PulseCharge: Enabling Interoperability in EV Data

Ricardo Venâncio[1,2][0009-0009-2697-9898], Clarisse Andreia Sousa[1,2][0009-0001-1793-2328], Pedro Pires[1,2][0009-0006-2384-3897], Tiago Fonseca[1,2][0000-0002-5592-3107], Luís Lino Ferreira[1,2][0000-0002-5976-8853]

[1] Polytechnic of Porto - School of Engineering
[2] INESC-TEC
Porto, Portugal
{ravrf, cassa, pmdrp , calof, llf}@isep.ipp.pt

Abstract: The increasing adoption of Electric Vehicles (EVs) and distributed renewable energy resources poses a growing challenge to power grids, particularly due to uncoordinated and inflexible energy demand. Intelligent energy management strategies, such as smart charging and demand response through energy flexibility optimization, offer a promising solution to mitigate these impacts, but their effectiveness depends heavily on access to State of Charge (SoC) data from EVs. While advanced charging infrastructure, such as DC (Direct Current) fast chargers, can often retrieve SoC data directly from the vehicle, many residential and small-scale AC (Alternating Current) chargers, where the greatest flexibility potential lies, do not provide this information. This lack of standardized, reliable SoC visibility severely limits the deployment of intelligent control systems in real-world settings. This paper presents PulseCharge, a system designed to address this gap by providing brand-agnostic, cloud-based middleware that enables SoC data acquisition across heterogeneous EVs. Designed with scalability, privacy, and interoperability in mind, PulseCharge supports multiple integration levels, from basic monitoring to full user-centric flexibility control, and interfaces with broader systems via standardized Pub/Sub messaging. By unlocking essential EV data at the household level, PulseCharge enables intelligent energy management to be deployed where it matters most.

Keywords: EV, REC, SoC, Middleware, Cloud.

1 Introduction

The transition to Electric Vehicles (EVs) and the growing integration of distributed renewable energy sources present several challenges, including the overloading of power grids, which are not prepared to handle the increased demand caused by simultaneous charging multiple EV. One possible solution to address this problem is to upgrade the electrical infrastructure, but this solution often proves impractical due to the high financial costs and slow deployment. As an alternative, a more practical strategy lies in the intelligent scheduling and management of energy loads across time, leveraging the inherent flexibility in when and how EVs are charged [1].

In our ongoing work, we have been developing EnergAIze [2], a modular and intelligent energy management framework designed to coordinate consumption across buildings, households, commercial facilities, and EVs, while respecting user preferences and broader Renewable Energy Community (REC) objectives. As we designed and deployed EnergAIze, a bottleneck emerged: in order to optimize EV charging behavior, the system requires access to essential data such as the vehicle's current SoC.

Regarding SoC data, while high-end commercial or DC fast chargers often have integrated support for retrieving SoC directly from the connected EV, most residential AC chargers, the sites where energy flexibility is most available, do not. Compounding this challenge is the lack of standardization in how SoC data is exposed by different automotive manufacturers. Each EV brand provides its own proprietary interface, and this is due to vehicles being critical systems that, if compromised, can provoke injuries or even death. As [3] explains, several car manufacturers had countless vulnerabilities exposed leading to a huge investment in the security of these systems.

This fragmentation has led researchers to experiment with On-Board Diagnostics II (OBD-II) solutions, which attempt to extract SoC and other telemetry from the vehicles' Controller Area Network (CAN) bus. Studies such as [4–7] illustrate this approach: some inject CAN frames to extract metrics like SoC, while others use dongles like the ELM327 paired with apps such as TorquePro to scan for usable data. However, these methods are unreliable and unscalable, as OBD-II implementations vary significantly across manufacturers, rely on non-standard Parameter IDs (PIDs), and lack long-term viability due to increasing restrictions. As highlighted in [8], many manufacturers are moving to restrict or encrypt OBD-II access altogether, citing cybersecurity and data privacy concerns.

To address these and provide EnergAIze with reliable SoC, we present PulseCharge, a scalable, privacy-preserving, and brand-agnostic subsystem for EV data acquisition. Rather than relying on invasive or insecure methods like OBD-II scraping, PulseCharge leverages a cloud-native architecture and standardized middleware that interfaces with both users and EVs through secure, extensible APIs, enabling the controlled retrieval of SoC directly from the manufacturers' sources, and related vehicle data across a heterogeneous set of EVs. PulseCharge supports multiple operational modes, from simple monitoring to full integration with intelligent control systems and communicates using a publish/subscribe model, based on AMQP, for seamless interoperability with other subsystems.

By resolving the SoC data acquisition barrier, particularly at the residential and small-commercial level, PulseCharge unlocks the potential for intelligent EV energy management at scale – precisely where it is most needed for a stable and sustainable energy transition.

2 Architecture

EnergAIze, the system from which PulseCharge is an integrating part, is composed of multiple subsystems, each with its own responsibility and independent operation. The

cooperation between these systems allows results that would be unattainable if approached individually.

2.1 High Level Architecture

PulseCharge's main goal is to serve as the middleware that provides a command interface to retrieve data from EVs. The EnergAIze subsystem that benefit from these features is Percepta [9], a middleware responsible for channeling and standardizing all environmental data. Figure 1 illustrates how the different modules integrate together.

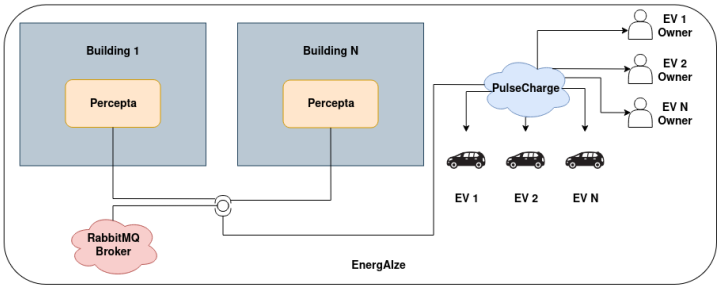

Fig. 1. PulseCharge's System Context

Percepta is deployed across each building. These replicas promote modularity and segregate responsibilities, because each building (e.g. houses, headquarters, etc) represents different people with different behaviors.

PulseCharge operates on the cloud alongside RabbitMQ Broker (RMQB), implying only one instance of these systems. RMQB enables seamless communication between all the building's replicas and PulseCharge. Depending on the building, Percepta is fed with the associated EV's SoC information in regular intervals, by default 45 minutes. This frequency can be altered by Percepta if needed.

2.2 Internal Components

PulseCharge collaborates with several other systems, which implies the existence of various Machine-to-Machine (M2M) communications, so that cooperation between these systems becomes an overall asset. In Figure 2, crucial components for the system operation are represented, external ones in orange, and the rest are internal.

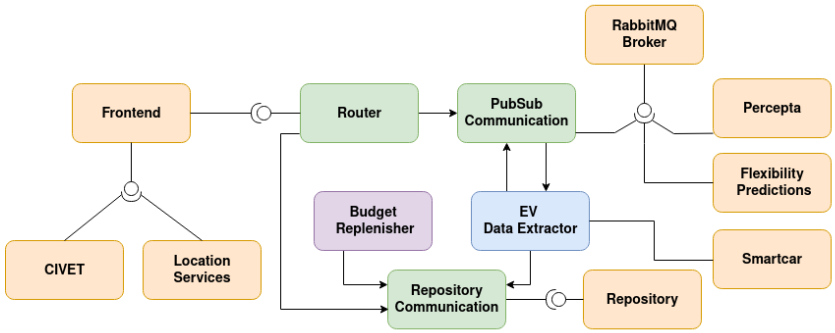

Fig. 2. Internal Architecture and Communication Flow

PulseCharge provides a frontend with two purposes: M2M communication and user interaction. CIVET – a long-distance route planner for EVs that focus on optimizing charging time, location and cost, while also balancing energy grid demand through collaborative energy communities along the journey – and Location Services recur to this module to indicate whenever a vehicle is in a transmittable state (as detailed in section 2.3, modes 3 and 4). End-users interact with the frontend through a dashboard to perform operations such as associate or disassociate a vehicle.

Router provides an interface, enabling frontend to issue the necessary requests, and forward them to the proper modules.

To access data from associated EVs, the EV Data Extractor is the primary component. It relies on an external module designated by Smartcar [10], a vehicle API that facilitates integration with EVs across different manufacturers. Smartcar provides access to various data points, such as those from the Battery Management System (BMS), vehicle location, and technical specifications. In addition to data retrieval, it also supports actuation capabilities, including sending destinations to the onboard navigation system, initiating or stopping charging sessions, and setting charge limits. This functionality enables the EV Data Extractor to consistently and efficiently acquire standardized data across a heterogeneous set of EVs.

This module limits the number of requests to 1000 per month for pulling data, so it is necessary to manage them properly to ensure they are evenly distributed until the request quota is renewed, avoiding running out of budget. Therefore, Budget Replenisher is proposed. This is the module responsible for controlling data acquisition from each EV. It ensures that a maximum of 16 requests is made within a 12-hour period (with accumulation allowed in case of unused requests), resulting in 992 requests per month (for months with 31 days), saving the remaining requests for any exceptional case.

To establish communication with the necessary systems in a transparent manner for the requester, the PubSub Communication is responsible for bridging this gap, abstracting the underlying messaging infrastructure. Persisting crucial information, such as system users, EVs and each one's quota, and their mapping, implies utilizing a database. Analogous to PubSub Communication, the Repository Communication module facilitates secure and reliable interactions with the database on behalf of requesters.

2.3 Operation Modes

PulseCharge features multiple levels of operation that add value to the final product – in other words, the more systems collaborate, the better the outcome. Figure 3 illustrates the pyramid of operation modes, where the incremental value decreases as it moves upwards.

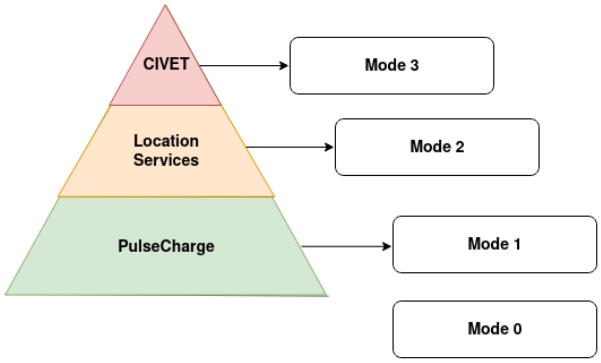

Fig. 3. Operations Mode Pyramid

The description of the modes of operation is given by:

- Mode 0 – There is no connection to PulseCharge, and it remains completely unaware of the vehicle status. The SoC is not transmitted.
- Mode 1 – This is the most basic operation mode, where it is possible to retrieve SoC from associated vehicles in a continuous way during the 12 hours assigned by the Budget Replenisher. Requests are made with a periodicity of 45 minutes (that results in 16 requests) to avoid running out of budget.
- Mode 2 – Location Services enables PulseCharge to request only during opportune times, by indicating whether a vehicle is in a transmittable or non-transmittable state. A vehicle is considered to be in a transmittable state only when it is charging or imminently about to begin charging.
- Mode 3 – CIVET provides PulseCharge with an estimated SoC that the EV is expected to have upon arrival at the charging location, allowing to set the EV in a transmittable state only when needed, optimizing data requests. Unlike Location Services, which suits into daily routines, CIVET fits into scenarios where the EV owner deviates from the usual routine.

3 Results

With the collaboration of a system user, it was possible to retrieve data from an EV over a period of four days. Figure 4 illustrates the fluctuations in the SoC of the EV from 11 July at 17:13h to 15 July at 13:36h. The vehicle in question is a 2019 DS 3 Crossback E-Tense, which offers an estimated range of 320 km, average energy consumption of 17kWh/100km, and 50kWh of battery capacity. The estimated charging durations are approximately 5 hours for standard charging and 23 minutes for fast

charging. This data was obtained with a periodicity of 45 minutes operating in mode 1 (independent).

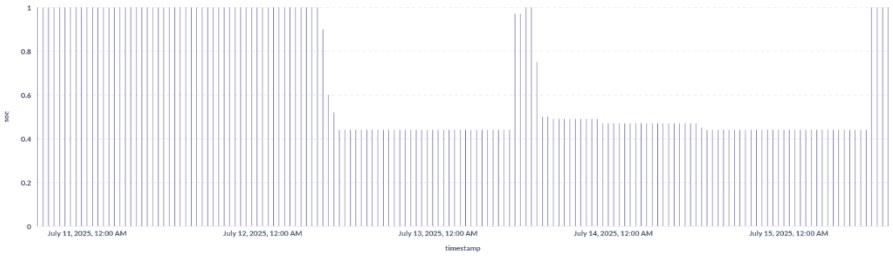

Fig. 4. System User's SoC from 11 July 17:13h until 15 July at 13:36h

The EV remained stopped with full charge throughout Friday and Saturday. On Sunday, a long-distance trip of approximately 120km was undertaken, explaining the SoC decay observed between 07:30h and 08:16h from 100% to 90% (10% decrease), between 08:16h and 09:01h from 90% to 60% (30% decrease), between 09:01h and 09:46h from 60% to 52% (8% decrease), and between 09:46h and 10:31h from 52% to 44% (8% decrease). This drop can be confirmed by performing the energy consumption estimative (based on the average energy consumption) and comparing it to the actual energy consumption (based on the battery capacity and the SoC drop obtained). The energy consumption estimative is given by the average energy consumption multiplied by the trip's distance, resulting in 20.4kWh. The actual consumption is represented by the battery capacity multiplied by the difference between the initial SoC and the final SoC (56%), resulting in 28kWh.

Upon analysis of the results, a discrepancy of 7.6 kWh was identified between the estimated and actual energy consumption. This deviation can be attributed to several external factors that significantly influence energy usage, including air conditioning (AC) usage, battery health, and velocity.

Subsequently, the next day, from 09:47h to 10:32h, the vehicle was recharged using a fast charger, resulting in a sharp increase of the SoC from 44% to 97%. Succeeding the recharge, the return trip began, and SoC dropped between 12:48h and 13:33h from 100% to 75% (25% decrease), and between 13:33h and 14:18h from 75% to 50% (25% decrease).

On the final day of observation, an abrupt spike in the SoC was recorded between 10:36h and 11:21h, rising from 44% to 100%. However, this data point is likely inaccurate, as the vehicle was charged using an AC charger during this interval. In the best-case scenario, the energy required to achieve a 56% charge within 45 minutes would be approximately the battery capacity multiplied by the SoC needed to achieve 100% divided by 45 minutes, resulting in 38.7kWh. Nevertheless, considering that the charger has a maximum output of approximately 3kWh, such a quick increase in SoC is technically implausible, suggesting erroneous readings during that period of time.

All other events were confirmed as accurate by the EV owner. This anomaly leads us to suspect that when the EV is being charged at home, particularly if the charging occurs beneath ground level, communication issues may arise, leading to delayed data

transmission. However, this cannot be confirmed with the current data, and further investigation is necessary to determine precisely the reason for occurrence.

4 Conclusion

PulseCharge lays the foundation for a scalable and interoperable EV data acquisition platform. Its integration within EnergAIze offers a robust approach to managing energy consumption and enhancing user comfort. As the system evolves with real usage data, it has the potential to significantly improve grid reliability and charging efficiency in the face of growing EV adoption.

The immediate next step involves onboarding new users to expand the dataset. This will enable the opportunity to analyze data patterns of users in detail and validate the system's effectiveness under real-world conditions.

Moreover, PulseCharge has potential to solve other issues that might occur during the development process. This potential includes advanced cases like considering comprise valuable user preferences (e.g. estimated time of arrival/departure, estimated SoC of arrival/departure, chosen charger if multiple are available) alongside with SoC, and send a location directly to an EV's onboard system indicating a close REC's charger, if the EV's range is not sufficient to reach the destination.

The periodicity of 45 minutes demonstrated to be insufficient to perform a valuable analysis with relevant conclusions, particularly in fast charging scenarios, therefore effort will be required to implement a mechanism that allows PulseCharge to extract data more frequently. Additionally, the causes behind the SoC delay transmission require further investigation.

References

1. Mischos, S., Dalagdi, E., Vrakas, D.: Intelligent energy management systems: a review. Artif. Intell. Rev. 56, 11635–11674 (2023). https://doi.org/10.1007/s10462-023-10441-3
2. Fonseca, T., Ferreira, L., Cabral, B., Severino, R., Praça, I.: EnergAIze: Multi Agent Deep Deterministic Policy Gradient for Vehicle-to-Grid Energy Management. In: 2024 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm), pp. 232–237. IEEE, Oslo (2024). https://doi.org/10.1109/SmartGridComm60555.2024.10738095
3. Elkhail, A.A., Refat, R.U.D., Habre, R., Hafeez, A., Bacha, A., Malik, H.: Vehicle Security: A Survey of Security Issues and Vulnerabilities, Malware Attacks and Defenses. IEEE Access 9, 162401–162437 (2021). https://doi.org/10.1109/ACCESS.2021.3130495
4. Khorsravinia, K., Hassan, M.K., Rahman, R.Z.A., Al-Haddad, S.A.R.: Integrated OBD-II and Mobile Application for Electric Vehicle (EV) Monitoring System. In: 2017 IEEE 2nd International Conference on Automatic Control and Intelligent Systems (I2CACIS), pp. 202–206. IEEE, Kota Kinabalu (2017). https://doi.org/10.1109/I2CACIS.2017.8239058

5. K., M., K.V., A.: IoT-based Web-Integrated OBD-II Telematics for Real-Time Vehicle Health Monitoring System. In: 2023 International Conference on Intelligent Technologies for Sustainable Electric and Communications Systems (iTech SECOM), pp. 91–96. IEEE, Coimbatore (2023). https://doi.org/10.1109/iTechSECOM59882.2023.10435307
6. Hong, S., Hwang, H., Kim, D., Cui, S., Joe, I.: Real Driving Cycle-Based State of Charge Prediction for EV Batteries Using Deep Learning Methods. Appl. Sci. 11, 11285 (2021). https://doi.org/10.3390/app112311285
7. Wong, C.: The Diagnostics Port – OBD2 in Detail and Manufacturer-Specific PIDs. MTHRFKNWIN. https://mthrfknwin.com/2016/02/04/diagnostic-port-obd2-detail-manufacturer-specific-pids/?utm_source=chatgpt.com, last accessed 2025/07/08
8. Hammerschmidt, C.: German Car Industry Plans to Close OBD Interface. eeNews Europe. https://www.eenewseurope.com/en/german-car-industry-plans-to-close-obd-interface/, last accessed 2025/07/07
9. Ferreira, L., Fonseca, T., Severino, R., Venâncio, R., Soares Sousa, C.: Percepta: High Performance Stream Processing at the Edge (2025). https://doi.org/10.13140/RG.2.2.17394.82886
10. Smartcar: Smartcar Vehicle API. https://smartcar.com/, last accessed 2025/07/13