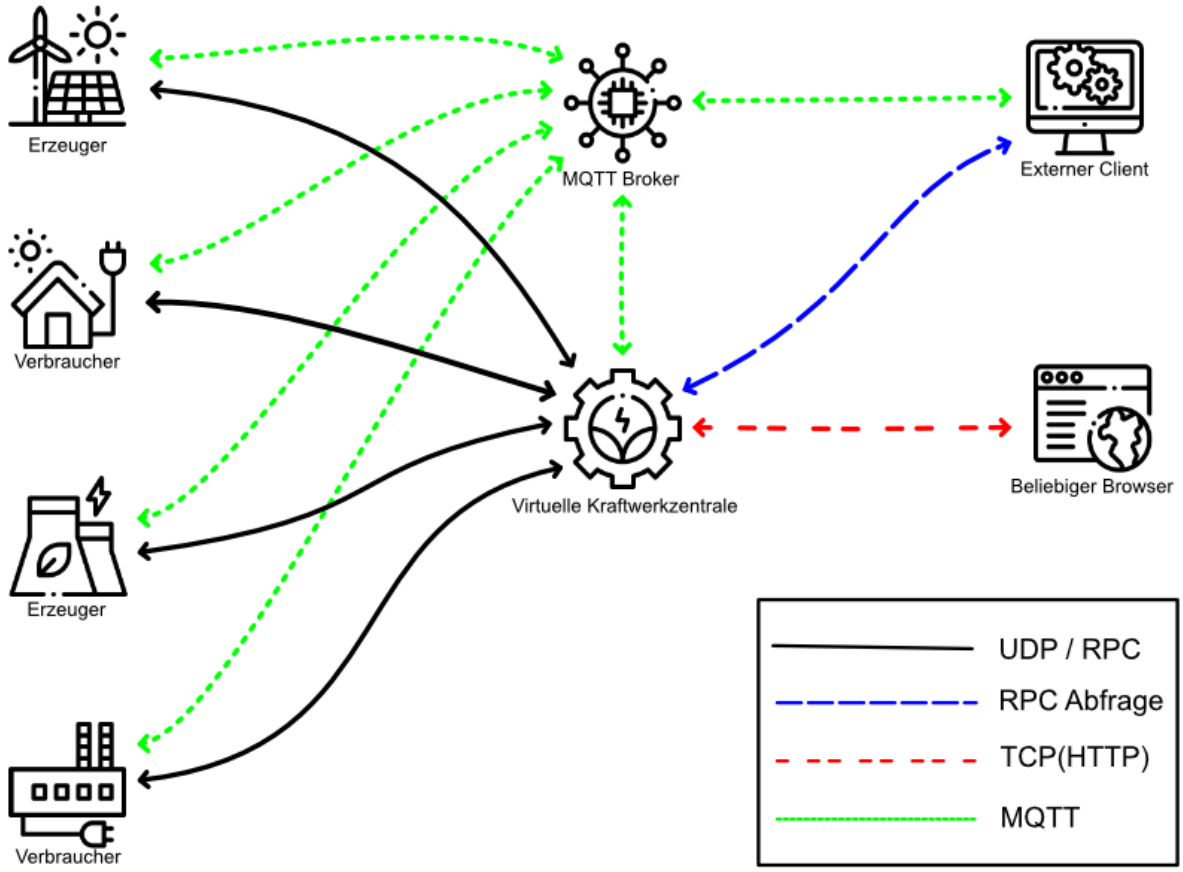
1. Software für Energieversorger sowie -verbraucher erstellen, die folgende Informationen übermitteln:
    a) Art des Teilnehmers
    b) Eindeutige ID oder Name zur klaren Identifikation
    c) Aktuelle Strommenge in kW (variiert je nach Versorger und Verbraucher)
    d) Kommunikation über UDP und später MQTT
    e) Steuerung über RPC
2. Es werden mind 4. unabhängige Erzeuger/Verbraucher erstellt (davon mind. 1 Verbraucher)
3. Software für Zentrale Kraftwerkssteuerung (ZK) erstellen mit folgenden Anforderungen:

    a) Horizontal skalierbar
    b) UPD Schnittstelle (später MQTT) für Verbraucher/Erzeuger
    c) REST-Schnittstelle, die mindestens `GET`-Anfragen verarbeiten kann
    d) Eigene HTTP-Server-Implementierung
    e) RPC-Schnittstelle für externen Client implementieren
4. Software für externen Client erstellen
    a) Kann RPC-Schnittstelle von ZK nutzen

2.4 Nicht-funktionale Anforderungen

1. Programmiersprache ist C++
2. Gitlab CI (Docker und Docker-compose)
3. Clean Code
4. Hygiene des Git-Repositories
5. Dokumentation
6. Lizenzen
7. `Dockerfile` und `docker-compose.yml` Beispiele
8. HTTP selbst implementiert

3 Systemdesign



4 Testing

4.1 Funktionale Tests

- Können UDP Packete von jedem Erzeuger zur zentralen Kraftwerkssteuerung gesendet werden?
- Funktioniert die RPC Verbindung zwischen Verbraucher/Erzeuger und Kraftwerkssteuerung?
- Kann die Kraftwerkssteuerung von Chromium und Firefox erreicht werden?
- Können Erzeuger/Verbraucher eindeutig erkannt werden?
- Ist der aktuelle Stromverbrauch/-erzeugung variierbar?
- Kann der Client per RPC die Kraftwerkssteuerung konfigurieren?

4.2 Nicht-Funktionale Tests

- Ist das Prgramm in C++ geschrieben
- Ist eine aktuelle Version des Projekts bei Abschluss auf dem Masterbranch?
- Lässt sich das Projekt mit einem Befehl deployen?
- Wird das Projekt über eine Pipeline gebaut?
- Ist das Projekt dokumentiert?
- Ist HTTP selbst implementiert?
- Liegt ein `Dockerfile` vor?
- Liegt dem Projekt eine Lizenz bei?
- Wurden Coding Best-Practices eingehalten?

4.3 Performance-Tests

- Verarbeitungsdauer von eintreffenden UDP-Paketen
- Belastungsgrenze UDP-Pakete pro Sekunde
- Belastungsgrenze MQTT-Nachrichten pro Sekunde
- Maximale Anzahl an verbundenen Verbraucher und Erzeugern über RPC

5 Projektplan

Datum	**Meilenstein**
Praktikum 1	Planung des Projekts
Praktikum 2	Kraftwerkssteuerung + Webserver Verbraucher/Erzeuger und funktionierende Kommunikation zwischen Komponenten
Praktikum 3	Einfügen von RPC Schnittstellen Testen der RPC Schnittstellen von Seiten Client und Kraftwerkssteuerung
Praktikum 4	Einführen eines MQTT-Broker Performancetest und Vergleich MQTT zu UDP
Praktikum 5	Kraftwerkssteuerung horizontal erweitern
Praktikum 6	ggf. Abschlussprotokoll

6 Deployment

Um das Projekt auszurollen, wird die HB Solutions GbR CI in Gitlab nutzen. In dieser werden Tests automatisch ausgeführt und die Entwickler stellen sicher, dass zu jedem Zeitpunkt ein lauffähiges System existiert.

Lokal, der Simulation dienend, stellt das Projekt eine `docker-compose.yml`-Datei zur Verfügung, um eine Testumgebung aufzusetzen. Für das spätere Deployment in der tatsächlichen Umgebung ist eine Ansible-Role angedacht.