2021 International Conference on Sustainable Energy and Future Electric Transportation (SEFET) | 978-1-7281-5681-1/20/$31.00 ©2021 IEEE | DOI: 10.1109/SeFet48154.2021.9375812

# Compact Electric Vehicle Charging Station using Open Charge Point Protocol (OCPP) for E-Scooters

Deeksha Devendra, Shreya Malkurthi, Abhinav Navnit, Aftab M. Hussain*
PATRIoT Lab, Center for VLSI and Embedded Systems Technology (CVEST),
International Institute of Information Technology (IIIT) Hyderabad, India
*email: aftab.hussain@iiit.ac.in

***Abstract*—An Electric Vehicle (EV) charger, also called Electric Vehicle Charging Station (EVCS) is an infrastructure element that supplies electrical energy for recharging EVs. The paper focuses on the design and fabrication of two-wheeler (escooter) electric vehicle charging equipment. Considering the future scenario of mass privatization of EV two-wheelers on Indian roads, the outline discusses the product design based on fabricability, affordability, and ability to mass manufacture. The proposed architecture follows the Level 2 charging standards (240 Volts), and is based on the open charge point protocol (OCPP). The suggested EVCS is constructed considering the safety prerequisites of system administrators, installers, consumers, government agencies and others. The design of EVCS links to three industries: equipment manufacturers, software industry and electric power networks. This paper presents design considerations by elaborating the hardware, software, and protocols followed to design the Level 2 charging standard EVCS.**

***Index Terms*—Electric Vehicle (EV), Electric Vehicle Charge Station (EVCS), Web Sockets, Open Charge Point Protocol (OCPP), Level 2 EV Charging**

## I. Introduction

Passenger vehicles are responsible for most road transportation emissions [1]. In India, the transport sector emits an estimated 261 tonnes of $CO_2$, of which 94.5% is contributed by road transport [2]. This encourages the use of EV vehicles as an alternative to expensive conventional internal combustion engine (ICE) vehicles. India has successfully adopted regulations for EV vehicles. In 2019, around 20 million two-wheelers per year (of these, 6 million are scooters) were sold in India. At present, EV scooters represent a tiny percentage of these sales, however, they are expected to completely replace ICE scooters in the future. The large number of EV two-wheelers entering the market would require support of a huge network of EV chargers. The design of the system needs to be such that it not only charges the vehicle but also keeps it safe while charging. When the charger detects hardware faults, it disconnects the power, and prevents battery damage and possible shock hazards [3]. There needs to be a safety lock out which prevents flow of current when there is no charging taking place. Additionally, there should be a stop switch which immediately interrupts the flow of AC power when pressed and stops the charging process at once in case of emergencies.The software is instructed in a way that it stops the AC power supply when it goes beyond a defined minimum threshold and interrupts the charging circuitry [4]. The EVCS is built keeping in mind safety requirements advised by safety experts.

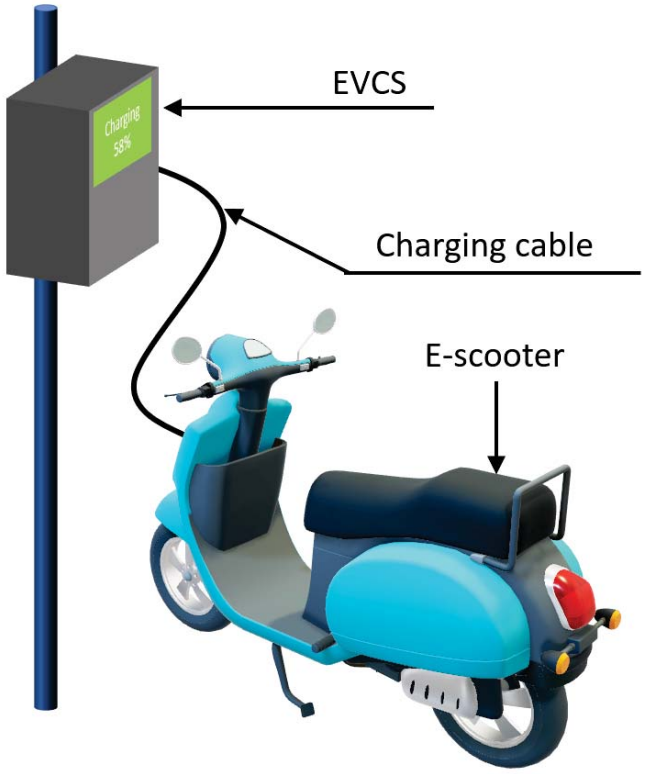


Fig. 1. Illustration of the compact electric vehicle charging station (EVCS) being used to charge an E-scooter.

The system architecture is built on the guidelines of Level 2 charging which denotes that the charger works on 240 volts single phase AC supply [5]. In this paper, we also discuss in detail the methodology to develop an electric vehicle charging station for 2-wheelers from initial steps of choosing hardware components to finally developing a working model. The development of the EVCS has been discussed in the paper in three sections: hardware, software, and protocols required for communication.

978-1-7281-5681-1/21/$31.00 ©2021 IEEE

## II. Hardware Design

This section details all the physical devices required to build an EVCS. Public EV chargers have very high duty cycles thereby the components should be selected based on their sturdiness and durability. With the vast option for electronic components present in the market, the choice of each device must be made by scrutinizing the pros and cons of all the options for each component and picking the one that works best for the application. Each component is selected based on the following three criteria. First, the component should fulfil the basic specification requirement and should be easy to interface with the chosen processor. Second, the selected component should be scrutinized based on the manufacturer of the component and the documentation available for the component. Third, the selected component should be of optimum price based on its function in the product. The architecture of the system consists of input device, processor, power meter, relay and output device. The system also needs to be packaged in a sturdy enclosure for outdoor deployments. The diagram in Figure 2 illustrates the basic block diagram of the system architecture. The processor, the power meter, and the relay receive power supply from 220 V AC mains. The AC power supply is converted into 5 V DC supply using a rectifier module which powers the processor board. The processor powers the keypad, RFID, and display monitor.

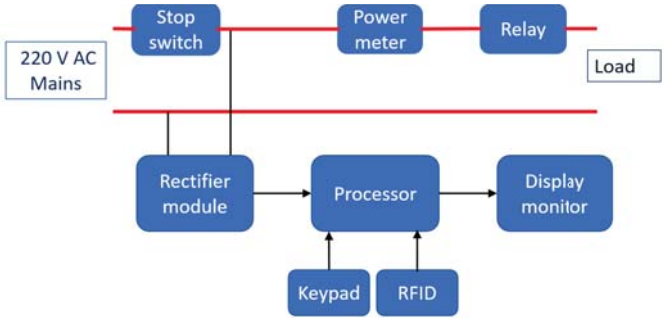


Fig. 2. Block diagram representation of the system architecture. The red lines indicate the connection from the input AC grid to the EV (load).

### A. Processing device

This is a key element in the design of the system. The processor controls all the input and output devices of the system and uses the OCPP protocol to maintain constant communication with the server. Thus, selection of an appropriate processor is one of the most significant tasks in designing the system. With numerous kinds of processors available in the market, various factors must be taken into account while making this selection: performance, power, operating voltage, and the peripheral devices. Our system is powered by Raspberry Pi 3 Model B+ processor. The system has 48 digital GPIO pins which are sufficient for interfacing with all the input/output devices. It allows for use of a Linux-based operating system for easy control of I/O devices such as display, UART devices, internet connectivity and so on.

### B. Input devices

Input devices are the ones responsible for accepting data from the user. In our system, input devices incorporated were keypad and RFID card reader. The keypad helps the user interface with the display while the RFID card reader enables the system to identify the user. In our design, we have used a flexible membrane keypad because it is inexpensive, water-resistant and shock resistant. It does not have separate moving parts, thus has a longer lifespan than mechanical keypad. The membrane keypad consists of a matrix of keys, which are set as rows and columns of conductive traces. The keys are arranged in such a way that the key press event is recognized based on the action of the area of the keypad matrix only when an electrical contact is established between one of the row traces and one of the column traces [6].

### C. Power meter

Power meter is an important part of the EVCS because it measures the amount of charging done by the user. The basic considerations to be made while choosing a power meter are the voltage/current ranges that must be monitored and the parameters to be measured. The power meter in our design is PZEM-004T Meter. It interfaces with the processor using TTL serial interface. Its small size and high AC current range make it suitable for use in a compact EVCS. It calculates instantaneous active power, voltage, amperage and active power accumulated (in Wh) over time. The serial UART of the component works at 5 V so we put a resistance of 1K to make it work at 3.3 V to interface with the processor.

### D. Relay

There are several different types of relays in the market like mechanical relays, solid-state relays, and FET switches [7]. In our EVCS design, we have used a solid-state relay. Electromechanical relays have limitations in package size, density, and speed. Solid-state relays are better alternative because they do not have moving parts. Also, we have not used FET relays because they are more suitable for DC voltages. The relay module used has input supply of 3-32 V DC, with output supply of 24-380 V AC.

### E. Output devices

The EVCS has a 16 A 3-pin plug as a controlled power output for charging. This is controlled using the solid-state relay, which is controlled using the processor. The user uses this plug to charge their e-scooter. We have included, in our design, a 7-inch touchscreen LCD display, as per Govt of India regulations for EVCS. We use this LCD to display the GUI and the user enters all the information and process steps on this LCD with the help of the keypad. It is connected to the processor by an HDMI cable. The photograph of the complete system is shown in Figure 3.

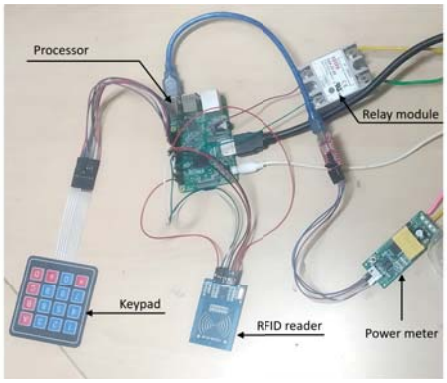


Fig. 3. Photograph of the fully functional system with all the major hardware components.

### F. Enclosure

The dimensions of the enclosure are 22 cm x 22 cm x 6cm. It has has cut-outs for the LCD display at the front and the 3-pin plug output at the side. We also included an emergency stop button that halts the complete process in case of an emergency. In this design, an AC supply is first connected to the stop button and then goes through the power meter and relay module to the output plug. The AC power is also used for powering the processor and other electronic components. The enclosure has been designed in Autodesk Fusion 360 [8]. It has been used to position all the circuit components used in the charger inside the box and to ensure this is the optimum design. The .stl files or. sldprt files (if Solid Works is used) for the components we used are readily available online and can be used to verify the enclosure design as shown in Figure 4.

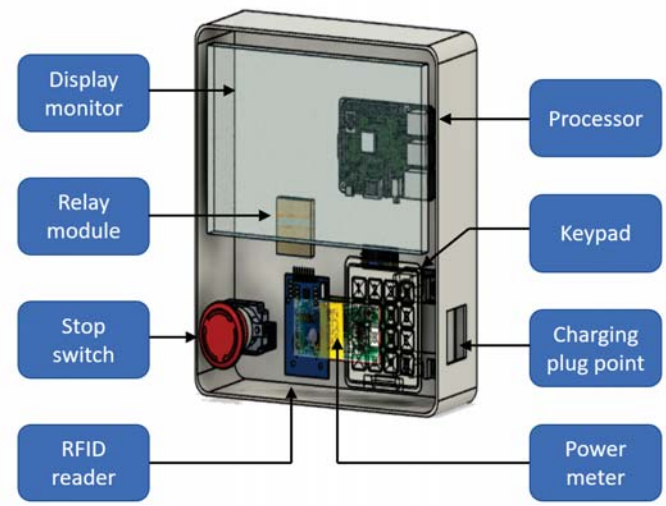


Fig. 4. Three-dimensional view of the enclosure design with all the hardware components.

## III. Software Development

The software is chosen such that all the necessary features exist in a single software, is easy to code, interfaces with all the hardware components, and is freely available for use. We chose Python3 to code our EVCS. Python3 is popular in the programming community for its simplicity and flexibility. According to the TIOBE index, Python is the 4th most popular language in the world [9], [10]. Its libraries make it easy for a developer to code for hardware interfaces and components. Moreover, Python's WebSocket library for handling server-client interactions makes it one of the best software for integrating EVCS. All the hardware components are structured into modules or methods or classes. Each module is called as per the hardware components' requirement in EVCS. This section discusses all these modules in brief. There is a setup required which defines the mode used to name the processor GPIO pins in the program. The setup is named as $GPIO.setmode$ and it can be either set to BOARD (raspberry pi pin numbers in ordered numbering) or to BCM (GPIO numbering defined by Broadcom). Some of the libraries to be imported are $RPi.GPIO$ for GPIO pins of Raspberry Pi, $serial$ for serial communication, and $Websockets$ for WebSockets communication.

### A. Keypad

Keypad matrix comprises of a set of conductive row traces and conductive column traces. Initially, the output of all the columns are set to $1$ and the output of all rows are set to $0$. When the key is pressed the row output changes to $1$. The keys are defined as a 2-D matrix. An empty list is set as a variable, say $z$, and a key is set as a flag. For instance, after the user has entered the amount (each number entered is defined as a variable key) in order to signal that the complete amount is entered '#' is pressed as flag otherwise every button pressed is appended in the list $z$. After # is pressed, the list of numbers are converted to a single number.

### B. RFID reader

The RFID reader used is RC522 RFID 13.56MHZ Reader Writer Module. Python supports the MIFARE library, $mfrc522$, for RFID reading, thus, we installed this library on our processor. The library is imported into the code and an object of the library is made in order to use the properties of the library. A method $read()$ of the library is called for it to read the assigned string on the card. A variable, which is the user id, is returned from this module.

### C. Relay

A GPIO pin on the Raspberry Pi board is defined as relay pin and the GPIO pin number allotted to the relay in Raspberry Pi is specified in the code. $GPIO.LOW$ allows to set the relay to low and $GPIO.HIGH$ gives instruction to switch on the relay.

### D. User interface

Tkinter is the standard object-oriented interface for designing graphical user interface (GUI) in Python. The complete user experience (described in the section below) is based on the use of the LCD display as the interface. The user can enter

the amount to be charged and monitor the charging processes using the display.

## IV. Communication Protocol

A protocol defines the set of rules for an interface between two or more systems to ensure compatibility. Protocols are generally selected based on their features like interoperability, accreditation standards, and its spread in the market [11]. The EV infrastructure requires protocols at two places: one from the EVCS to the EV and other from the charging point to the central management system.

### A. WebSockets

WebSocket protocol enables two-way communication between client and server, with the possibility of full duplex communication. It helps real time communication between the central management system and the EVCS. WebSocket helps build scalable, real-time web communication system. It efficiently provides a socket connection to the internet with minimal overhead. It delivers an enormous reduction in network traffic and latency compared to Ajax polling and Comet solutions that are often used to transmit real-time data to simulate full-duplex communication by maintaining two HTTP connections [12]. In our system, the EVCS sends the amount the user wants to charge and authorization id number. The central management system sends a query to the database to check if the authorization id entered by the user exists in the database and if the amount entered by them is less than or equal to what they have in their E-wallet. The libraries to be imported for web sockets communication are $asyncio$, $websockets$, $json$. Data is sent and received between client and server in JSON (JavaScript Object Notation) format. JSON is a lightweight language-independent text format which is used a data interchange format that is easy to parse and generate. Let $g$ be a variable which specifies the data to be transferred from the client side. This variable has two name-value pairs for amount and vehicle ID. The code serializes the data so that it can be transmitted through the network using TCP/IP protocol in the form of series of bytes. JSON library in python provides a method $dumps()$ which converts the python object into a JSON object. There is then a requirement of deserialization which decodes the delivered JSON data to a python object. The method used for this process is $loads()$. A variable, say $u$, states the address and port over which communication takes place and the $send()$ method is used to setup connection request with the defined host and address. The $recv()$ method is used on the receiving end of the setup network.

The chart shown in Figure 5 illustrates the bi-directional message transfer between EVCS and central management system. The arrows and writing in red colour symbolize the message flow from EVCS to the server. The purple colour shows the information flow from the server to the EVCS. The communication begins with client sending the request to the server. The server sets the two-way network. Users' RFID (authorization id) and amount of charging to be done

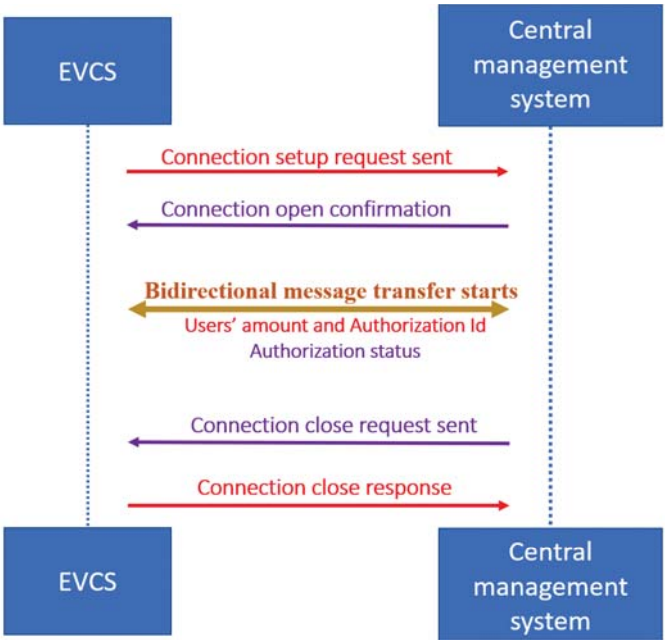


Fig. 5. WebSocket communication for implementation of the client-server interactions between the EVCS and central management system.

is sent by the client. The authorization status is sent by the server (central management system). When all the data is sent through the channel, the communication portal is closed with the consent of both the server and the client. Apart from server and client, an API is required for interactive communication between client and server. The websockets API in provide event driven responses without polling the server for reply.

### B. Open Charge Point Protocol (OCPP)

OCPP is a communication protocol between electric vehicle charging stations and the central management system at the application layer. The Open Charge Point Protocol (OCPP) offers a way to coordinate communication and flow of information between charging point and central management system, keeping direct interactions with EVs and the grid in an Internet of Energy [13]. The use of OCPP puts our EVCS design at an advantage over other stations as it allows users to roam between charging stations irrespective of the vehicle manufacturer. Hence, charging stations become interoperable and are no more vendor specific [14]. In Figure 6, the transmissions marked and written in red are the communication from EVCS to central management system and those marked in purple are the ones labeling the communication from central management system to the client.

## V. User Experience

The entire process of charging from the users' perspectives is described as followed. The user approaches the EVCS and is greeted with a welcome screen. The user then swipes an RFID card provided by the service provider to authenticate himself. The user enters the amount for which charging should be done via the keypad. As the user instructs for proceeding to

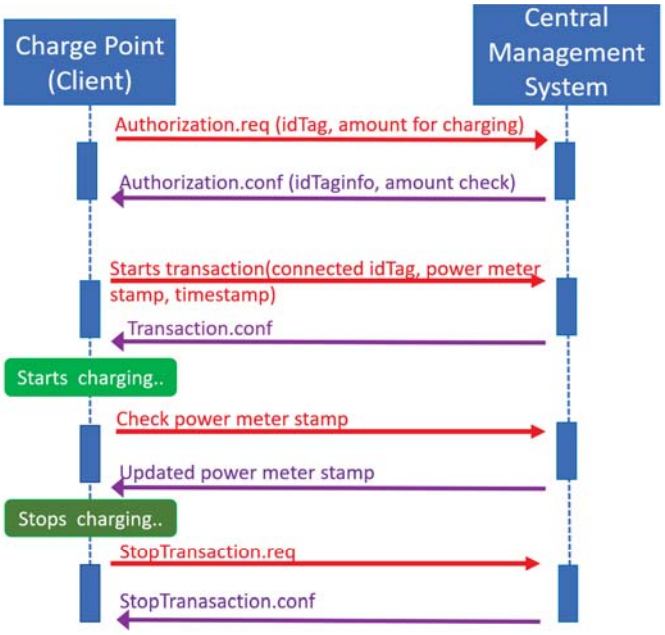


Fig. 6. Schematic illustration of the open charge point protocol (OCPP) used in the EVCS architecture for client-server communication.

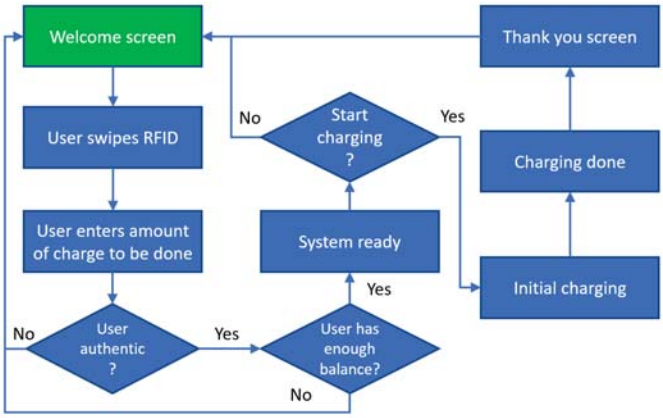


Fig. 7. Flow chart of the user experience for the designed EVCS.

authorization, web socket communication begins. The RFID information and charging amount are communicated to the server. The server checks the RFID string in the database to authenticate the user. The E-Wallet balance of the user is also verified against the charging amount. The display shows invalid authorization if RFID string is not in the database or if the e-balance is insufficient. The system then defaults back to the welcome screen. If the authorization is successful and the user has sufficient balance for charging, the system indicates that it is ready for charging. If the user confirms, the charging process begins. During the charging process, the system continuously displays the amount of charging that has been done. In case there is a power cut during the charging process, there is a battery back-up given to the EVCS system to move the data (amount charged) from primary volatile storage into non-volatile storage before the battery gets exhausted. Finally, at the end of the charging process, a thankyou screen is displayed. The system then defaults back to the welcome screen for the next user. This process has been represented in the flow chart shown in Figure 7.

## VI. Conclusion

In this paper, we have described the detailed system design and implementation of a level 2 charging standard EVCS using OCPP protocol. We have discussed in detail the specific components to be chosen with the necessity of each specific type. The libraries to be downloaded and used in the software part have also been enumerated. This work outlines the architecture for an EVCS which is affordable with no compromise on quality, safety and interoperability. It involves components that are easily available in the market and the software chosen are of standard accreditation. The future course of action on upgrading the EVCS involves substituting the input and output devices like keypad, RFID reader and display monitor with smartphone. In other words, we can modify the designed EVCS to make it compatible with any smart phone using QR-code based localization.

## References

[1] A. Singh, S. Gangopadhyaya, P. K. Nandaa, S. Bhattacharya, C. Sharma, and C. Bhana, "Trends of greenhouse gas emissions from the road transport sector in India," *Sci. Total Environ.*, vol. 390, no. 1, pp. 124–131, 2008.
[2] R. Shrivastava, S. Neeta, and G. Geeta, "Air pollution due to road transportation in India: A review on assessment and reduction strategies," *J. Environ. Res. Dev*, vol. 8, no. 1, p. 69, 2013.
[3] M. Tybel, A. Popov, and M. Schugt, "Assuring Interoperability between Conductive EV and EVSE Charging Systems," 2016.
[4] K. J. Brown and C. E. Ramirez, "Electric vehicle supply equipment with metering and communicatons," Patent, US20110320056A1, 2011.
[5] J. Sears, D. Roberts, and K. Glitman, "A comparison of electric vehicle Level 1 and Level 2 charging efficiency," in *2014 IEEE Conference on Technologies for Sustainability (SusTech)*. IEEE, 2014, pp. 255–258.
[6] M. O. Larsson, "Keypad, Keypad Matrix and Electronic Device," Patent, US20100122897A1, May 2010.
[7] V. Gurevich, *Electric relays: Principles and applications*. CRC Press, 2018.
[8] G. Verma, *Autodesk Fusion 360 Black Book*. BPB Publications, 2018.
[9] K. Srinath, "Python–The Fastest Growing Programming Language," *Int. Res. J. Eng. Technol*, vol. 4, no. 12, pp. 354–357, 2017.
[10] G. van Rossum, "Python programming language," *USENIX annual technical conference*, vol. 41, no. 1, p. 36, 2007.
[11] M. Neaimeh and P. B. Andersen, "Mind the gap-open communication protocols for vehicle grid integration," *Energy Inform.*, vol. 3, no. 1, p. 1, 2020.
[12] Q. Liu and X. Sun, "Research of web real-time communication based on web socket," *International Journal of Communications, Network and System Sciences*, vol. 5, no. 12, pp. 797–801, 2012.
[13] C. Alcaraz, J. Lopez, and S. Wolthusen, "OCPP protocol: Security threats and challenges," *IEEE Trans. Smart Grid*, vol. 8, no. 5, pp. 2452–2459, 2017.
[14] J. Schmutzler, C. A. Andersen, and C. Wietfeld, "Evaluation of OCPP and IEC 61850 for smart charging electric vehicles," *World Electr. Veh. J.*, vol. 6, no. 4, pp. 863–874, 2013.