Bridging Rapid Simulation and Real-Time Execution: A Software-in-the-Loop Testing Platform for OCPP-Based EV Charging Station Management Systems

Christian Skafte Beck Clausen [0000-0003-3118-7253], Bo Nørregaard Jørgensen [0000-0001-5678-6602] and Zheng Grace Ma [0000-0002-9134-1032]

SDU Center for Energy Informatics, Maersk Mc-Kinney Moeller Institute, The Faculty of Engineering, University of Southern Denmark, Odense, Denmark
zma@mmmi.sdu.dk

Abstract. Electric vehicle (EV) charging management systems require scalable and standards-compliant coordination across distributed infrastructures. However, current testing approaches often lack the flexibility to support both accelerated development workflows and high-fidelity real-time validation. This paper presents a dual-mode Software-in-the-Loop (SiL) testing platform for validating EV Charging Station Management Systems (CSMSs) that communicate using the Open Charge Point Protocol (OCPP) 2.0.1. The platform enables the same CSMS software to be evaluated across discrete-time and wall-clock-time modes without modification, supporting rapid scenario testing and protocol-accurate verification. A case study involving 126 home charging stations in Denmark evaluates two smart charging strategies under real-world electricity pricing and demand conditions. Results show that discrete-time testing enables rapid performance evaluation, while wall-clock execution maintains protocol fidelity with minimal time drift. The platform offers a robust and reusable method for validating EV charging software and can support future applications involving distributed energy resources (DERs), smart grids, and cyber-physical energy systems.

Keywords: Software-in-the-Loop (SiL) Testing, Charging Station Management System (CSMS), Open Charge Point Protocol (OCPP 2.0.1), Smart Charging, Validation and Verification (V&V), Simulation, Electric Vehicle (EV) Charging, Energy Management System (EMS), Smart Grid, Distributed Energy Resources (DER)

1 Introduction

The electrification of road transport is widely seen as a critical enabler for reducing greenhouse gas emissions [1]. However, this transition poses significant challenges for system operators and market actors alike, as large-scale integration of EVs increases the complexity and variability of both local and system-wide electricity demand. In this context, management systems for EV charging are essential for aligning flexible charging behavior with user preferences, grid constraints, and market signals.

The deployment of these systems increasingly depends on standardized communication protocols to ensure interoperability across hardware and software vendors. One such protocol, the Open Charge Point Protocol (OCPP), has emerged as a de facto standard for managing interactions between charge stations (CSs) and CS management systems (CSMSs). The recent OCPP 2.0.1 specification [2] introduces enhanced capabilities for smart charging, security, and extensibility, but also increases complexity in terms of control logic, timing, and asynchronous communication.

To ensure that such systems behave reliably under diverse and dynamic operating conditions, Software-in-the-loop (SiL) testing has gained traction as a method for validating real-world control software against simulated environments. SiL allows developers to execute and evaluate software logic prior to deployment, bridging the gap between purely analytical validation and hardware testing [3]. However, the majority of SiL approaches in the energy domain tend to either operate exclusively in real time, making them unfit for iterative development workflows with rapid feedback or for validating communication-centric applications using real-world application protocols like OCPP.

In particular, support for both discrete (accelerated execution) and real-time (wall-clock execution) domains while maintaining application protocol fidelity is not yet widely supported across existing platforms. Moreover, previous SiL testing studies often simplify the communication layer by using non-standard application protocols. However, such approximations are inadequate for reusing the same management software in a more refined deployment environment for verification purposes.

According to IEEE Standard 1012-2024 [4], validation concerns whether the right system is being built, while verification ensures it is built correctly. The proposed SiL platform addresses both: discrete-time execution enables domain experts to rapidly validate system behavior in simulated environments, whereas wall-clock execution facilitates verification in a refined deployment environment that resembles realistic conditions, particularly communication and timing aspects. This dual perspective aligns with established engineering principles, such as the V-model in automotive development, and extends them to the energy informatics domain.

This paper addresses these limitations by presenting a Software-in-the-loop testing platform for EV CSMSs based on previous work [5]. The platform allows the same CSMS software to operate either against a simulated environment in discrete time or in real time using OCPP 2.0.1 message exchanges. The proposed approach enables rapid scenario testing during development and high-fidelity testing during verification, ensuring that systems perform consistently across execution modes. The platform is evaluated through a case study based on Danish household charging, incorporating real driving patterns, market prices, and charging strategies.

2 Literature Review

In-the-loop paradigms are widely used to verify control logic across abstraction levels in complex systems [3]. These paradigms include Model-in-the-Loop (MiL), Software-in-the-Loop (SiL), Processor-in-the-Loop (PiL), and Hardware-in-the-Loop (HiL), each supporting different phases of system development. MiL is primarily used in

early-stage validation where the controller and plant models are both simulated, often using tools like MATLAB/Simulink or Modelica. PiL executes compiled controller code on target processors to evaluate timing behavior. HiL integrates physical components with simulated environments to test hardware-software integration under real-time constraints. While HiL is critical for final-stage testing, it introduces significant cost and complexity.

SiL, in contrast, offers a balance between fidelity and flexibility. It executes real-world control logic against a simulation model and supports closed-loop evaluation without physical hardware. SiL has been widely applied in domains such as automotive control [6, 7], robotics [8], and renewable energy systems [9]. In energy applications, SiL supports testing of control strategies for Volt/Var regulation [10], dispatch algorithms [11], and solar thermal systems [12]. Real-time digital simulators like RTDS and platforms such as TRNSYS and Dymola are often used to represent the virtual plant, while control logic is implemented in environments like MATLAB or AMESIM [13].

Despite its potential, SiL remains underutilized in the energy sector. Most studies examine single-controller, single-model scenarios, limiting insights into how SiL might scale to m-to-n configurations such as energy management systems coordinating multiple DERs. Moreover, interoperability issues persist in SiL testing. Components such as controllers and simulation models are often built using different software tools and communication protocols like TCP/IP or OPC UA [13]. These differences in communication methods, data formats, and software environments make it difficult to connect systems and ensure valid operation within a unified simulation setup prior to verification.

Accelerated SiL testing is another area with limited methodological coverage. To enable long-duration scenario analysis, such as annual grid operation or EV charging patterns, SiL simulations must execute faster than real time. In this paper, a scenario refers to a predefined combination of operating conditions, input data, and objectives under which the system under test is evaluated (e.g., charging demand profiles, electricity price signals, or grid constraints). Scenario-based testing thus means assessing how the software behaves across such defined conditions. Some studies employ discrete-time simulations [14, 15], yet there is little reported work on hybrid or temporally adaptable designs that bridge discrete and real-time execution while preserving application layer protocol fidelity. This is particularly limiting in domains where management software must operate under both offline simulation and real-time deployment constraints.

The literature also lacks a standardized framework for implementing SiL tests. Most contributions focus on application-specific outcomes, e.g., improved energy dispatch or controller tuning, without offering generalizable methods, architectures, validation or verification procedures. The absence of such a framework impedes broader adoption and prevents SiL from being leveraged as a testing approach in future digital energy systems.

Addressing these limitations requires new approaches to software design that accommodate both accelerated and real-time execution, standardized interfaces to improve interoperability, and methodological guidance for multi-agent, distributed, or modular SiL testing in energy contexts. These developments are essential to support

rigorous, scalable validation of management software prior to deployment in critical energy infrastructure.

3 Methodology

3.1 SiL Testing Framework

This paper applies the SiL testing framework proposed by Clausen [5]. Within this framework, two interdependent components form a closed feedback loop: the energy management system (EMS) and a simulation model. In a given test scenario, the EMS retrieves data from the simulation model and applies its decision-making logic to simulated entities. During test execution, key performance indicators (KPIs) are collected from both the EMS and the simulation model to assess the collective system performance.

The framework emphasizes scenario-based testing, where predefined scenarios define the scope and structure of each test case. Tests are executed with the actual EMS software and simulation components running in a closed-loop environment. Central to the methodology is the simulation-based setup, where feedback between the EMS and simulated environment replicates real-world interactions.

The framework supports both integration and system testing. Test scenarios evaluate the behavior arising from EMS integration with the simulation model. Depending on scenario complexity, tests may range from isolated functional checks to large-scale system evaluations spanning hours, days, or weeks. The approach follows grey-box testing principles, requiring insight into both external requirements (e.g., use cases) and internal architecture (e.g., timing, coordination logic, and communication protocols).

Validation and verification are both supported. During validation, the EMS is tested against functional requirements in a rapid, discrete time environment. Verification then ensures that further refinements continue to meet expectations in a real-time (wall-clock) setting. Outputs from the validation stage act as a test oracle and baseline in the verification stage.

3.2 SiL Testing Execution Modes

To validate the behavior of CSMS software before deployment, this study employs back-to-back testing, utilizing two distinct execution modes to compare and ensure consistency: (1) discrete time execution for rapid testing and (2) wall-clock-time execution to mimic real-world deployment scenarios. The aim is to validate the CSMS by analyzing its behavior across these two configurations.

- *Discrete Time Execution.* In this mode, the SiL testing environment operates in discrete time using discrete-event simulation. The CSMS is instantiated within the simulation runtime itself, sharing a single environment (i.e. the Java Virtual Machine). This setup facilitates in-memory communication between the simulation model and the CSMS via OCPP 2.0.1 messages while eliminating the need for a complete network stack. By advancing the simulation clock deterministically and independently of wall-clock time, this approach enables

rapid testing under a wide range of conditions. It allows developers to evaluate the scalability of their algorithms with potentially thousands of simulated charging stations while focusing on application protocol correctness, message handling, and algorithmic behavior.

- *Wall-Clock-Time Execution.* In this mode, the SiL testing environment simulates real-world operating conditions by executing the CSMS as a standalone system, separated from the simulation model. Communication occurs over a network stack using OCPP 2.0.1, introducing realistic conditions such as concurrency and asynchronous communication. The simulation runs in wall-clock time, synchronized with system time, ensuring interactions mimic those of a real deployed CSMS. This configuration enables evaluation of the CSMS's performance, reliability, and robustness in handling real-world network conditions.

The results from both configurations are compared as part of the back-to-back testing process to ensure consistency and correctness. Discrepancies between the two modes can reveal issues such as timing-related errors, protocol mismatches, or performance bottlenecks, helping evaluate the system's readiness for deployment.

3.3 SiL Testing Platform Development

This paper applies the aforementioned SiL testing framework by developing a platform to test a case study for EV smart charging strategies for residential home charging. The aim of this platform is to demonstrate the utility of the SiL testing framework within an energy ecosystem characterized by complex topologies and emergent behaviors, where actors compete for access to a shared resource, namely the power grid.

The platform leverages an existing agent-based simulation model [16, 17] developed in AnyLogic [18] which is connected to a management system using the OCPP 2.0.1 standard. The platform is described in more detail in the original case study [5]; in this paper, it is presented in a condensed format.

4 Case Study

The case study involves EV smart charging for a residential area in Denmark. Smart charging for EVs is an advanced approach to managing EV charging in a way that optimizes energy use, enhances grid stability, and aligns the driver's charging preferences. At its core, smart charging dynamically adjusts the charging rate of EVs based on real-time grid conditions, electricity prices, and renewable energy availability to reduce both costs and CO2 emissions. This flexibility enables efficient load management, reducing loads on the power grid during peak demand periods through techniques like peak shaving and demand response.

Denmark currently has 402.000 registered EVs (April 2025) [19] which corresponds to about 17% of all domestic registered vehicles. With the current adoption rate the number is expected to grow to 1 million EVs by 2030 [20]. EVs require installation of charging infrastructure, and the number of charging stations (CSs) is expected to grow proportionally to the number of EVs. A CS, also commonly known as a charge point,

is a power supply device that supplies electrical power for recharging plug-in electric vehicles. The statistics of public available CSs are provided by the Danish Ministry of Transport who reported around 11,000 public CSs in early 2023. A year later, primo 2024, the number almost doubled to 21,000 public CSs. By the end of 2024 the number exceeded 31,000 CSs public [21]. Approximately 69 pct. of public CSs are normal chargers (≤ 22 kW), 18 pct. are fast chargers (23-99 kW), and 13 pct. are rapid chargers (≥ 100 kW). There are no official statistics on the number of installed private chargers because Denmark has no legislation that requires it to be registered. This number can be roughly estimated with the number of registered EVs. Assuming that 20–35% of EV owners have a privately owned CS, the estimated number ranges between 80,400 and 140,700 CSs.

This estimated growth threatens the future power grid stability which is solved partly by expanding the grid's capacity and by utilizing the grid's current capacity more intelligently. Intelligent coordination in smart charging algorithms utilizes the EV owner's charging flexibility to allocate power grid resources in a way that avoids peak loads.

The behavior of smart charging algorithms is critical to test because dysfunctional behavior can affect power grid customers. A SiL testing platform based on the SiL testing framework is developed to verify smart charging and system behavior in realistic scenarios to ensure correct behavior prior to hardware- and field testing.

5 Platform Design and Implementation

5.1 Requirements

The following requirements were derived through experimental prototyping and iterative development.

- **R1. Temporal Modality Support.** The platform must support both discrete-time and wall-clock time execution modes, enabling early-stage validation with rapid feedback cycles and later-state verification under realistic temporal constraints.
- **R2. Interoperability.** The platform must support standard application protocols such as OCPP 2.0.1 to ensure realistic message exchange and compatibility with industry standards. This allows independently developed CSMSs to coexist within a shared SiL testing environment, enabling holistic system-level testing scenarios.
- **R3. Abstracted Communication Layer.** The platform must abstract parts of the communication stack below the application protocol (e.g., transfer and transport protocols). This enables validation at the application protocol level without requiring the entire network stack, which reduces performance overhead significantly.
- **R4. Observability and Traceability.** The platform must expose KPIs, for example, protocol message logs and time synchronization metrics, to validate and verify a given test scenario.

5.2 Architecture

The SiL testing platform consists of the following principal components as shown in Fig. 1. Each component is described below the figure.

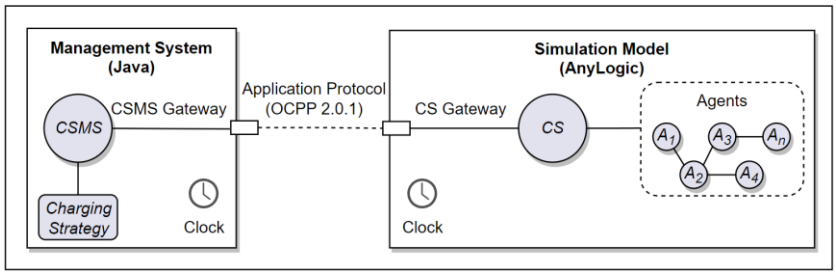

Fig. 1. Architecture of the SiL testing platform.

Simulation Model. Developing a simulation model requires significant efforts, and therefore, this study reuses a previously validated simulation model. This model was developed by the author's colleagues Kristoffer Christensen and Magnus Værbak [16, 17]. The simulation model contains logic for charging stations (CSs), electric vehicles (EVs), households, utilities, and market actors. It is implemented as an agent-based discrete event simulation in AnyLogic 8.8.3, based on a combined and validated model of a Danish residential district with 126 households equipped with home charging stations. Key features include various agent types (e.g., EVs, CSs, transformers), configurable parameters (e.g., scenario duration, EV types, PVs, load profiles), and realistic behavior based on empirical data (e.g., EV arrival patterns, charging preferences). The model supports both traditional and real-time pricing strategies and enables execution of short- and long-term test scenarios in discrete time. Output data includes KPIs such as load profiles and charging behavior, are exported in 10-minute resolution for post-analysis to satisfy **R4** (Observability and Traceability). The model supports external Java libraries by importing JAR files, enabling the model to integrate with CSMSs via OCPP 2.0.1.

Communication Layer (OCPP 2.0.1). A custom communication layer was implemented to enable SiL testing using the OCPP 2.0.1 protocol. It is developed in Java 11 for compatibility with AnyLogic 8.8.3 and facilitates interoperability between CSMSs and the simulation model. The design decouples application logic from the underlying transport (e.g., WebSocket over TCP/IP) via an abstracted communication API. This abstraction allows test scenarios to be executed with or without a full network stack for validation or verification purposes.

The communication layer comprises four packages: (i) `schemas`, which provides auto-generated Java classes from the official OCPP 2.0.1 JSON schemas; (ii) `rpc`, which defines `CALL`, `CALLRESULT`, and `CALLERROR` message types with JSON serialization support for generic payloads; (iii) `api`, which defines high-level interfaces (e.g., `CsmsEndpoint`, `CsEndpoint`, and `Session`) for implementing CSMS and CS application logic; and (iv) `impl`, which contains the logic to expose endpoints by establishing bidirectional gateways for message exchange over the underlying transport. In practice, the connection between the simulation model in AnyLogic and

the CSMS software is realized by importing the compiled communication layer as JAR libraries into the AnyLogic model, allowing simulation agents to call the provided Java interfaces directly. Charging station agents implement the `CsEndpoint` interface, while the CSMS implements the `CsmsEndpoint` interface. At runtime, a gateway agent binds these interfaces, enabling OCPP 2.0.1 messages to flow bidirectionally between simulated CSs and the CSMS either in-memory (discrete-time mode) or over the network (wall-clock mode). This design satisfies requirement **R2** (Interoperability) and **R3** (Abstracted Communication Layer).

Time Management and Scheduling. To support SiL testing across both discrete and wall-clock time domains, a time management and scheduling library was developed in Java 11. The library introduces a temporal abstraction via a `TimeProvider` interface with two implementations: `DiscreteTimeProvider` and `WallClockTimeProvider`. This abstraction allows the same application logic in the CSMS to operate without modification in different temporal modes, supporting both rapid- and real-time execution. A custom scheduler complements the temporal abstraction by mode-aware task execution. The scheduler is single-threaded and utilizes a priority queue to ensure time-ordered and FIFO execution. Concretely, the scheduler is responsible for all time-dependent tasks in the CSMS, including OCPP protocol timers (e.g., timeouts), scheduled charging events, and state-machine transitions. Tasks are executed strictly in increasing timestamp order, with FIFO behavior in case tasks have the same timestamp. In discrete-time mode, the scheduler advances according to the simulation clock, ensuring deterministic and reproducible results independent of wall-clock speed. In wall-clock mode, the scheduler synchronizes with the system clock, which in turn is synchronized to wall-clock time via the Network Time Protocol (NTP). By providing a scheduling abstraction, the library satisfies requirement **R1** (Temporal Modality Support).

Charging Station Management System. The CSMS is implemented in Java 11 and integrates application logic, communication endpoints, scheduling, and state management to support OCPP 2.0.1-based scenarios. It implements the `CsmsEndpoint` interface and manages per-session state using a structured map of session data objects. Communication with each CS is facilitated through proxy objects of the `CsEndpoint` interface. Charging strategies (e.g., traditional and real-time pricing) are encapsulated and can be configured depending on a given scenario.

The implementation uses a pluggable scheduler that operates in either discrete or wall-clock time, depending on the test mode. Stateful protocol interactions are modeled using statecharts, enabling management of transactional OCPP use cases such as Smart Charging (K15). Timed transitions are implemented via scheduled events, ensuring correct protocol sequencing and timeouts.

Charging Stations: Each CS is implemented as part of the AnyLogic 8.8.3 simulation model and utilizes the OCPP 2.0.1 communication layer. Each CS implements the `CsEndpoint` interface to handle incoming messages from its CSMS. Session bindings are managed by a gateway agent, which maintains mappings between internal

charging station agents and their corresponding `CsmsEndpoint` message forwarder proxy instances.

5.3 Hardware and Deployment

The SiL testing platform is deployed on server-grade hardware and consists of the following hardware components: ***CPU:*** Intel Xeon Silver 4214, 12 cores, 24 threads, 2.20 GHz base clock, 3.20 GHz turbo ***RAM:*** 96 GB DDR4, 2400 MHz, ECC. ***Hard Drive:*** 128 GB, 13.4 MB/s (4K random write), 423 MB/s (1 MB sequential write), 498 MB/s (1 GB sequential write). Read speeds are not reported as they are insignificant for the application, which primarily performs write operations. ***Network:*** 1 Gbit/s full duplex switched network. ***OS:*** Ubuntu 20.04.1 LTS. Deployment depends on the execution mode. In discrete-time execution mode, each CSMS is deployed within the same Java Virtual Machine as the AnyLogic model to exploit in-memory communication with minimal overhead. In wall-clock execution mode, each CSMS and the AnyLogic model are deployed in a distributed environment with separate Docker containers that communicate over a full TCP/IP stack to reflect realistic deployment conditions.

6 Scenario Design

6.1 OCPP 2.0.1 Use Case

The following OCPP 2.0.1 use case is implemented in the SiL testing platform for demonstration purposes.

- *K15 – Charging with load leveling based on High Level Communication (Smart Charging)* [22]: Tests the charging strategy as determined by the CSMS.

6.2 Topology and EV Charging Strategies

The scenario is configured, as shown in Fig. 2, with two CSMSs that operate simultaneously and are configured in a one-to-many relationship with its provisioned CSs. Each CSMS represents a CSO that manages a subset of 63 CSs (126 CSs in total) as indicated by the simulation model configuration in Section 6.3. A CS is limited to charging and connecting with a single EV at a time, thereby simulating a private home charging setup.

Operator A employs a CSMS with a *Traditional Charging Strategy* that provides unrestricted smart charging profiles to its managed CSs. With traditional charging, the EV will charge as soon as it is plugged into the CS and continue charging until the battery is full.

Operator B employs a CSMS with a *Real-time Pricing Strategy* that optimizes the charging profile of each EV according to the preferred departure time, charging parameters, and hourly electricity spot prices. The EV and driver provide the departure time and charging parameters to perform this optimization.

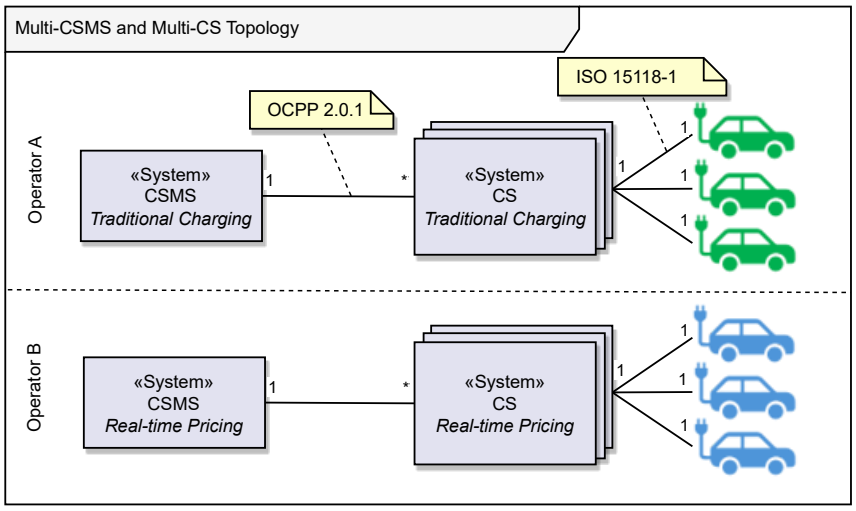

Fig. 2. The SiL testing topology featuring multiple CSMSs and CSs [5].

6.3 Simulation Model Configuration

The simulation model is treated as the tool responsible for simulating the entire environment except for the CSMS. The model parameters and settings are listed in Table 1. For a detailed description of each parameter, e.g. the driving distance model or EV model, the reader is referred to the original work [16].

Table 1. Simulation Model Configuration [5].

Parameter	Setting
Charging infrastructure	Private home charging in Mode 3 [23]
Number of domestic consumers	126
Scenario start date	2024/01/01 00:00:00 UTC +1
Scenario end date	2024/01/08 00:00:00 UTC +1
Scenario duration	7 days
Maximum allowed number of EVs at a domestic consumer	1
Maximum allowed number of CSs at a domestic consumer	1
Domestic consumer baseload data	Embedded in the simulation model's dataset. Based on real consumption data.
Total number of CSs	126. Each consumer has one installed.
Driving distance per trip	Embedded in the simulation model's dataset. Generated based on a probability distribution.
First possible departure hour	05:00
Last possible departure hour	09:00
First possible arrival hour	14:00
Last possible arrival hour	22:00

Parameter	Setting
EV models	Embedded in the simulation model's dataset. This includes battery capacity, charging power, etc.
CS models	Embedded in the simulation model's dataset.
Number of electricity suppliers	1
Number of CS suppliers	1
EV charging strategy	50/50 split of: 63 consumers with Real-time pricing 63 consumers with Traditional charging
Total grid capacity (transformer capacity)	400 kVA
Household capacity	17.3 kVA
CS maximum charging capacity	11 kW (three phases, 400V, 16A)
Electricity costs model	Embedded in the simulation model's dataset. Based on real hourly rates from Nord Pool spot market (DK1) and the Danish Tariff model 3.0.

In discrete time execution mode, the simulation model is responsible for time synchronization with the CSMS. In wall-clock-time execution mode, a custom feedback scheduling mechanism is implemented in AnyLogic to synchronize its virtual clock automatically [5, 24]. The internal feedback scheduling mechanism relies on a Proportional-Integral-Derivative (PID) controller that adjusts the simulation's real-time scale, corresponding to a virtual speeder, based on the model's time drift. A timed event triggers PID updates every 10 ms, balancing responsiveness with CPU load.

7 Results

This section presents the results of the two SiL test scenarios: discrete-time SiL and wall-clock-time SiL. The Simulation KPIs are presented first followed by the exchanged OCPP 2.0.1 messages, and finally the SiL performance metrics.

7.1 Simulation Model KPIs

Fig. 3 illustrates the KPIs for EV charging behavior recorded during the SiL test, spanning from 00:00 on January 1st, 2024, to 00:00 on January 8th, 2024. The data is divided into two groups. The first group represents consumers managed by the CSMS following the traditional charging strategy, while the second group represents consumers managed by the CSMS using the real-time pricing strategy. The raw electricity spot price is included to reflect its impact on the number of EVs charging under the real-time pricing strategy.

- *Traditional charging strategy.* This strategy charges each EV immediately when the driver arrives home and plugs it into the CS during the daily period from 14:00 to 01:00. The strategy charges each EV continuously until its battery is full, without considering the electricity spot price. This consistent charging behavior is reflected in the plot across all days.

- *Real-time pricing strategy.* This strategy calculates the optimal charging schedule for each EV based on the hourly electricity spot prices to minimize total charging costs. It respects the EV driver's expected departure time as a constraint. The plot reveals this behavior through nightly surges in charging activity, typically occurring between 23:00 and 05:00. Charging is scheduled during hours with the lowest prices, leading to temporary pauses and later resumptions when prices drop. This results in cost savings and is depicted in the plot as midnight spikes in charging activity, followed by drops, and then smaller surges during the night and early morning.

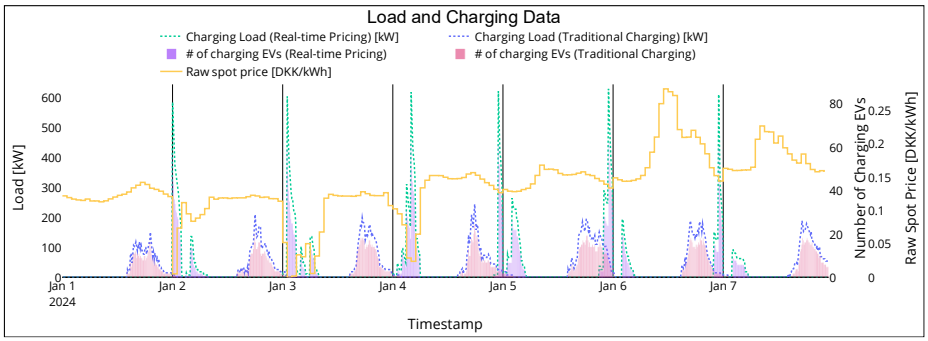

Fig. 3. KPIs for the EV charging behavior in the AnyLogic simulation model [5].

The two datasets were compared, and differences were found between the two datasets:

- *Time-dependent KPIs.* This analysis includes output variables calculated by the simulation model as functions of time, such as the EV's state-of-charge. The resulting floating-point values showed insignificant discrepancies within the range $[0,\ 10^{-4}]$. These discrepancies arise from time differences inherent in wall-clock SiL testing, where measurements are not perfectly synchronized, unlike in discrete-time SiL testing. However, all remaining time-independent KPIs were identical.
- *Number of Charging EVs.* This KPI measures the number of EVs charging at a given time. On January 5th, 2024, at 15:50, the wall-clock SiL test recorded one additional charging EV compared to the discrete-time SiL test. After analyzing this discrepancy with a simulation model developer, it was concluded that the EV state-of-charge was identical in both tests. This behavior was identified as a bug in the measurement logging approach within the simulation model.

In summary, the simulation KPIs from both discrete-time and wall-clock time SiL tests exhibit similar EV charging behavior.

7.2 OCPP 2.0.1 Message Exchange

This section presents the results of the OCPP 2.0.1 messages exchanged during the wall-clock time SiL test. In wall-clock execution, communication occurs over a full TCP/IP stack, which makes it possible to observe network-related effects such as

transmission delays, message reordering, and potential packet loss. In practice, no message losses were observed under the tested conditions, but the platform can capture them through protocol logs if they occur. A similar analysis was not conducted for the discrete-time SiL test because, in this setup, OCPP 2.0.1 messages are exchanged directly in memory within the simulation environment, bypassing the TCP/IP stack entirely.

Use Case: K15 – Smart Charging. Every EV in the simulation model is configured to charge once a day. Once an EV is plugged in a CS the use case "*K15*" is activated, and the CSMS sends a charging profile to the CS. Therefore, the expected number of exchanged charging profiles ($N_{chargingprofiles}$) is given by:

$$N_{chargingprofiles} = c \times d = 126 \times 7 = 882 \quad (1)$$

where:

- c = number of CSs,
- d = number of days.

The collected data confirms that the expected number of `SetChargingProfile-Request` and `SetChargingProfileResponse` were successfully exchanged during the SiL test. While the raw dataset is not reproduced here, it is documented in detail in [5]. This means that the stateful charging behavior is working as expected, because a `SetChargingProfileRequest` is sent to a CS if, and only if, the state transitions correctly.

Fig. 4 depicts the charging behavior of a sample CS managed by the Real-time Pricing CSMS. The transaction state alternates between '*Started*' and '*Ended*', determined by the payload of the `TransactionEventRequest` sent from the CS to the CSMS when the charging cable is connected. The figure demonstrates that the CS charges the EV multiple times within the same transaction, as indicated by fluctuations in the charging load. The transaction state remains as '*Started*' because the SiL test ended at midnight between January 7th and January 8th.

Fig. 5 illustrates the charging behavior of a sample CS managed by the Traditional Charging CSMS. As with Real-time Pricing, the transaction state alternates between 'Started' and 'Ended' based on the `TransactionEventRequest` payload. In this case, the CS begins charging the EV immediately upon being plugged in by the driver and stops when the EV's battery is fully charged. Consequently, the charging load aligns directly with the '*Started*' and '*Ended*' transaction events, contrasting with the behavior observed under Real-time Pricing.

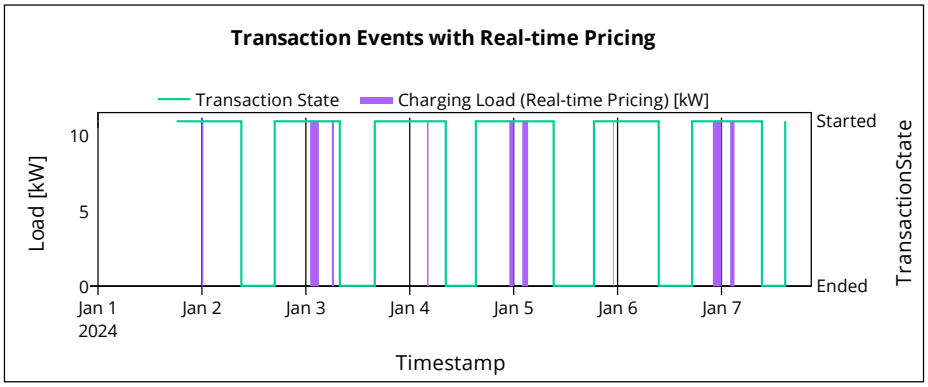

Fig. 4. Transaction events with the Real-Time Pricing strategy [5].

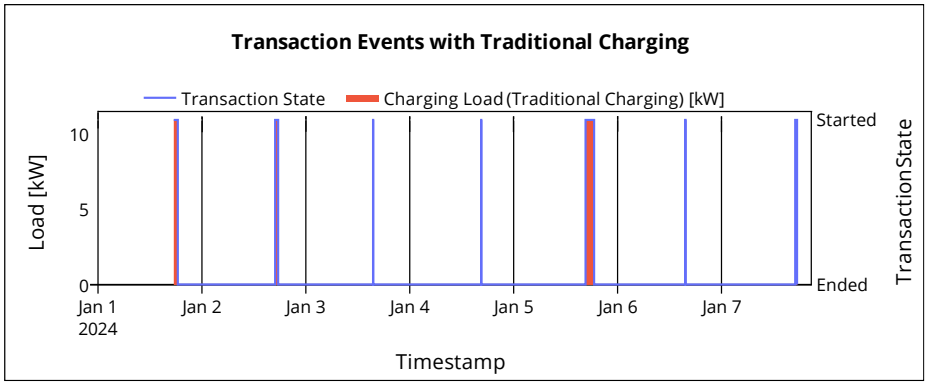

Fig. 5. Transaction events with the Traditional Charging strategy [5].

7.3 SiL Execution Metrics

This section contains the results and analysis for the following SiL performance metrics:

- *Simulation model time drift and real-time scale over time.* These metrics evaluate the performance of the PID controller agent described in Section 6.3. and its robustness in synchronizing the simulation model with wall-clock time.
- *Execution times of the discrete-time and wall-clock-time SiL tests.* These measurements provide a comparative assessment of the two SiL testing approaches.

Simulation Model Time Drift and Real-time Scale Over Time. The AnyLogic simulation model synchronizes wall-clock time using a PID controller to ensure consistent timing behavior during execution. Fig. 6 illustrates the PID controller's behavior during the startup phase of the simulation. Ideally, the time drift (T_{drift}) is 0, indicating perfect synchronization between the simulation model and wall-clock time. If $T_{drift} < 0$, the simulation model lags behind wall-clock time, whereas $T_{drift} > 0$ indicates the

simulation model running ahead of it. At AnyLogic's startup, T_{drift} initially drops below -30,000 ms, even though the real-time scale (RT_{scale}) is set to a maximum value of 10 times wall-clock time. This temporary desynchronization occurs due to the initialization of internal agents and AnyLogic's single-threaded computational model. After this initialization phase, T_{drift} approaches 0. By approximately 33 seconds, the PID controller adjusts RT_{scale} to compensate for the drift, stabilizing T_{drift} at 0 and achieving synchronization.

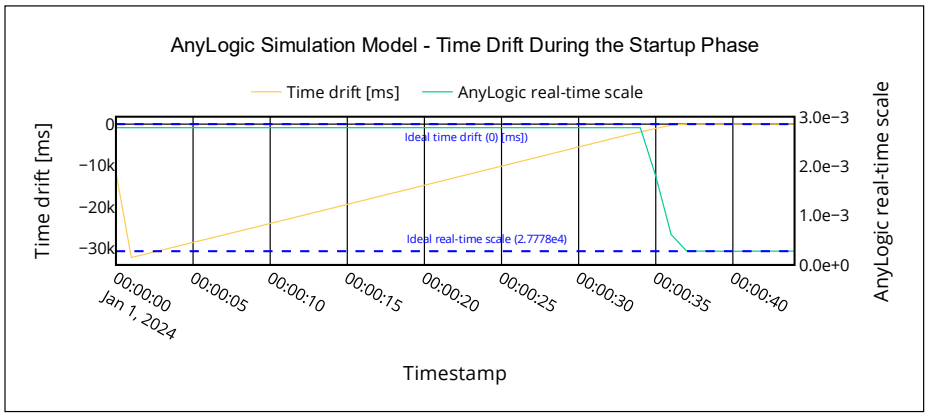

Fig. 6. AnyLogic Simulation Model: Time-drift during the startup phase [5].

Fig. 7 depicts the behavior of T_{drift} and RT_{scale} throughout the duration of the wall-clock SiL test. Frequent positive and negative spikes in T_{drift} are observed, with the PID controller dynamically adjusting RT_{scale} to maintain synchronization between the AnyLogic simulation model and wall-clock time. Despite significant negative spikes in T_{drift}, the PID controller demonstrates robustness, successfully preserving synchronization. The negative spikes in T_{drift} are likely caused by intensive computations or discrete events within AnyLogic that temporarily delay the execution of other operations. Conversely, positive spikes may result from PID overshooting during corrective adjustments.

Fig. 8 presents a histogram of the T_{drift} distribution, based on 20,049 observations ($N_{T_{drift}} = 20{,}049$). Notably, 97% of the observations fall within the range of ± 3 ms, with the remaining 3% outside this interval. This level of deviation is considered acceptable because the scenario does not have stringent time-critical requirements for the wall-clock SiL test.

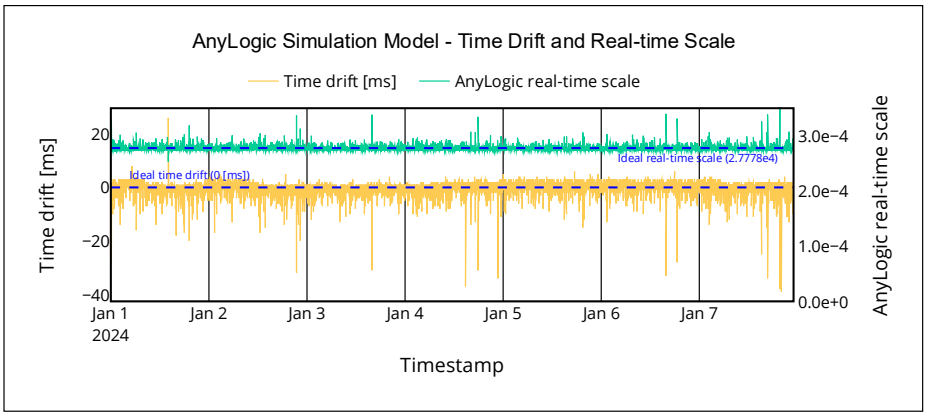

Fig. 7. AnyLogic Simulation Model: Time-drift and real-time scale [5].

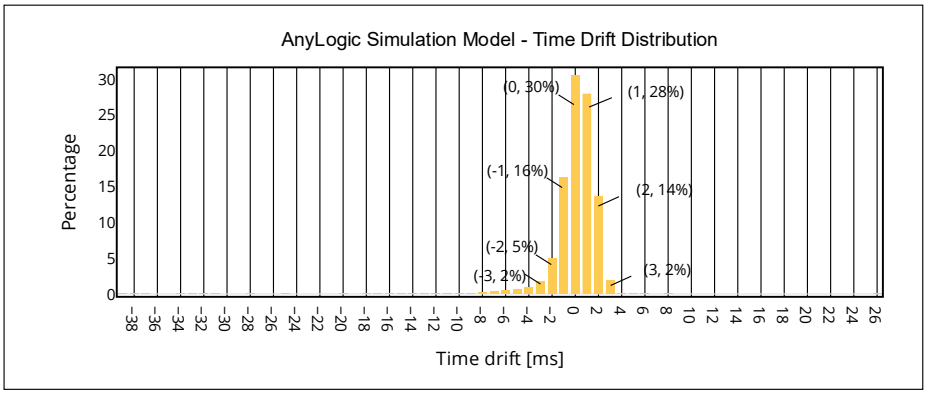

Fig. 8. AnyLogic Simulation Model: Time drift distribution [5].

Execution times of the discrete-time and wall-clock-time SiL tests. Table 2 presents the execution times for the discrete-time and wall-clock-time SiL tests. The discrete-time SiL test completed the 7-day simulation scenario in 1 minute and 49 seconds. In contrast, the wall-clock-time SiL test synchronizes its ticks with wall-clock time and the scenario takes 7 days to execute.

The difference in execution times highlights a trade-off between SiL fidelity and performance. Increasing the fidelity of the SiL test has a substantial impact on execution time. However, the discrete-time SiL test offers significant advantages, as it enables the testing of long-term scenarios in a fraction of the time required by wall-clock SiL tests.

The execution time of the wall-clock-time SiL test could potentially be reduced by employing an accelerated wall-clock time provider from the time management framework outlined in Section 5.2. However, this approach may necessitate tuning the PID controller in the AnyLogic simulation model to accommodate the accelerated mode. Additionally, the acceleration increases the frequency of exchanged OCPP 2.0.1

messages, potentially leading to network congestion. Therefore, this approach should be investigated in future studies.

Table 2. Execution times of the SiL tests [5].

Test Scenario	Execution time
Discrete time	T_{start} = 2024/11/26 15:09:29 T_{end} = 2024/11/26 15:11:18 $T_{total} = T_{end} - T_{start}$ = 1m 49s
Wall-clock time	T_{start} = 2024/11/26 17:55:16 T_{end} = 2024/12/03 17:55:19 $T_{total} = T_{end} - T_{start}$ = 7d 0h 0m 3s

8 Discussion

The presented SiL testing platform successfully addresses several limitations identified in the literature on validation and verification of EV CSMSs, particularly those relying on the OCPP 2.0.1 protocol. The platform demonstrates practical utility in enabling scenario-based testing across both discrete-time and wall-clock-time execution modes, a feature absent in most existing SiL approaches in the energy domain.

From a functional perspective, the platform met its design requirements, including support for dual temporal modalities (**R1**), interoperability with standardized protocols (**R2**), abstracted communication layers (**R3**), and observability of system behavior (**R4**). The use of an agent-based simulation environment integrated with real-world data on electricity pricing and EV driving patterns provided a high degree of realism, facilitating meaningful performance evaluation.

In discrete-time mode, the platform enabled rapid testing and iteration of control strategies, completing a 7-day simulation in under 2 minutes. This speed advantage is particularly valuable for development workflows requiring frequent regression testing or optimization across large parameter spaces. Wall-clock-time testing, while significantly slower, offered high-fidelity evaluation of protocol-level behavior under realistic timing and concurrency conditions. The successful back-to-back validation of control logic across both modes confirms the platform's capability to reuse the same CSMS implementation without modification, which is critical for bridging development and deployment workflows.

The message exchange analysis confirmed full protocol compliance, with correct sequencing and successful transmission of OCPP 2.0.1 messages across 126 charging stations. Minor discrepancies in state-of-charge values and charging EV counts between time modes were traced to the simulation model's logging mechanisms rather than flaws in the CSMS or communication layer, underlining the importance of robust KPI instrumentation in simulation models.

While comparable platforms are scarce in academic literature, this work demonstrates that the combination of standardized protocols and dual-time execution in SiL testing is not only feasible but also essential for ensuring that smart charging systems

can scale reliably within real-world energy ecosystems. The findings imply that similar approaches can be generalized to other domains involving DERs and protocol-driven management software, including home energy management and microgrid systems.

9 Conclusion and Future Work

This paper introduced a standards-compliant, dual-mode SiL testing platform for validation and verification of OCPP 2.0.1-based EV CSMSs. The platform enables switching between discrete-time simulation for development and wall-clock-time execution for high-fidelity verification, offering a consistent environment for testing smart charging strategies in a simulated residential setting.

The case study involving 126 private home charging stations in Denmark demonstrated that the same CSMS logic could be tested across temporal domains, achieving protocol integrity, scalability, and timing accuracy. Key performance results include sub-2-minute execution for a 7-day scenario in discrete-time mode and minimal time drift in wall-clock mode, with over 97% of observations within ±3 ms. This hybrid testing approach enables efficient exploration of system behavior under both accelerated and real-time constraints, thereby supporting more robust control software development for EV infrastructure.

The main contribution of this work lies in operationalizing SiL testing with OCPP 2.0.1 compliance, showing that tested CSMS software can be transferred between testing and deployment contexts. The platform also establishes a reusable methodology for future research on SiL testing for energy management systems.

Acknowledgments. This paper is part of the Digital Energy Hub project, funded by the Danish Industry Foundation, and IEA EBC Annex 96 - Grid Integrated Control of Buildings (EUDP no. J.nr. 134251-549133).

Author Contributions. Based on the CRediT (Contributor Roles Taxonomy). **CSBC**: Conceptualization, methodology, software, formal analysis, investigation, writing - original draft, Writing – review & editing, visualization. **BNJ**: Supervision, writing- review and editing, project administration. **ZGM**: Supervision, project administration, funding acquisition.

Conflict of Interests. The authors declare the following potential conflict of interest: BNJ serves as General Co-chair and ZGM serves as Program Co-chair and Publication Chair of the 'Energy Informatics.Academy Conference 2025 (EI.A 2025)'. To avoid any bias, the paper review and acceptance decision were handled independently of these authors' conference roles.

Ethics Approval. Not applicable.

Data and Source Code Availability. Access may be granted upon request, subject to institutional and project-specific policies.

References

[1] (2020). *COMMUNICATION FROM THE COMMISSION TO THE EUROPEAN PARLIAMENT, THE COUNCIL, THE EUROPEAN ECONOMIC AND SOCIAL COMMITTEE AND THE COMMITTEE OF THE REGIONS Powering a climate-neutral economy: An EU Strategy for Energy System Integration*. [Online] Available: https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=COM:2020:299:FIN

[2] *Open Charge Point Protocol (OCPP) 2.0.1*, Open Charge Alliance, 2024. [Online]. Available: https://openchargealliance.org/my-oca/ocpp/

[3] C. S. B. Clausen, B. N. Jørgensen, and Z. G. Ma, "A scoping review of In-the-loop paradigms in the energy sector focusing on software-in-the-loop," *Energy Informatics,* Journal article vol. 7, no. 12, 2/27 2024, doi: 10.1186/s42162-024-00312-8.

[4] *1012-2024 - IEEE Standard for System, Software, and Hardware Verification and Validation*, 2025.

[5] C. S. B. Clausen, "A Software-in-the-loop Testing Framework for Energy Management Systems," ed: Syddansk Universitet. Det Tekniske Fakultet, 2025.

[6] I. Raghupatruni, T. Goeppel, M. Atak, J. Bou, and T. Huber, "Empirical Testing of Automotive Cyber-Physical Systems with Credible Software-in-the-Loop Environments," *International Conference on Connected Vehicles and Expo,* 2019.

[7] M. A. Taut, G. Chindris, and A. C. Taut, "Software-in-the-Loop System for Motor Control Algorithms," *International Symposium for Design and Technology in Electronic Packaging,* pp. 419--426, 2019.

[8] B. A. M, Z. L, and A. M, "Software In the Loop Simulation for Robot Manipulators," *ENGINEERING TECHNOLOGY AND APPLIED SCIENCE RESEARCH,* vol. 7, no. 5, pp. 2017--2021, 2017.

[9] C. A. V. Guerrero *et al.*, "A New Software-in-the-Loop Strategy for Real-Time Testing of a Coordinated Volt/Var Control," pp. 6--10, 2016.

[10] C. A. V. Guerrero, P. M. Silveira, and J. M. C. Filho, "Adaptation of the clonal selection algorithm to the real-time coordinated Volt/VAr control through a software-in-the-loop strategy," *Electric Power Systems Research,* vol. 194, 2021, doi: 10.1016/j.epsr.2021.107092.

[11] F. a. Bonassi, "Software-in-the-loop testing of a distributed optimal scheduling strategy for microgrids' aggregators," *IEEE PES Innovative Smart Grid Technologies Conference Europe,* pp. 985--989, 2020.

[12] M. Huber, C. Bons, and D. Mueller, "Exergetic evaluation of solar controller using Software-In-The-Loop method," *Energy Procedia,* vol. 48, pp. 850--857, 2014, doi: 10.1016/j.egypro.2014.02.098.

[13] H. T. Nguyen, G. Yang, A. H. Nielsen, and P. H. Jensen, "Hardware- and Software-in-the-Loop Simulation for Parameterizing the Model and Control of Synchronous Condensers," *Ieee Transactions on Sustainable Energy,* vol. 10, no. 3, pp. 1593--1602, 2019, doi: 10.1109/TSTE.2019.2913471.

[14] O. Frotscher *et al.*, "Software-in-the-Loop-simulation of a District Heating System as Test Environment for a Sophisticated Operating Software," pp. 223--230, 2019, doi: 10.5220/0007809602230230.

[15] V. Osadcuks and A. Galins, "Software in the Loop Simulation of Autonomous Hybrid Power System of an Agricultural Facility," *Engineering for Rural Development,* pp. 500--505, 2012.

[16] K. Christensen, "Multi-Agent Based Simulation Framework for Evaluating Digital Energy Solutions and Adoption Strategies," Ph.D. thesis, Syddansk Universitet. Det Tekniske Fakultet, 2022.

[17] M. Værbak, "Agent-Based Framework for Simulating Evolution of Distributed Energy Resources in Energy Systems," ed: Syddansk Universitet. Det Tekniske Fakultet, 2022.

[18] A. Borshchev, *The Big Book of Simulation Modeling: Multimethod Modeling with AnyLogic 6*. 2013.

[19] Danmarks Statistik. "Der er nu over 400.000 elbiler på de danske veje." Danmarks Statistik. https://www.dst.dk/da/Statistik/nyheder-analyser-publ/nyt/NytHtml?cid=49510 (accessed July 2025.

[20] Klimarådet, "Kendte veje og nye spor til 70 procents reduktion," 2020/03 2020. [Online]. Available: https://klimaraadet.dk/sites/default/files/imorted-file/70_pct_analyse_endelig.pdf

[21] The Danish Ministry of Transport, "Udvikling i ladeinfrastruktur og bestand af el- og plug-in-hybridbiler," 2024. Accessed: January, 2025. [Online]. Available: https://www.trm.dk/media/xw3dloqy/ladeinfrastruktur-og-opladelige-biler-1-kvartal-2024-a.pdf

[22] Open Charge Alliance, "OCPP 2.0.1 Part 2: Specification (Edition 2)," p. 274, 2023. [Online]. Available: https://openchargealliance.org/my-oca/ocpp/.

[23] *DS/EN IEC 61851-1:2019 Electric Vehicle Conductive Charging System – Part 1: General requirements*, International Electrotechnical Commission, 2017. [Online]. Available: https://webshop.ds.dk/standard/M298491/ds-en-iec-61851-1-2019

[24] L. Sha *et al.*, "Real Time Scheduling Theory: A Historical Perspective," *Real-Time Systems,* vol. 28, pp. 101-155, 11/01 2004, doi: 10.1023/B:TIME.0000045315.61234.1e.