Anh Pham

DEVELOPMENT OF SYSTEM DEMONSTRATOR TO MANAGE ENERGY DISTRIBUTION TO ALL CHARGING STATIONS OF A HOUSING COMPANY

Bachelor of Science Thesis
Faculty of Information Technology and Communication Sciences
Supervisor: Prof. Kari Systä
December 2022

ABSTRACT

Anh Pham: DEVELOPMENT OF SYSTEM DEMONSTRATOR TO MANAGE ENERGY DISTRIBUTION TO ALL CHARGING STATIONS OF A HOUSING COMPANY
Bachelor of Science Thesis
Tampere University
International Bachelor Programme in Science and Engineering
December 2022

In the current world of minimizing the use of fossil fuel, many individuals select electric vehicles as a modern solution for an emission-free private mean of transportation. A consequence of this increasing trend is the need of charging load management and energy distribution. Uncontrolled charging can affect not only the battery health but also availability for other vehicles, especially those in the same neighborhood. A system demonstrator is needed to limit charging capacity and prevent overloading. The demonstrator is implemented in EVCommunities project.

Keywords: electric vehicles, system demonstrator, charging station

PREFACE

I would like to express my gratitude to Professor Kari Systä for giving me an opportunity to work with the EVCommunities project and his supervision of this thesis. Moreover, I want to thank other colleagues of the EVCommunities project; Professor Outi Sievi-Korte, Terhi Kilamo and David Hästbacka have given me tremendous support during my research work.

Tampere, 3rd December 2022

Anh Pham

CONTENTS

1. Introduction 1
2. Background 3
    - 2.1 Basics of electrical system in buildings 3
    - 2.2 Basics of electric vehicles 4
        - 2.2.1 Battery electric vehicles (BEV) 4
        - 2.2.2 Plug-in hybrid electric vehicles (PHEV) 5
        - 2.2.3 Hybrid electric vehicles 6
        - 2.2.4 Plug-in charging 6
    - 2.3 Energy communities and energy markets 8
        - 2.3.1 Peer to peer (P2P) energy sharing 8
        - 2.3.2 Collective Self-consumption Models 9
    - 2.4 Adoption of smart grid technologies 10
    - 2.5 Smart technologies in buildings 11
3. Research project overview - EVCommunities 12
    - 3.1 Background 12
    - 3.2 Goals 12
    - 3.3 Motivation, purpose, and benefit of a system demonstrator 13
4. Implementation of a system demonstrator for EVCommunities 14
    - 4.1 Technical criteria and requirements 14
    - 4.2 Implementation steps 15
        - 4.2.1 Setting up the user client 15
        - 4.2.2 Setting up the back-end server and monitor client 25
5. Observation 31
    - 5.1 Achievements 31
    - 5.2 Current issues and further improvements 31
6. Conclusion 34

References 35

LIST OF SYMBOLS AND ABBREVIATIONS

AC	Alternate current, an electric current which periodically reverses direction and changes its magnitude continuously with time
BEV	Battery electric vehicles, a type of electric vehicle that exclusively uses chemical energy stored in rechargeable battery packs, with no secondary source of propulsion
CO2	Carbon dioxide
CORS	Cross-Origin Resource Sharing, a mechanism that allows restricted resources on a web page to be requested from another domain outside the domain from which the first resource was served
DC	Direct current, one-directional flow of electric charge
DER	Distributed energy resources, small-scale energy resources usually situated near sites of electricity use
EMS	Energy management system, a system of computer-aided tools used by operators of electric utility grids to monitor, control, and optimize the performance of the generation or transmission system
EV	Electric vehicle, a vehicle that uses one or more electric motors for propulsion
HTML	HyperText Markup Language, the standard markup language for documents designed to be displayed in a web browser
ICE	Internal combustion engine, a heat engine in which the combustion of a fuel occurs with an oxidizer in a combustion chamber that is an integral part of the working fluid flow circuit.
JSON	JavaScript Object Notation, an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays
LED	light-emitting diode, a semiconductor device that emits light when current flows through it
LEM	Local energy market, a marketplace to coordinate the generation, supply, storage, transport, and consumption of energy from decentralised energy resources

PHEV	Plug-in hybrid electric vehicles, a hybrid electric vehicle whose battery pack can be recharged by plugging a charging cable into an external electric power source, in addition to internally by its on-board internal combustion engine-powered generator
UI	User interface, where users can interact with the website or application
URL	Uniform Resource Locator, a reference to a web resource that specifies its location on a network

1. INTRODUCTION

The world is finding different approaches to replace fossil fuel and reduce carbon emission. Many aspects of people's lives are affected, from purchasing solar panels to using LEDs, and means of transportation is inevitable. While some people decide to use public transport, others select better alternatives of their private vehicles. Electric car is a progressive trend in Europe, with 550 000 and 1 325 000 units registered in 2019 and 2020, respectively [1].

High demand requires efficient management from the charging systems and mechanics to increase energy efficiency, especially in residential areas. With high installation cost and requirements, Finnish housing companies are hesitated to install charging stations for their residents' electric cars [2]. To meet the needs from EV users, an effective energy charging stations controller should be constructed to handle charging capacity of all EVs while still maintain electricity usage and safety in the building. This thesis focuses on developing and modeling a system demonstrator using fictional charging stations that solves the challenge, which can be illustrated in a web application.

The architecture of a standard web application can be divided into three main parts: the front-end, the back-end, and the database. The front-end provides interfaces for users to interact through their devices such as mobile phones, tablets, or laptops; the back-end handles the algorithm and user requests; and the database store necessary data for functioning purpose of the application. The system demonstrator, which is built based on this principle, can be used as a virtual system controller to retrieve user input, apply algorithms, and monitor and manage charging capacity of multiple EVs. Its benefits, impacts, and drawbacks will be discussed in multiple perspectives in this thesis. The system demonstrator is designed within a research project that provides necessary background information and statistics.

The remaining chapters of the thesis proceeds as follows. Chapter 2 details the background of EVs, the general electricity situation in a community, and why EVs are having a crucial role in the current energy market. Chapter 3 introduces the EVCommunities project with its overall structure and customers' usage and benefits. Chapter 4 analyze the system demonstrator's technical criteria and requirements, its implementation steps, including user clients and data monitoring. Observation from the demonstrator and conclusion will be discussed in chapter 5 and 6.

2. BACKGROUND

2.1 Basics of electrical system in buildings

This section discusses the basic level of electrical system in buildings, from the power source to an outlet in an apartment. The design is different between small and large buildings in terms of power distribution system complexity, but regardless of the building size, all electrical source comes from an electricity transmission grid. In Finland, the grid is operated by Fingrid, which provides thousands of kilometers of transmission lines cross the entire country, including 5100km of 400kV main transmission lines, 1300km of 220kV transmission lines, and 7300km of 20kV transmission lines [3].

In residential buildings, electric utilities provided by the power company include a transformer, which will be attached to a utility pole. The transformer reduces the high voltage from the transmission line down to residential use (which is 240V in Finland). The electricity is then passed to a meter, which keeps a record of power consumption. Next, wires transfer electricity to a panel board that has a main service breaker and a series of circuit breakers - usually one for each apartment - that control power flow to different building circuits. A circuit breaker helps protect its corresponding circuit from damage due to short circuit or high magnitude of current flow.

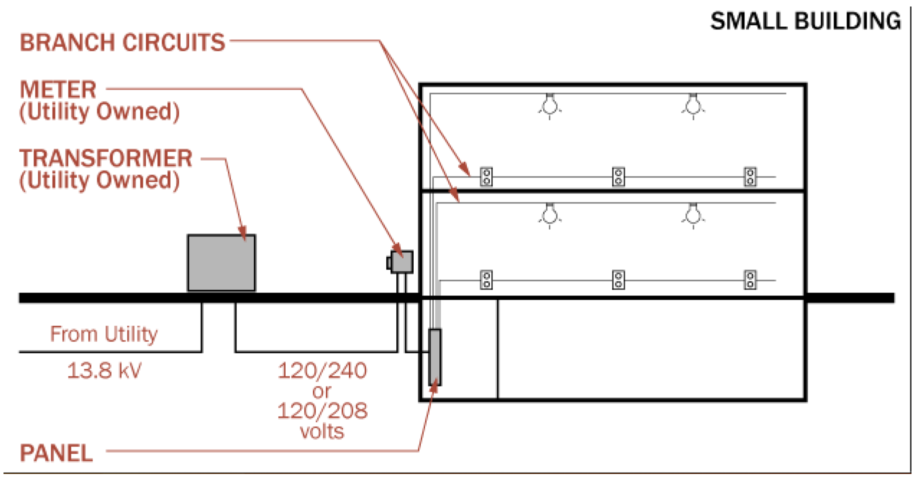


Figure 2.1. *Electrical system of a small building [4]*

On the other hand, larger buildings require higher electrical load, so larger and sturdier equipment is obliged. Similar to smaller buildings, the transformer in large buildings lowers the voltage, but electricity is then transmitted to a switchgear and travels along a feeder for safe and efficient electricity distribution to different areas in the building. Depending on the types of building, there might be an additional transformer to dial down the power for convenience outlets.

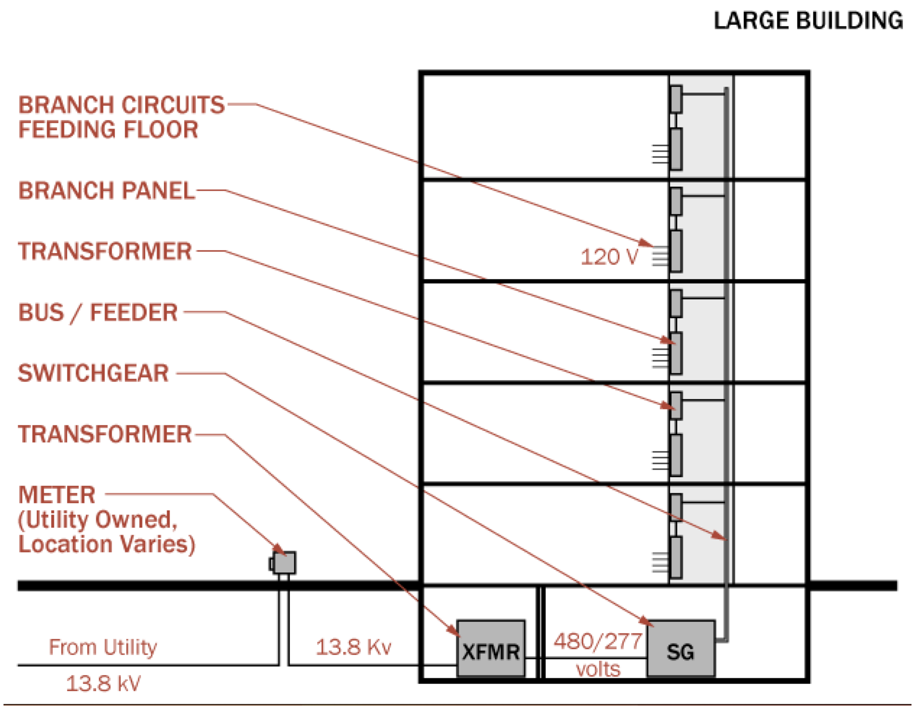


Figure 2.2. *Electrical system of a large building [4]*

Based on this setup of the energy system, EV charging stations can be installed and brought in to use in any modern buildings. According to the Energy Performance in Building Directive revised in 2018, all renovated residential buildings that have more than 10 parking spaces must have a pre-wired charging point equipped, and 20% of parking spaces in commercial buildings must be pre-wired [5]. New and currently under renovation buildings can provide sufficient infrastructure to utilize the latest EV models.

2.2 Basics of electric vehicles

The Alternative Data Fuels Center provides detailed information about the structures, components, and functionalities of different types of electric vehicles: battery electric vehicles [6], plug-in hybrid electric vehicles, [7], and hybrid electric vehicles [8], which will be summarized in the next three sections.

2.2.1 Battery electric vehicles (BEV)

All-electric vehicle, plug-in electric vehicle, or battery electric vehicle, is a type of EV that does not include petrol or diesel in the system; it can gain power through plug-in charging and run purely on electricity.

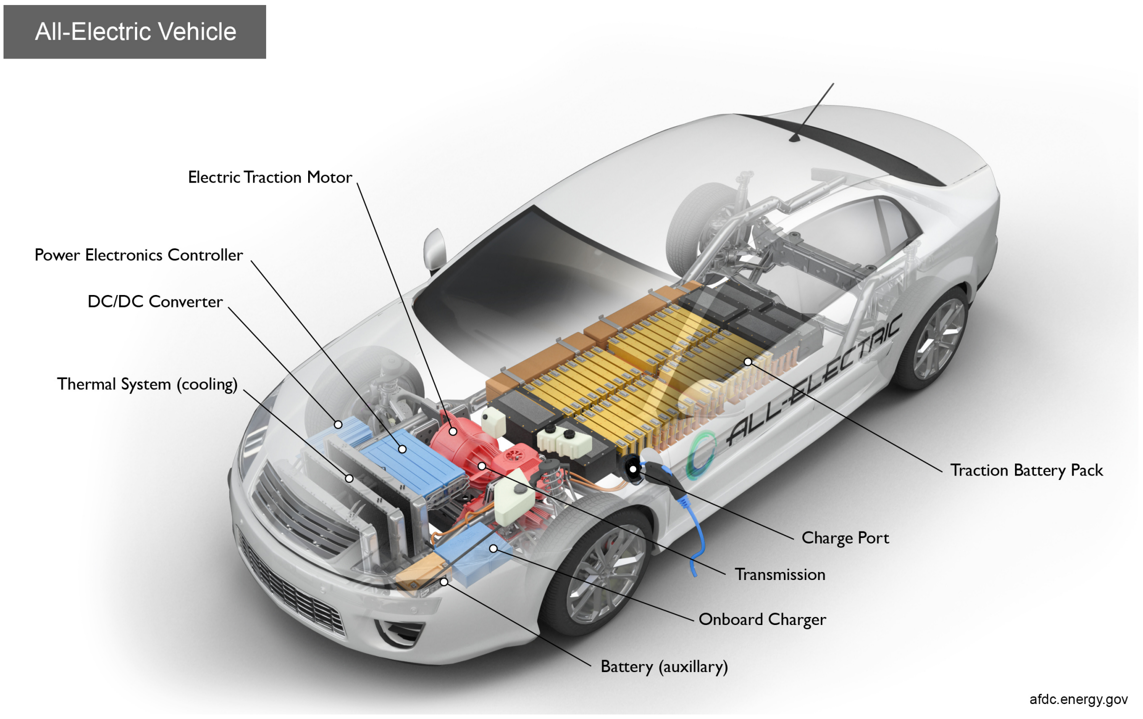


Figure 2.3. *Components of a battery electric vehicle [6]*

Every BEV has a charging port that allow connection between the vehicle and the AC power supply, which is then taken to the onboard charger and converted into DC power; the conversion will not appear if the users charge their vehicle from a DC charge point. Other functionalities of the onboard charger are communicating and monitoring charging equipment and different battery characteristics such as current, voltage, and temperature. The DC power generated from the onboard charger is used to recharge the traction battery pack - the main electricity storage. An electric traction motor retrieves energy from the battery pack only after the power has passed a DC/DC converter that converts high-voltage to low-voltage DC power for suitable usage. Energy from the traction motor needs to go through an electric transmission to transfer mechanical power to drive EV's wheels. A power electronics controller is included between the traction motor and the battery pack to monitor energy flow from the battery pack and controlling the motor's speed and its produced torque. A BEV also has an auxiliary battery that runs vehicle accessories by providing energy retrieved from the DC/DC converter.

2.2.2 Plug-in hybrid electric vehicles (PHEV)

As the name suggests, this kind of EV, besides the internal existence of batteries, uses another fuel, most commonly gasoline, to power an internal combustion engine (ICE). A PHEV mainly runs on electric power, and it only switches to use ICE when the battery is almost depleted. PHEV batteries, besides using a plug-in outlet, can be charged through

ICE and regenerative breaking - a mechanism that converts kinetic energy generated by slowing down a moving vehicle to electric power that can be stored and used when needed.

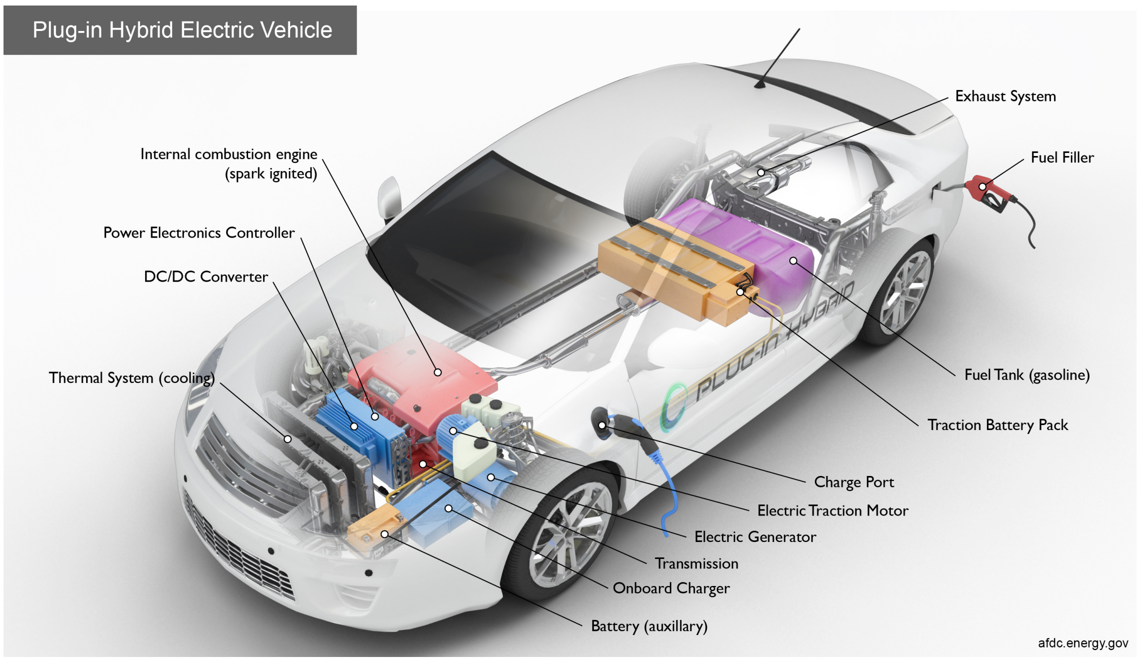


Figure 2.4. *Components of a plug-in hybrid electric vehicle [7]*

In addition to similar electrical components for using batteries included in a BEV, a PHEV has an electric generator that generates and transfers electricity from wheels while braking to the traction battery pack. For fuel usage, a PHEV includes a fuel tank that stores gasoline on the vehicle to be used when needed.

2.2.3 Hybrid electric vehicles

Compare to a PHEV, hybrid EV batteries cannot be charged by plugging-in; instead, it gains electricity through ICE and regeneration braking. A hybrid EV has similar fuel usage components in a PHEV. However, on the electricity side, since the batteries are not rechargeable by plugging in, the charging port and onboard charger are excluded.

2.2.4 Plug-in charging

BEV and PHEV are the two types of EV that support plug-in charging. This section provides basic information of different charging levels and cable types, which is gathered from Evesco [9].

EV charging can be separated into three categories: slow, fast, and rapid charging, or level 1, level 2, and level 3 charging.

- Slow charging is achieved by plugging an EV into a standard AC socket installed in

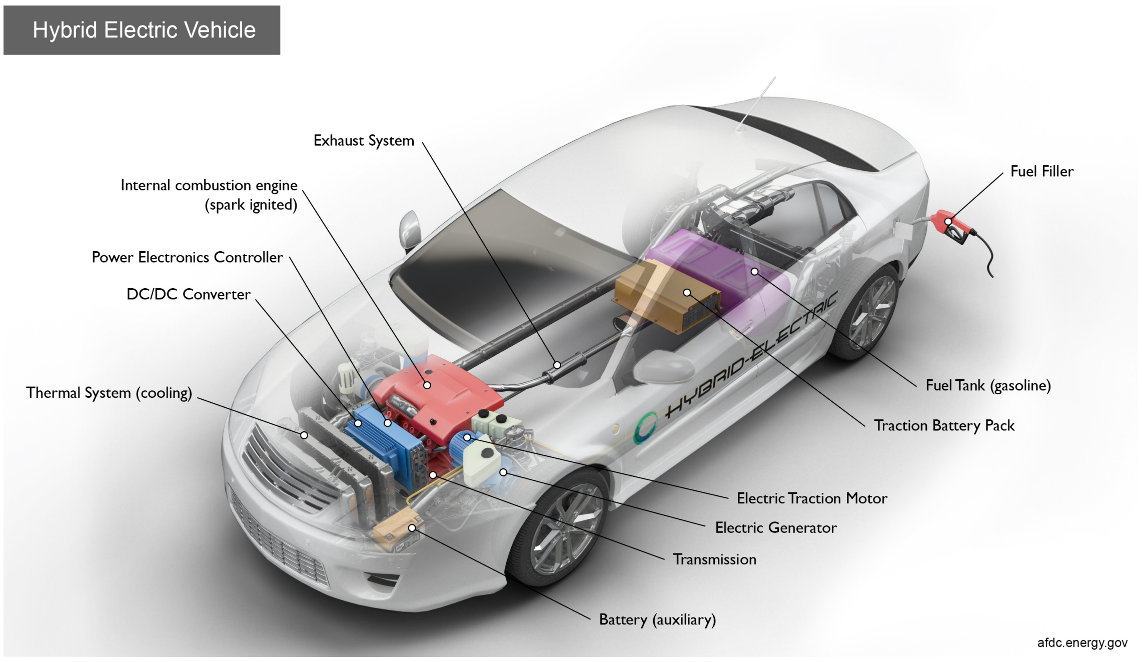


Figure 2.5. *Components of a hybrid electric vehicle [8]*

any households. This type of socket only provides from 1 kW to 1.8 kW of power and connects to the EV using a cable that has a standard household plug on one end and a J1772 EV connector on the other. An hour of level 1 charging only leads to a range from 4 to 11 kilometers of traveling.

- Fast charging uses a level 2 charger that can be found in a standard AC charging station, for example in public parking or any commercial locations. This charger delivers the power of 3 - 22 kW, which can provide a maximum of 120km of travel distance for an hour charge. This charging level uses Mennekes connector in Europe and J1772 connector in North America and Japan.
- Rapid charging uses direct current (DC) to bypass the conversion from the onboard charger installed in BEV and PHEV and charge the battery directly. As a result, this type of charging can deliver more power and faster, which is ideal for short-stop locations. Using a proper connector type (CCS 1 in North America, CCS 2 in Europe, and CHAdeMO in Japan), one hour of charging in this station with an output power between 30 kW and 360 kW can result in a travel distance from 193 to 2250 kilometers, depending on the charge acceptance rate from the EV.

Different types of EVs require different charging methods and gain different power input. Proper energy management is required to adopt with the rise of EVs.

2.3 Energy communities and energy markets

Despite having potential impact on reducing CO2 emission and oil consumption, overusing energy can lead to severe environmental problem such as air, water, and thermal pollution, and solid waste disposal [10], especially when the electric energy distribution process is a hierarchy model [11]. Therefore, advanced technology should be implemented that allows EV owners to share and receive energy from other owners.

2.3.1 Peer to peer (P2P) energy sharing

Peer to peer energy sharing can be defined as a virtual marketplace that allows party to buy and sell energy, managing price and volume risk, and gaining access to financial and non-financial benefits [12], which is a type of energy transaction that ensure prosumers' benefits and system operation's efficiency simultaneously. Unlike distributed energy resources (DER) where the the profit is marginal when trading energy, prosumers can participate in P2P energy market, negotiate the price with other peers, and trade energy actively and effectively. The P2P system can be divided into two layers: physically and virtually [13]. Grid connection, communication infrastructure, and smart metering are main components of the physical layer [14].

- Grid connection is essential for balancing energy demand and generation. These points also have smart meters that can evaluate P2P system's performance.
- Communication infrastructure is used to discover and support information exchange between prosumers. The infrastructure has to fulfill security, reliability, and latency of the system.
- Smart metering helps a prosumer choose suitable peers in the community based on information such as demand or market communication.

The virtual layer includes four basic elements: energy management system, information system, market operation, and pricing mechanism [14].

- Energy management system (EMS) bids in the market on behalf of its prosumer to simultaneously share and secure prosumer's energy supply. EMS makes decision based on various parameters set by the prosumer, including price and source of energy and prosumer's role in the market.
- Information system integrates energy parameters to a market platform and monitors the market's operation to assist prosumer's decision.
- Market operation provides real-time buying and selling orders service that allows prosumers to experience the energy sharing process precisely and efficiently.
- Pricing mechanism is needed to balance energy supply and demand within P2P sharing community. Additional cost such as taxes or subscription fees might include

in the energy price, but all pricing schemes should reflect the energy state in the community regardless.

Electric vehicles can also participate in P2P sharing market as a mobile storage. Through charging and discharging capabilities, EVs can provide demand flexibility [15] and frequency regulation [16]. Furthermore, since a battery of an EV can be transferred to the battery of another EV [17], EV is a promising element for the future of P2P energy sharing. Consequently, many studies and research have proposed different mechanisms to process EVs' P2P energy sharing. A blockchain-based technology is implemented in [18] to enable PHEVs to share energy in a localized P2P energy market, or an operational framework is developed in [19] for an EV charging station and a solar generation unit equipped in office buildings to participate in P2P energy trading, which provides noticeable benefits for both parties.

2.3.2 Collective Self-consumption Models

Self-consumption can be defined as using own electricity generated by various methods such as photovoltaic panels or mini wind generators [20]. The same principle applies to collective self-consumption, but on the bigger geographical and social scale [21]. This provides numerous benefits for the user, such as reducing energy cost, maximizing renewable energy used, and avoiding the installation of new power plants. Therefore, many collective self-consumption models have been presented, each has its own advantages depending on the community. This section introduces and compare three different models: static model, dynamic model, and local energy market.

- In static allocation models, energy is divided equally or with a fixed coefficient to all consumers, based on their contribution to the community. For instance, in a Finnish community such as apartment buildings, energy is first reserved for common areas then divided to households by the housing company stocks' ownership proportion [22]. At first glance, this seems to be a fair decision, especially for energy service companies, as more contributed households gain more benefit, but the core factor of collective self-consumption is to have a good ratio between energy produced and energy used. However, higher contributions per capita or financially do not equal neither higher energy consumption nor production, so other methods should be applied to the residential level of communities.
- On the other hand, in dynamic allocation models, members are encouraged to monitor and adapt the consumption based on the production [22]. This method tackles all the disadvantages in static models, since the self-consumption rate is higher when the energy is consumed approximately similar to those produced, which lightens the community's value.
- The last model to be discussed is a local energy market (LEM), where energy can

be traded between households in a local community. LEMs can be implemented with different designs [23]. A service provider controls the trade in a centralised market, while some forms of technologies, such as blockchain-based solutions, can provide a decentralised experience and members can trade energy directly.

Electric vehicle has its own roles in either of the models. In static and dynamic allocation models, EV is a consumption attribute, which affect how much energy is consumed by each of the community members, and good management will lead to a balance output in static allocation and an efficient self-consumption ratio in dynamic allocation. EV's participation can be further illustrated in LEM, where it can be used as a eco-friendly and efficient mobile trading mechanism between households. To make the most out of these methods, intelligence solutions are essential in any types of energy communities, which will be discussed in the following section.

2.4 Adoption of smart grid technologies

The traditional grid, or the electric grid, is a network of power stations, electrical substations, electric power transmission, and electric power distribution [24] that delivers energy from the power plant to apartments and buildings. Despite being considered as an engineering marvel [25] with the ability to create and move energy to different areas, the increasing power demand in recent years can cause extreme outcomes, such as blackout, which can lead to a domino effect of broken banking system or security vulnerability. Therefore, more advanced type of electric grid is required to adapt with the situation, which leads to the definition of a smart grid.

A smart grid includes a form of communication between the power source and the users so that the grid can response quickly based on the demand, which is accomplished by adding a wide variety of operations and measures included in the smart grid, such as advanced metering infrastructure, smart boards and circuit breakers, renewable energy resources, and sufficient utility grade fiber broadband [26]. The adoption of smart grid provides both long-term and short-term benefits for either the power plant, the energy provider, and the consumers. With latest technologies, a smart grid provides more efficient electricity transmission, which reduces peak user demand and lowers electricity rates. Moreover, higher amount of renewable energy will be available, and information security is improved. On one hand, this reduces the frequency of power disturbance for the power plant, hence reduces maintenance resources from the energy provider, and on the other hand, consumers will receive lower electricity bills and have a more convenience experience while still being security-protected.

2.5 Smart technologies in buildings

Energy management should be enhanced not only from the grid operators but also from the consumer side. While the applied methods vary depending on the community size and demand, this section only considers flats, or buildings, since 46.1% of the entire European population living in this type of housing unit in 2019 [27]. Different types of components and technologies are installed in a smart buildings, such as automated controls, meters, sensors, energy management systems, and analytic software. Smart plugs that control plug loads will be further discussed in detailed as one of the key contribution to building technologies.

When a device is plugged into an electrical outlet, the amount of energy drawn from the source is called a plug load. Therefore, this amount comes from the appliances of end users: refrigerators, laundry machines, and even electric vehicles, which lead to the accumulation of 47% of US's building energy consumption in 2020 [28]. Interestingly, plug loads have not been a regulated building load until recently. The International Green Construction Code has new standards that required receptacles to be controlled by switches, timer, or occupancy automatically [29]. Attached meter is also required in plug load to send energy consumption data for tenants billing. Recent smart buildings monitor and administer energy output from plug loads by using smart plugs that control receptacles automatically by remotely switching on and off appliances based on sensors feedback. Moreover, advanced power strip is another solution for energy flow management of all of the plugs on the strip.

Given the nature of consuming significantly higher energy compared to any other types of household appliances while being plugged in, it is necessary for EVs to have an intelligent solution that serves the similar purpose as a smart plug or an advanced power strip: monitor and control energy flow. The next section of the thesis introduces EVCommunities - a project that aims to monitor, manipulate, and improve EV charging.

3. RESEARCH PROJECT OVERVIEW - EVCOMMUNITIES

3.1 Background

Electric and plug-in hybrid electric vehicles are practical examples of the carbon neutrality trend; however, certain issues exists with the deployment of electric cars. First, installation of a charging station is a substantial investment [30], and the corresponding cost should be shared. Many garages and basements are mutually owned and used by the community residing in the area, and with different usage between individuals, having a fair share of the cost is a perplex challenge [31]. Secondly, extensive charging of EVs pressurizes the electricity grid [32], but inventive scheduling of charging can contribute a substantial impact to the problem.

Therefore, modern ICT intelligence is the solution for EVs to reach their full potential on saving energy cost and consumption. Initially, from the community perspective, charging infrastructure owned by EV users whose vehicles are moving can be rent to other users, which reduces the installation costs and land occupation of charging stations. Furthermore, with sufficient battery capacity and advanced technology, several EVs can be optimized as an energy storage, which increases flexibility markets alongside with participation in demand-response functions and external use. Most importantly, a software can be constructed as a system controller to manage and manipulate charging capacity between individual EVs in a community to avoid energy overloading and misuse.

3.2 Goals

Based on all benefits mentioned above, EVCommunities project's target is to develop foundations of a software system controller that can manipulate and improve EV charging, provide solutions for bookkeeping, and implement community-support features. One target of the controller is to allow users to control charging behaviour of their vehicles. For instance, the users can set when the battery should be full or select the maximum input power from the charging stations. Various challenges might arise when putting the system controller into operation: from a technical perspective, it is vital to develop an architecture for interoperability of independently developed subsystems, while discovering acceptable and commercially viable set of functionalities for the users. Therefore, to avoid spending budget on unnecessary appliances (e.g., actual EVs and charging stations) while the controller is still in the early steps of development, other alternative forms of illustrations need to be created for testings of the algorithm.

3.3 Motivation, purpose, and benefit of a system demonstrator

The core objective of the developed application is to create a simulation software for the system controller. In a charging community, such as apartments belonging to a housing company, the provided maximum power cannot be distributed for every EV owners to charge their vehicles simultaneously, but on the other hand, not every users requires charging at the same time. Therefore, a control system needs to be implemented to schedule charging power between charging stations while still fulfilling users' needs. The application should have certain features that allow users to set the time when the vehicle is needed to use, check the vehicle's current charging status, and get informed about the decision from the controller's algorithm. For instance, notifications should be displayed when the controller allows more charging power when less people is using, or another one when the charging network is crowded and the power is reduced. On the monitoring side, such as housing company managers or building owners, they should be able to observe the capacity distribution throughout the charging station network in order to have meaningful discussions and improvements.

Furthermore, the simulation software benefits algorithm testings, where charging capacity of the stations need to be limited both individually and as a whole. Each station's charging load at a certain period of time needs to be managed and updated depending on the total charging duration. However, when a large number of EVs are charging simultaneously, the total charging load can still be over the limit of the maximum capacity provided by the source, which can cause circuit break inside a charging spot. Therefore, an additional algorithm should be placed between the source and the stations to prevent overloading. This illustrates another purpose of the application: preventing hardware problems and issues. Without proper algorithm testing, which leads to electricity overload, the circuit breaker can be tripped, and the power of the entire circuit is shut down.

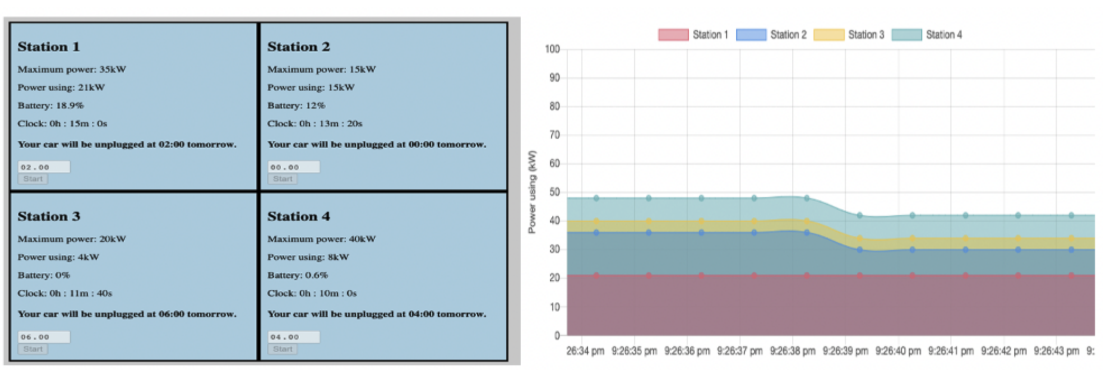
4. IMPLEMENTATION OF A SYSTEM DEMONSTRATOR FOR EVCOMMUNITIES

4.1 Technical criteria and requirements

From the user side, the minimum interface of the demonstrator need to show EV's current battery percentage, allow users set charging duration (or when the vehicle should be fully charged), and the current power input being applied to the vehicle. On the other hand, for monitoring purpose, the power being used by all user clients should be observed in real-time. With these provided requirements, a complete system demonstrator includes

- A user client which is implemented using a React - a JavaScript library for building user interfaces [33]. The interface illustrates predefined maximum power allowed, an input field to set when the charging is completed, and current charging power and duration and battery percentage as state variables, which is manipulated by an algorithm to update station's charging status. Lastly, there should be a button that sends such variables to the back-end once being clicked. Although one user client is meant to be used as one station, multiple stations will be included in the same client, which reduces memory usages for the local machine (for example, instead of sending single data from multiple clients, one client sends all data from the stations) while still preserving all necessary functionalities.
- A back-end server that stores the current power being used by all stations in the client.
- A front-end interface that visualizes the current state of all charging stations. The values are displayed by optimal graphs and charts.

When the application is completed, the clients should be friendly and easy to use while preserving all mentioned functionalities. With the simplicity of the demonstrator, the user should be able to view what can be performed on the client without further guidance, while no action is needed from supervisors to monitor the status. A rough design of the user and monitor client can be described in figure 4.1.



Station 1

Maximum power: 35kW

Power using: 21kW

Battery: 18.9%

Clock: 0h : 15m : 0s

Your car will be unplugged at 02:00 tomorrow.

02.00

Station 2

Maximum power: 15kW

Power using: 15kW

Battery: 12%

Clock: 0h : 13m : 20s

Your car will be unplugged at 00:00 tomorrow.

00.00

Station 3

Maximum power: 20kW

Power using: 4kW

Battery: 0%

Clock: 0h : 11m : 40s

Your car will be unplugged at 06:00 tomorrow.

06.00

Station 4

Maximum power: 40kW

Power using: 8kW

Battery: 0.6%

Clock: 0h : 10m : 0s

Your car will be unplugged at 04:00 tomorrow.

04.00

Figure 4.1. *The expected user client (left) and the monitor client (right)*

4.2 Implementation steps

This section illustrates how to implement the user client, the back-end server, and the monitor client respectively.

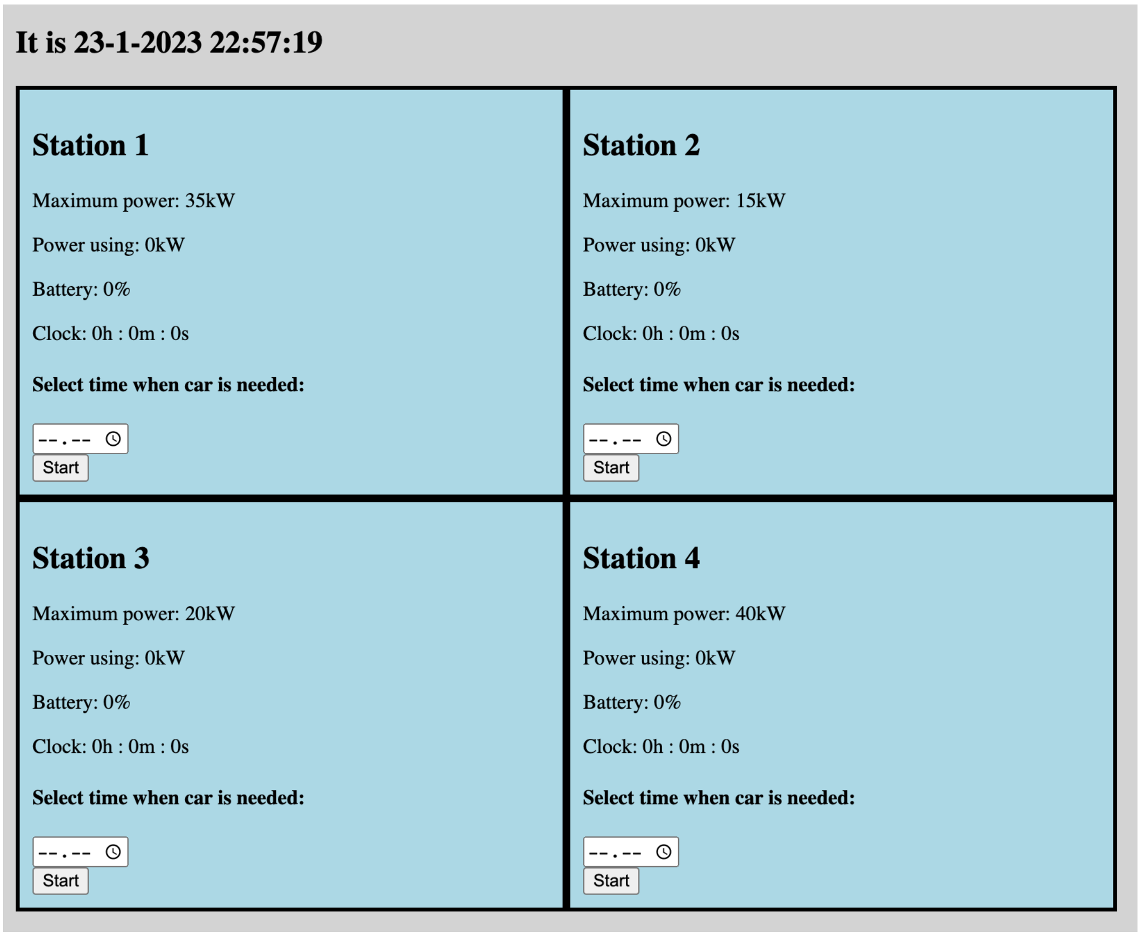
4.2.1 Setting up the user client

To fulfilled the requirements mentioned above, the design of a user client should be designed in a way that includes the information and functionalities in figure 4.2.

Each station includes the station name, maximum power, current power using (both measured in kilowatts), current battery percentage (increases from 0% to 100% maximum), a clock that measures total charged duration (since the user starts charging), a field to select when the charging should be completed, and a button to trigger the charging process. For the simplicity of the demonstrator, since all stations have same roles and functionalities, they will be included in one client that share the same charging algorithm. The client can be implemented using React.

A React app can be created using the `npx create-react-app user-client` command, where `user-client` is the name of the client directory. In this demonstrator, the `public` folder can be ignored, and in order to implement the client in figure 4.2, the `src` folder includes the following files

- `index.js`: Traditionally, this file is the actual entry point for not only React but also any Node.js app. It already contains necessary libraries and methods to render `App.js`, so it does not need additional code.
- `App.js`: The root component of the application. Theoretically, all `App.js` logic can be included in `index.js`, but it is beneficial to follow the community convention to have the entry file and other component files separately.
- `index.css`: Styles and layouts the application


It is 23-1-2023 22:57:19

Station 1

Maximum power: 35kW

Power using: 0kW

Battery: 0%

Clock: 0h : 0m : 0s

Select time when car is needed:

Station 2

Maximum power: 15kW

Power using: 0kW

Battery: 0%

Clock: 0h : 0m : 0s

Select time when car is needed:

Station 3

Maximum power: 20kW

Power using: 0kW

Battery: 0%

Clock: 0h : 0m : 0s

Select time when car is needed:

Station 4

Maximum power: 40kW

Power using: 0kW

Battery: 0%

Clock: 0h : 0m : 0s

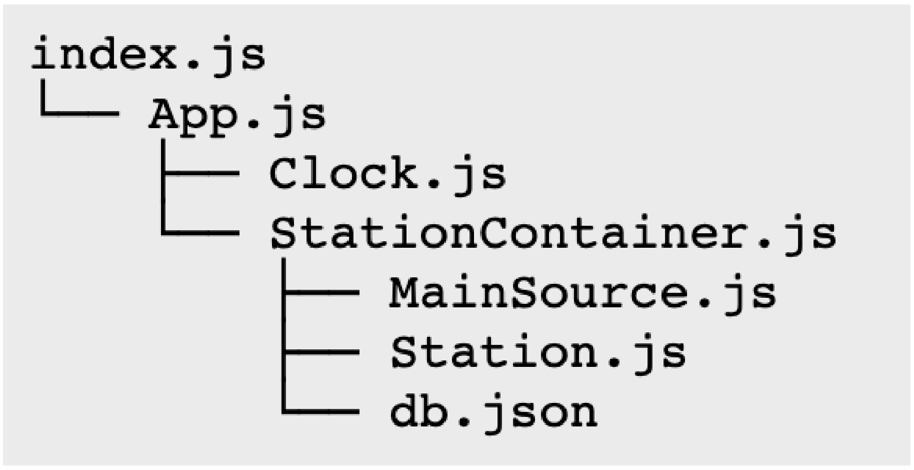
Select time when car is needed:

Figure 4.2. *The final user client*

- `Clock.js`: A component that defines the clock in real-time, which helps the user in scheduling how long the charging occurs
- `Station.js`: A component that implements a single charging station with necessary information and functionality
- `MainSource.js`: A component that shows the maximum allowed power to be used by all stations and the unused (idle) power.
- `StationContainer.js`: The parent component of `Station` and `MainSource`, which summarizes the power being used by all stations.
- `db.json`: A JSON file that includes necessary information of all charging stations - station id, name, and maximum power used.

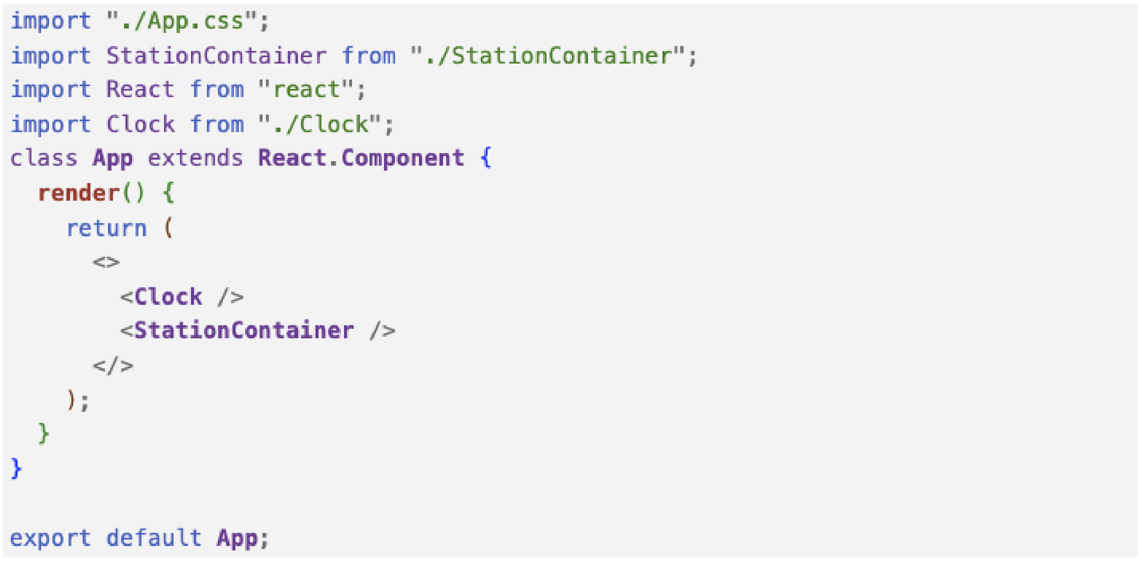
The structure of the application can be described hierarchically in figure 4.3.

Since all components of the application (`App`, `Clock`, `StationContainer`, `MainSource`, and `Station`) are implemented as class components, they need to extend from `React.Component` when being defined. Child components of `App` are `Clock` and `StationContainer`, so they have to be imported and included in the `render()` method of the `App` component.



```
index.js
└── App.js
    ├── Clock.js
    └── StationContainer.js
        ├── MainSource.js
        ├── Station.js
        └── db.json
```

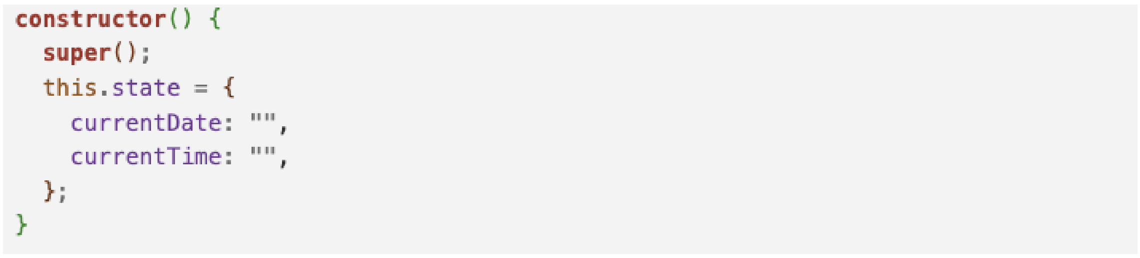
Figure 4.3. *Hierarchy structure of the application*



```
import "./App.css";
import StationContainer from "./StationContainer";
import React from "react";
import Clock from "./Clock";
class App extends React.Component {
  render() {
    return (
      <>
        <Clock />
        <StationContainer />
      </>
    );
  }
}

export default App;
```

Figure 4.4. *Implementation of App.js*

The core objective of `Clock.js` is to show the real-time clock on the client with exact date, hour, minute, and second, and therefore, state variables need to be initialize in the constructor.



```
constructor() {
  super();
  this.state = {
    currentDate: "",
    currentTime: "",
  };
}
```

Figure 4.5. *Initialize state variables in Clock.js*

Currently the application has two state variables: `currentDate` (the calendar format, for example 04/01/2023) and `currentTime` (the clock format, for example 14:30:00), and both of them are strings. `currentTime` needs to be updated every second, so it is sensible to update these variable (using `setState` method) in a `setInterval` function with a 1000ms delay. Moreover, the function needs to be invoked only once, so it should be declared in `componentDidMount`.

```
componentDidMount() {
  setInterval(() => {
    var today = new Date();
    var date = today.getDate() + "-"
             + (today.getMonth() + 1) + "-"
             + today.getFullYear();

    this.setState({
      currentDate: date,
      currentTime: today.toLocaleTimeString(),
    });
  }, 1000);
}
```

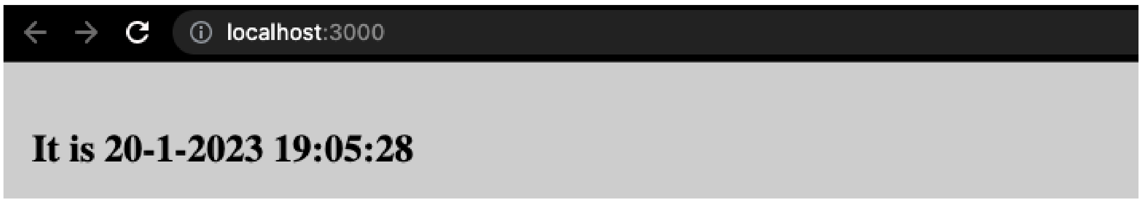
Figure 4.6. *Update date and time variables*

The `render()` method of class `Clock` can have some HTML code that display the date and time precisely.

```
render() {
  return (
    <h2>
      It is {this.state.currentDate} {this.state.currentTime}
    </h2>
  );
}
```

Figure 4.7. *Station ID, name, and maximum power are added to each station*

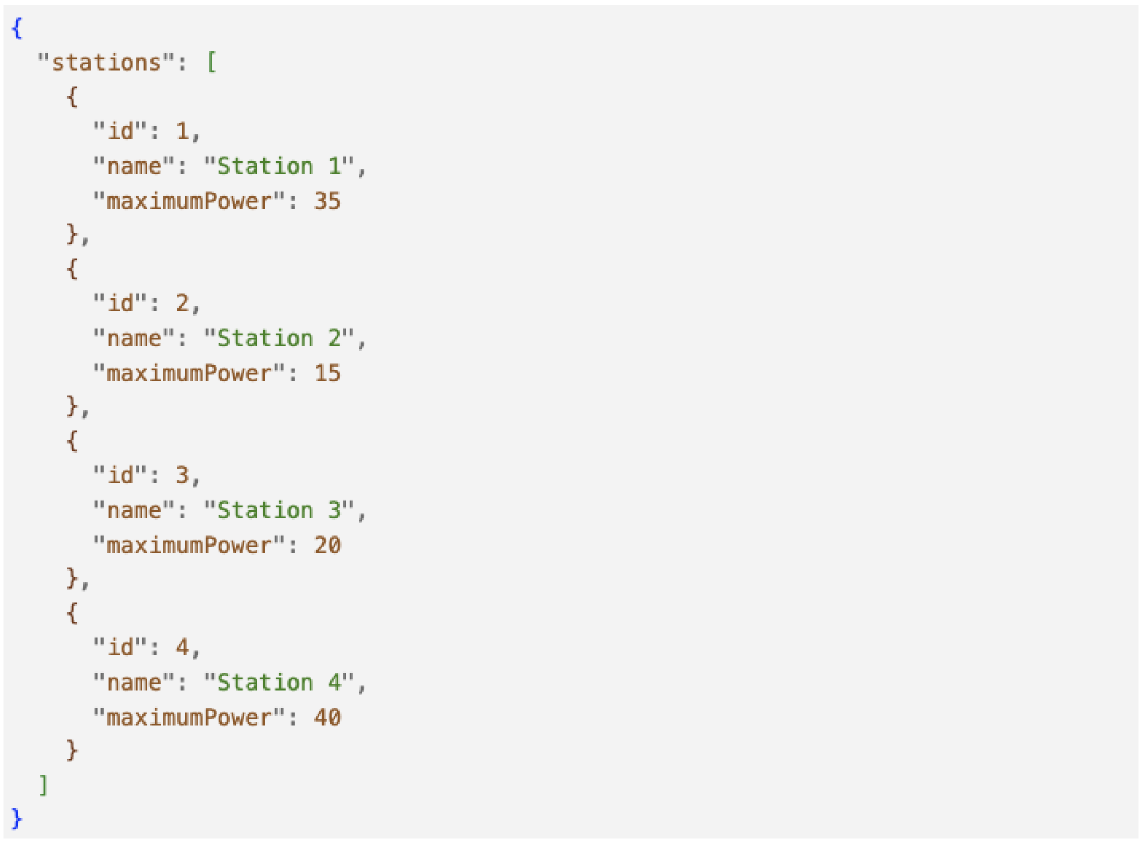
Running the application with `npm install && npm start` will render the expected real-time clock, which updates every second. The client can then be accessed at `http://localhost:3000`.


localhost:3000

It is 20-1-2023 19:05:28

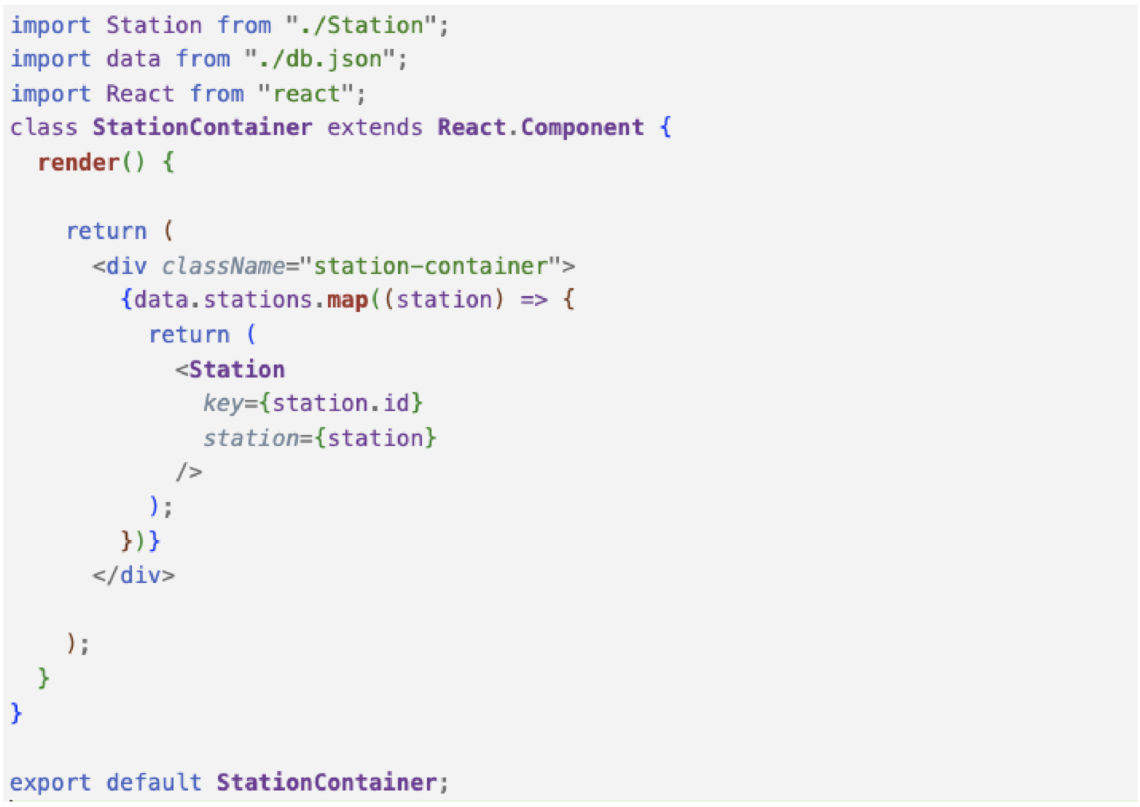
Figure 4.8. *Clock being displayed in the client*

The clock helps the user roughly estimate the total charging duration, based on the current time and when the charging should be completed. The latter is a feature that is implemented in `Station` component. Since all stations are included in one client in the demonstrator, the `StationContainer` component populates station data retrieved from `db.json` to variables in `Station`. Essential data can be added to `db.json`, which is populated in `StationContainer`.



```
{
  "stations": [
    {
      "id": 1,
      "name": "Station 1",
      "maximumPower": 35
    },
    {
      "id": 2,
      "name": "Station 2",
      "maximumPower": 15
    },
    {
      "id": 3,
      "name": "Station 3",
      "maximumPower": 20
    },
    {
      "id": 4,
      "name": "Station 4",
      "maximumPower": 40
    }
  ]
}
```

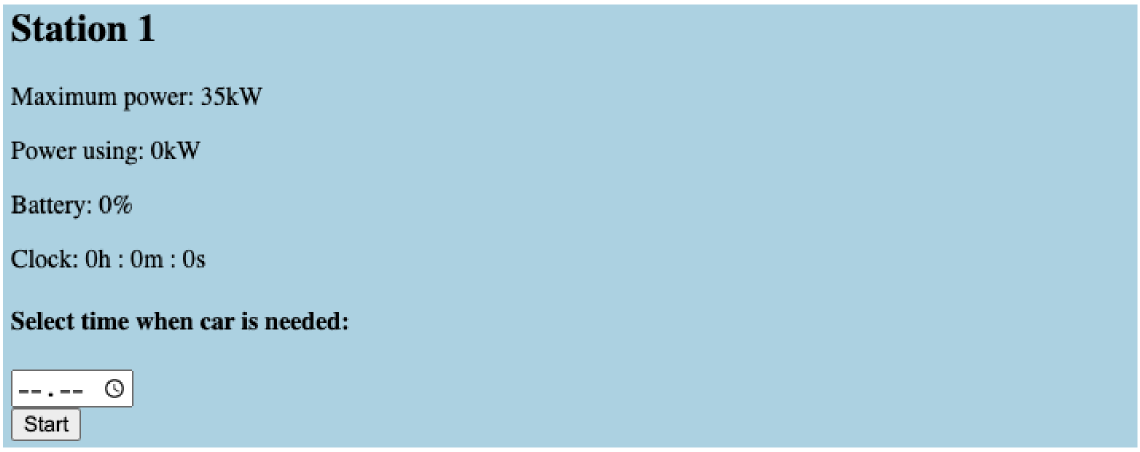
Figure 4.9. *Station ID, name, and maximum power are added to each station*



```
import Station from "./Station";
import data from "./db.json";
import React from "react";
class StationContainer extends React.Component {
  render() {

    return (
      <div className="station-container">
        {data.stations.map((station) => {
          return (
            <Station
              key={station.id}
              station={station}
            />
          );
        })}
      </div>

    );
  }
}

export default StationContainer;
```

Figure 4.10. *Implementation of StationContainer component*

For the `Station` component, based on the requirements mentioned in 4.1, the client for users should have information regarding station name, maximum power, current power, current battery percentage, charged duration, and the ability to select when the charging is completed.



Station 1

Maximum power: 35kW

Power using: 0kW

Battery: 0%

Clock: 0h : 0m : 0s

Select time when car is needed:

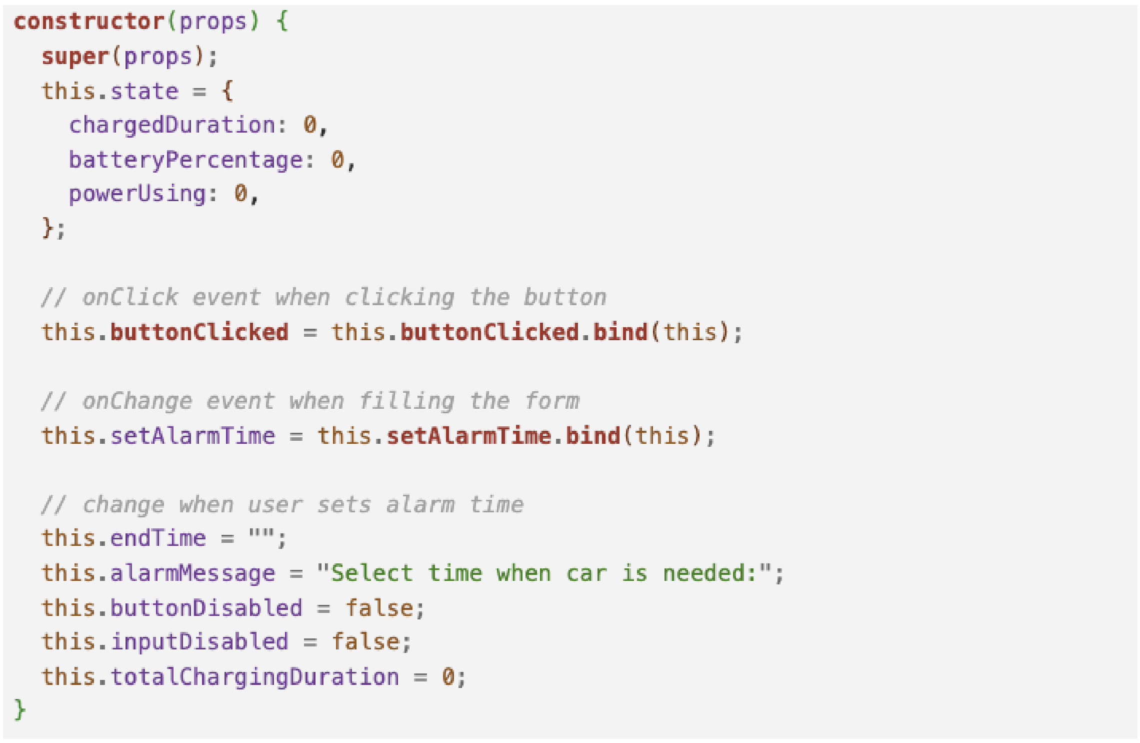
Figure 4.11. *The implemented user client*

From the figure above, this component requires three state variables for the user to monitor.

- `chargedDuration`: how long since the charging duration began, measured in seconds. For demonstrating purposes, this value will increase 100 times faster than the actual time. This helps the values of power using and battery percentage easier to be illustrated and observed.

- `batteryPercentage`: the current battery percentage value of the vehicle, in the range of 0 - 100.
- `powerUsing`: the current power being charged to the vehicle, changing between 0 and the maximum power allowed by the station.

Besides these state variables, other variables are initialized and certain functions need to be bound in the constructor so that the application can work properly. Furthermore, since the station properties (or props) are passed from `StationContainer` to `Station`, props need to be included as a parameter.



```
constructor(props) {
  super(props);
  this.state = {
    chargedDuration: 0,
    batteryPercentage: 0,
    powerUsing: 0,
  };

  // onClick event when clicking the button
  this.buttonClicked = this.buttonClicked.bind(this);

  // onChange event when filling the form
  this.setAlarmTime = this.setAlarmTime.bind(this);

  // change when user sets alarm time
  this.endTime = "";
  this.alarmMessage = "Select time when car is needed:";
  this.buttonDisabled = false;
  this.inputDisabled = false;
  this.totalChargingDuration = 0;
}
```

Figure 4.12. *Constructor implementation of Station class*

The user can select when the charging is finished using an input form with `type="time"` property. This form provides `onChange` property that calls `setAlarmTime` function when the user changes the value. The value, or the time set by the user, can be retrieved to calculate total charging duration.

Similar to input form's `onChange` event, the button element also has `onClick` property that calls a function when the user click the button. The charging is started once the button is clicked, so certain variables are updated as a consequence.

In the `buttonClicked` function, the `setInterval` method repeatedly calls three functions every second to update the power using, battery, and charging duration. Based on

```
setAlarmTime(event) {
  const inputAlarmTime = event.target.value;
  // convert time to seconds and calculate the time difference
  // between current and user-chosen time.
  // the maximum charging duration is 24 hours.
  const currentSecond =
    new Date().getHours() * 3600 +
    new Date().getMinutes() * 60 +
    new Date().getSeconds();
  const targetSecond =
    inputAlarmTime.split(":")[0] * 3600 +
    inputAlarmTime.split(":")[1] * 60;

  // set the end time of charging with better format
  this.endTime =
    inputAlarmTime +
    (targetSecond > currentSecond ? " today" : " tomorrow");

  // calculate the total charging duration in seconds
  this.totalChargingDuration =
    targetSecond > currentSecond
      ? targetSecond - currentSecond
      : targetSecond - currentSecond + 86400;
}
```

Figure 4.13. *Updating values when the user select time*

```
buttonClicked = () => {
  // Disable previously-edited input and button to prevent unwanted behavior
  this.inputDisabled = true;
  this.buttonDisabled = true;

  // Display selected end time
  this.alarmMessage = "Your car will be unplugged at " + this.endTime + ".";

  // The power using and charging duration are updated every second
  setInterval(() => {
    // Update the power using based on charging duration
    this.powerRouting(this.totalChargingDuration);
    // Update the battery and charging duration state
    this.checkBattery();
    // Send updated data to server
    this.sendDataToServer();
  }, 1000);
};
```

Figure 4.14. *Display, update, and send data when user triggers the charging*

the total charging duration calculated, the algorithm used in the user client mentioned in 4.1 classifies three different charging cycles

- Fast charging: When the total charging duration is short (less than 10000 seconds - approximately 3 hours), a large proportion of maximum power allowed is used in the first few minutes, which periodically decreases when the charged duration reaches closer to the total duration. In the first 20% of the charging period, the maximum power is used, which reduces to 60% in the next 20% of the duration and 20% in the remaining charging progress.
- Medium charging: This charging route is used when the total duration is larger than 10000 seconds but less than 20000 seconds (around 3 to 5.5 hours). Only 60% of the capacity is used in the first half of the duration, which reduced to 20% in the remaining half.
- Slow charging: The algorithm uses this route if the total charging duration is more than 20000 seconds, which is a long period of time. This only requires a small power to be used, which is 20% of maximum capacity.

To summarize, the power distribution can be described in figure 4.15.

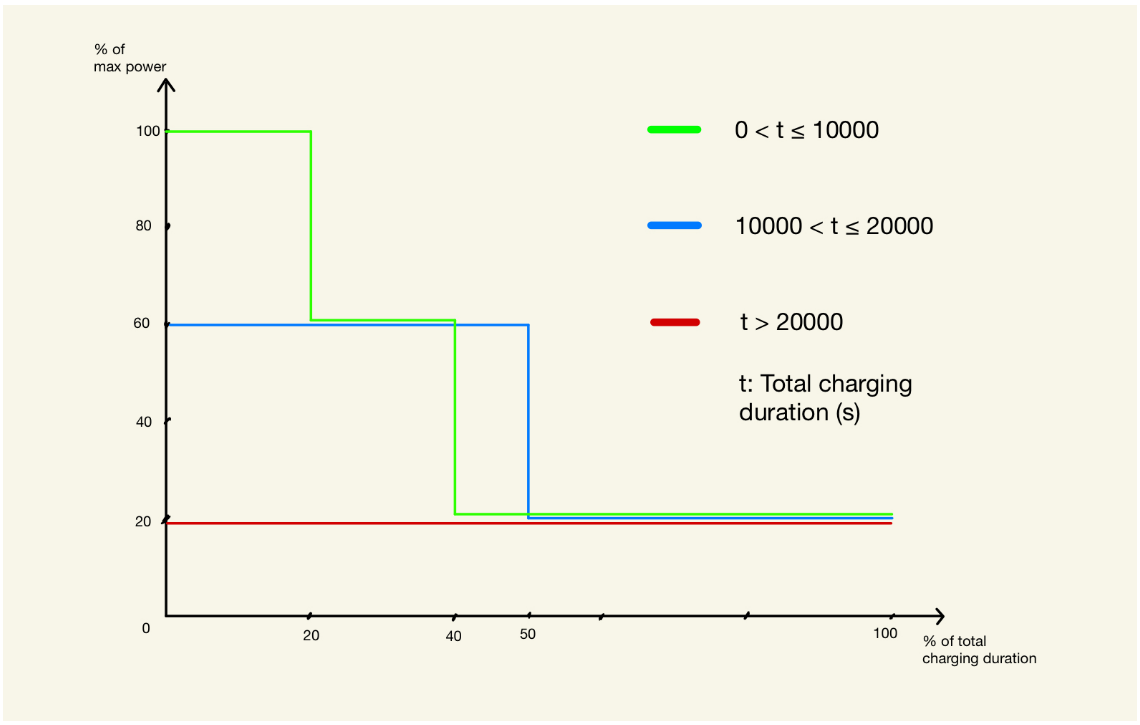
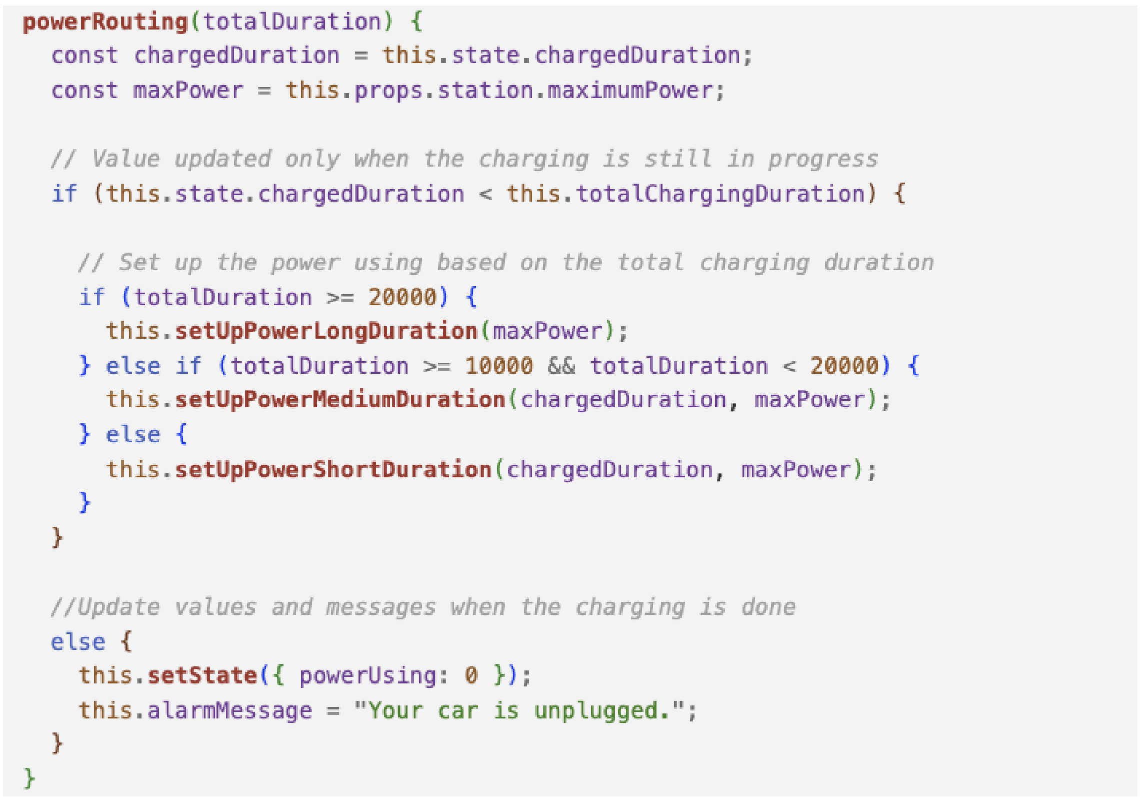


Figure 4.15. *Power distribution algorithm based on total charging duration*

For the simplicity of the demonstrator, it is assumed that all vehicles are identical, and the battery percentage's increase speed in the charging period only depends on the power using. A vehicle's charging energy A (measured in kilowatt-hour) during period t, assuming with the fixed charging power P can be calculated using equation 4.1.



```
powerRouting(totalDuration) {
  const chargedDuration = this.state.chargedDuration;
  const maxPower = this.props.station.maximumPower;

  // Value updated only when the charging is still in progress
  if (this.state.chargedDuration < this.totalChargingDuration) {

    // Set up the power using based on the total charging duration
    if (totalDuration >= 20000) {
      this.setUpPowerLongDuration(maxPower);
    } else if (totalDuration >= 10000 && totalDuration < 20000) {
      this.setUpPowerMediumDuration(chargedDuration, maxPower);
    } else {
      this.setUpPowerShortDuration(chargedDuration, maxPower);
    }
  }

  //Update values and messages when the charging is done
  else {
    this.setState({ powerUsing: 0 });
    this.alarmMessage = "Your car is unplugged.";
  }
}
```

Figure 4.16. *Routing different charging algorithms*

$$A = Pt \tag{4.1}$$

Therefore, the battery percentage can be calculated using equation 4.2

$$k = \frac{A_{charged}}{A_{total}} = \frac{Pt_{charged}}{A_{total}} \cdot 100\% \tag{4.2}$$

When the user starts charging for δt hours, the battery percentage increase during this period would then be

$$\delta k = \frac{P\delta t}{A_{total}} \cdot 100\% \tag{4.3}$$

In order to realistically calculate δk, a real-life battery from an EV needs to be chosen. The first generation of the Renault Zoe has the battery capacity at 22kWh [34], which will be applied in equation 4.3. Therefore, the battery percentage increase per 100 seconds $(\delta t = 100s)$ with respect to the power using P is

$$\delta k = \frac{P(100s)}{22kWh} \cdot 100\% = \frac{P(100s)}{(22kW)(3600s)} \cdot 100\% = \frac{1}{36} \cdot \frac{P}{22kW} \cdot 100\% \tag{4.4}$$

As observed in 4.4, the percentage changes proportionally to the power using; the more power being used, the faster the percentage increases. Therefore, both battery and power need to be updated periodically in all three previously separated charging algorithms. Furthermore, 4.4 provides the numeric calculation of battery increase based on the current power using, and combined with the definition of `chargedDuration` (which increases 100 times faster than the actual time for demonstration purpose), this equation can be applied directly to the code implementation. React's `setState` method provides a snapshot from the previous state of the variables, which enable state variables' values to be updated based on their previous values. This method can be used to update battery percentage, for example, in figure 4.17.

```
setUpPowerLongDuration(maxPower) {
  this.setState({ powerUsing: 0.2 * maxPower });
  this.setState((prevState) => ({
    batteryPercentage: this.roundNumber(
      prevState.batteryPercentage + (100 / 36) * (this.state.powerUsing / 22)
    ),
  }));
}
```

Figure 4.17. *Updating power and battery when the charging duration is long*

Next, in the `checkBattery` function called in 4.14, the charging duration count is updated and displayed while the battery is not yet full. On the other hand, the count stops, the power used is reduced to zero when the battery percentage reaches 100%. The `sendDataToServer` function will be implemented later, after the server is created.

The application is now usable at the address `http://localhost:3000`. With some styling added to `App.css`, the UI will be more user-friendly, which results in figure 4.2.

4.2.2 Setting up the back-end server and monitor client

The back-end server needs to continuously receive data - especially the power using - from the user client, and it is the responsibility of the monitor client to visualize such data clearly for observation, which will be displayed as a stacked line chart like figure 4.19.

To easily distinguish the role and purpose between the user and the monitor client, each of them will be served in a different port, which leads to the use of Cross-Origin Resource Sharing (CORS) in the back-end to enable data sharing from the user client. Besides, the demonstrator uses Express.js for building the server, from handling requests from users to sending responses. A `backend` folder can be created in the same directory as `user-client`, and it includes `index.js` file that implements route handling in figure 4.20.

The server can be started from the terminal with `node index.js`. When it is ready, the `sendDataToServer` function from the user client needs to be completed. Axios is used in the here to make HTTP requests, which is the power using by each station, to the server.

```
checkBattery() {

  // Update charging duration when the battery is not full
  // The time increases by 100 seconds every second
  // for the demonstration purpose
  if (this.state.batteryPercentage < 100) {
    this.setState((prevState) => ({
      chargedDuration: Math.min(
        prevState.chargedDuration + 100,
        this.totalChargingDuration
      ),
    }));
  }

  // Update values and messages when the battery is full
  else {
    this.setState((prevState) => ({
      chargedDuration: prevState.chargedDuration,
    }));
    this.setState({
      batteryPercentage: 100,
      powerUsing: 0,
    });
    this.alarmMessage = "Your car is unplugged.";
  }
}
```

Figure 4.18. *Updating clock and power using depending on the current battery state*

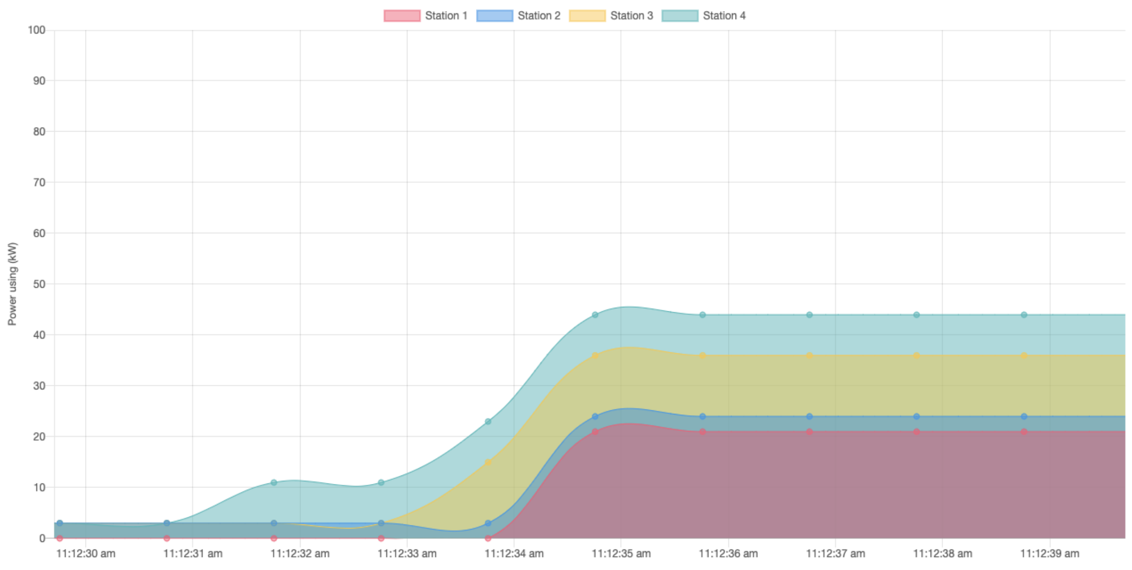


Figure 4.19. *Monitor client's chart design*

```
const express = require('express');

const app = express();
const cors = require('cors');
app.use(cors());
app.use(express.json())

const PORT = 8888
app.listen(PORT)
console.log(`Server running on port ${PORT}`)
```

Figure 4.20. *Initialize back-end implementation with Express and CORS*

The POST request requires the URL, where the data will be sent to, and the content that will be sent, which is a JSON object of the station. Besides what has been pre-defined in each object in `db.json` (station id, name, maximum power), the current power using must be included. A `post` method is implemented accordingly in the back-end to serve this request.

```
sendDataToServer() {
  var mutualUrl = `http://localhost:8888/`;
  var currentStation = this.props.station;
  currentStation["powerUsing"] = this.state.powerUsing;
  axios.post(mutualUrl, currentStation).catch((err) => console.log(err));
}
```

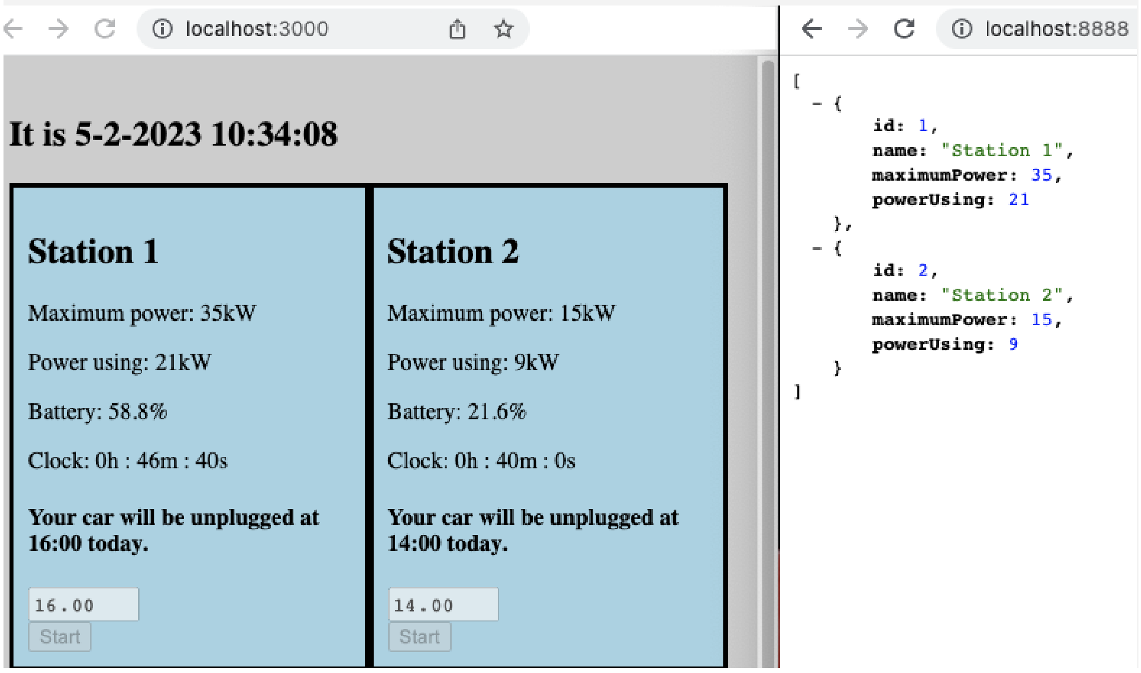
Figure 4.21. *Sending request from the user client*

```
app.post("/", (request, response) => {
  // Parse request body and create a station JSON object
  const body = request.body;
  const newStation = {
    id: body.id,
    name: body.name,
    maximumPower: body.maximumPower,
    powerUsing: body.powerUsing,
  };

  // Check if the station is already in the list. If not, add it to the list
  // If yes, update the station's information with new power using
  const idx = allActiveStations.findIndex((station) => station.id === body.id);
  idx === -1
    ? (allActiveStations = allActiveStations.concat(newStation))
    : (allActiveStations[idx] = newStation);
  response.json(newStation);
});
```

Figure 4.22. *Handling client's post request*

In addition to routing a POST request, a GET method can also be included to verify that the data is sent to the server properly. After that, the live data on the server can be observed at `http://localhost:8888/` while the user starts charging on one or more station on the client side.

```
app.get("/", (request, response) => {
  response.json(allActiveStations);
});
```



localhost:3000

It is 5-2-2023 10:34:08

Station 1

Maximum power: 35kW

Power using: 21kW

Battery: 58.8%

Clock: 0h : 46m : 40s

Your car will be unplugged at 16:00 today.

16.00

Start

Station 2

Maximum power: 15kW

Power using: 9kW

Battery: 21.6%

Clock: 0h : 40m : 0s

Your car will be unplugged at 14:00 today.

14.00

Start

localhost:8888

```
[
  - {
        id: 1,
        name: "Station 1",
        maximumPower: 35,
        powerUsing: 21
    },
  - {
        id: 2,
        name: "Station 2",
        maximumPower: 15,
        powerUsing: 9
    }
]
```

Figure 4.23. *Observing live station data from the server with get method*

When the data is ready, it can be fetched from the implemented server and displayed on the monitor side. To provide a clear and precise illustration in figure 4.19, Chart.js - a JavaScript charting library on modern websites - is a solution that provides high flexibility, responsiveness, and performance for the demonstrator, which also has the chartjs-plugin-streaming plugin that supports live streaming data. Besides, the plugin also required another date library, such as moment.js, in order to work properly. All 3 libraries and plugins have to be included in a new index.html file, where the graph will be displayed. In this file, a `<canvas>` tag is declared in order to draw a chart.

```
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <script src="https://cdn.jsdelivr.net/npm/moment@2.29.1/min/moment.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/chart.js@2.9.4"></script>
    <script src="https://cdn.jsdelivr.net/npm/chartjs-plugin-streaming@1.9.0"></script>
  </head>
  <body>
    <canvas id="myChart" width="400" height="200"></canvas>
    <script src="livechart.js"></script>
  </body>
</html>
```

Figure 4.24. *Creating HTML page for the live chart*

As hinted above, another script `livechart.js` will handle the data fetching and chart displaying on the canvas, which can be accessed using the component's ID.

```
var ctx = document.getElementById("myChart").getContext("2d");
```

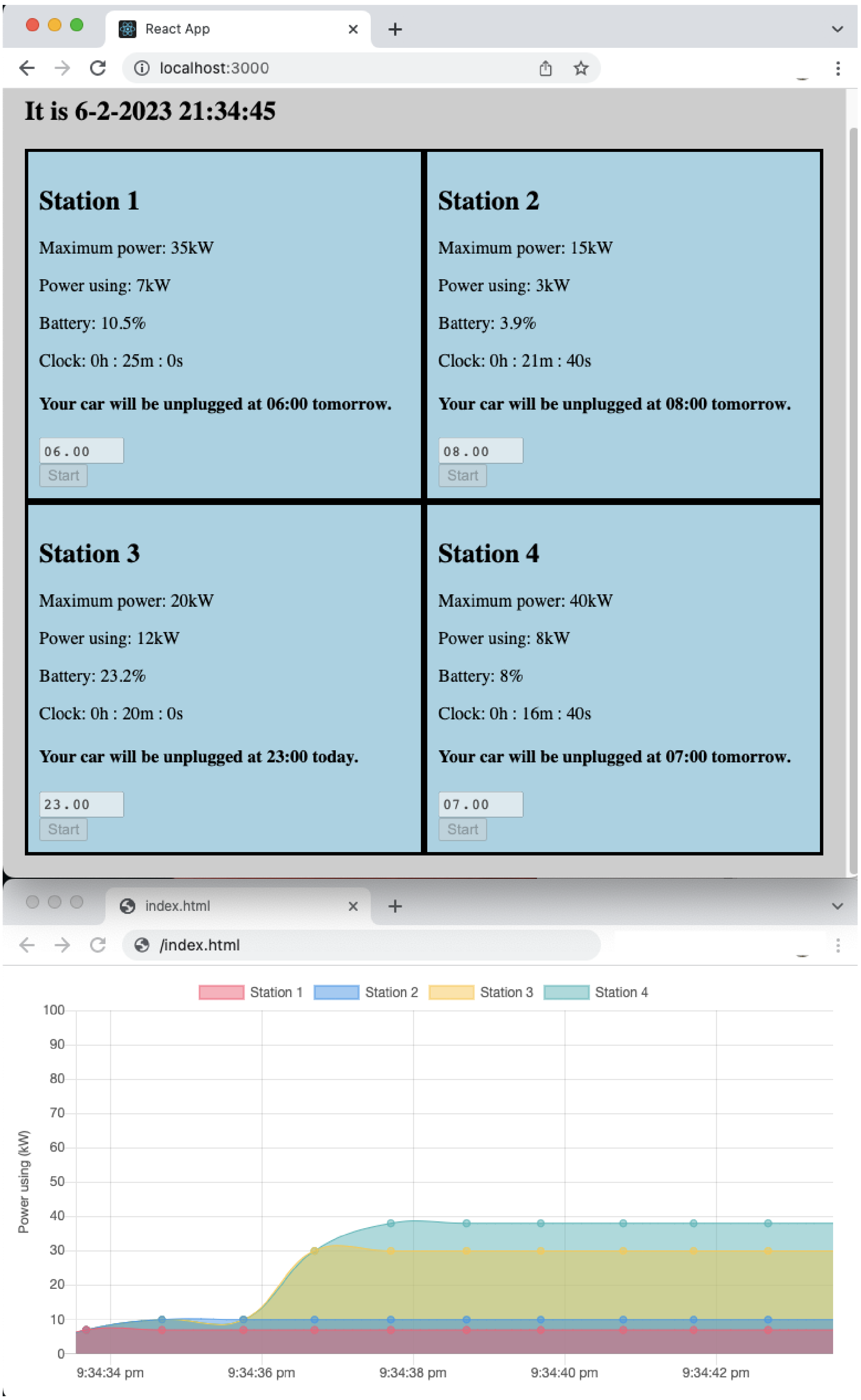
Figure 4.25. *Accessing the canvas component*

Next, a function that fetch data from the server needs to be created and called continuously in order to update the chart. The `fetchData` function is responsible for this task, and it is put in `setInterval` with a 1000ms delay. The newly refreshed list of stations is stored in an array.

```
var allStationsData = [];
async function fetchData() {
  const res = await fetch("http://localhost:8888");
  allStationsData = await res.json();
}
setInterval(function () {
  fetchData();
}, 1000);
```

Figure 4.26. *Setting up data fetching*

When the data is ready, it can be added continuously to the chart in real-time by following the instruction provided by the plugin's developers [35]. With both the user client and the back-end server running, the monitor page can run locally by opening the HTML file on a browser.



It is 6-2-2023 21:34:45

Station 1

Maximum power: 35kW

Power using: 7kW

Battery: 10.5%

Clock: 0h : 25m : 0s

Your car will be unplugged at 06:00 tomorrow.

06.00

Start

Station 2

Maximum power: 15kW

Power using: 3kW

Battery: 3.9%

Clock: 0h : 21m : 40s

Your car will be unplugged at 08:00 tomorrow.

08.00

Start

Station 3

Maximum power: 20kW

Power using: 12kW

Battery: 23.2%

Clock: 0h : 20m : 0s

Your car will be unplugged at 23:00 today.

23.00

Start

Station 4

Maximum power: 40kW

Power using: 8kW

Battery: 8%

Clock: 0h : 16m : 40s

Your car will be unplugged at 07:00 tomorrow.

07.00

Start

Figure 4.27. *The final product*

5. OBSERVATION

5.1 Achievements

Using React, the user client is successfully implemented with necessary functionalities. The user is able to observe station's information (station name, maximum power), set when the charging should be completed, and view the current charging status (the power using and battery percentage) in a simple but clear user interface. On the monitoring side, the stacked line chart provides a crucial indicator of how much power is being consumed at the same time as well as how much free power is left, how much power each station is using, and the charging process of that station. The algorithm implemented for each station has made a positive impact on the total charging power overall. From figure 4.27, only approximately 40kW of power is being used simultaneously, which saves more than half of the total maximum power allowed by 4 stations, which might allow 5-6 more stations to be installed and used comfortably.

Despite having multiple achievements, the demonstrator still has room for updates and improvements from both the user client and the monitor client. This will be discussed in the next section.

5.2 Current issues and further improvements

New features can be added to the user client to enhance customization and flexibility. Adding and removing stations is a crucial update. Realistically, in a building or a housing company, hundreds of charging stations will be occupied, and currently, it is possible to have such modifications by updating `db.json`, but this action requires understandings of coding and programming, which is not the case for every managers. Therefore, it would be beneficial for building managers to interactively updating stations' status by adding buttons, e.g., "Add a new station", "Remove a station", directly to the client. Next, the ability to modify a station's information is very important. To adapt with real-life situation, station's name should be updated, possibly based on users' name or apartment number. The maximum power allowed also depends on the station's infrastructure and the EV's capability. For example, if a station supports rapid charging (level 3 charger) but the EV only supports fast charging (level 2 charger), it is unnecessary to reserve a large amount of power for this station.

Besides, despite making a noticeable contribution to the reduction of power being used simultaneously, the implemented charging algorithm still has certain disadvantages. Currently, the algorithm devalues the purpose of charging in a community, where all stations use power from the same source. The charging power is always determined individually for each station without the consideration of other stations. Therefore, if the number of stations increases while the total power allowed remains the same, it is very likely that the limit will be broken. One solution to prevent this issue is to have an algorithm that checks if the sum of power reached a certain value (smaller than the actual limit allowed), the EVs with highest battery percentage will be temporarily discharged, and the process can be resumed when the power using drops below such value. Additionally, if there is surplus power available, stations with higher power requirement or stations that need to complete charging first should be prioritized to use more power. Another issue with the algorithm is that all EVs are identical, i.e., has the same battery capacity, which leads to the direct numeric calculation in equation 4.4. This does not provide flexibility for the building owners if they want to have the demonstration with the actual EVs in use. This issue can be resolved by adding an ability to choose the desired EV model for each station, and its corresponding battery capacity needs to be stored in the database, and the algorithm should use it to form the correct equation. In case of insufficient EV database (not fully updated with the latest models), the client should have an option to enter the battery capacity manually. Another contributor to equation 4.4 is the charging interval, which is currently 100s in the implementation. Similar to the fixed value of battery capacity above, this restricted users to view charging status in longer or shorter intervals over a certain period of time. Further customization can be foreseen as a solution, where the user can select which interval to be used in the demonstration, an the algorithm should be updated accordingly.

Finally, improvements need to be added to the monitor client. Currently, the graph only displays the current power using of all stations in real-time. It is not yet possible to view data from the past (e.g., how the stations behave during peak period the day before) which means that if there is an error with the algorithm that cause unexpected charging behaviour of the stations, it cannot be observed from the monitor side. The `chartjs-plugin-streaming` plugin does not have the feature to save past data, and it also requires the current server to replace old data by new data to support chart drawing. Therefore, a solution for this issue is to have all data that is sent to the server should also be saved locally in a JSON file with specific timestamps, and another program is needed to map such data to a chart. Another missing feature of the monitor client is that it can only observe station data but not directly manipulate station's status. For example, if there is a station that behaves badly, no action can be performed to fix the situation. A solution for this problem is that beside the chart, all stations' information should be displayed, possibly in a table format, and the value of power using must be editable that directly impact the station’s status.

6. CONCLUSION

The thesis first introduces the details of different types of electric vehicles with various charging capabilities and how they can contribute to energy communities and energy markets. Combined with the need of reducing CO2 emission for a more sustainable future, EV holds an important role in today's society. However, the massive development of EVs in recent years raises an issue on how to optimize the EV's energy consumption, especially for charging. This leads to the foundation of the EVCommunities project, and one of its target is to develop a system controller than can manipulate and improve EV charging in a community. Such controller also faces certain problems regarding the initial costs of infrastructure such as charging stations and EVs, and this paths way to the main topic being discussed in the thesis - implementation of a system demonstrator that illustrates how the charging behaviour of multiple stations simultaneously as a simple web application using React.

The demonstrator can be separated into three parts: the user client, the back-end server, and the monitor client. The user client includes multiple charging stations with different maximum charging power allowed but sharing the same functionality: receiving input from users on when the charging should be completed and starting the charging process following a button click. The back-end server serves as a database that is continuously updated with new charging data from the stations. The monitor client fetches such data and display it on a stacked line chart. Despite certain drawback occurs in different areas, the demonstrator has successfully provided all basic functionalities, from indicating the charging starting and ending point to viewing all station's charging status in real-time. The charging algorithms can certainly be improved with new features and functionalities, and those can now be added easily to the code implementation before applying them to the actual system controller. Using the demonstrator supports the testing of the algorithms and data visualization without substantial financial and infrastructure cost. Lastly, in a bigger picture, this will benefit EV charging in the community scale, create a noticeable impact on the society's direction on using EVs, and make the world a better place.

REFERENCES

[1] *New registrations of electric vehicles in Europe*. Nov. 18, 2021. URL: https://www.eea.europa.eu/ims/new-registrations-of-electric-vehicles#footnote-3URSKG9P (visited on 09/14/2022).

[2] *Housing companies hesitant to install electric car charging stations*. Feb. 15, 2019. URL: https://yle.fi/news/3-10649047 (visited on 09/15/2022).

[3] *Electricity system of Finland*. Fingrid. May 19, 2017. URL: https://www.fingrid.fi/en/grid/power-transmission/electricity-system-of-finland/ (visited on 11/08/2022).

[4] *Electrical System in Buildings - Archtoolbox*. Section: Electrical. Mar. 6, 2021. URL: https://www.archtoolbox.com/electrical-system-in-buildings/ (visited on 12/30/2022).

[5] *Questions & Answers on Energy Performance in Buildings Directive*. European Commission - European Commission. Apr. 17, 2018. URL: https://ec.europa.eu/info/news/questions-answers-energy-performance-buildings-directive-2018-apr-17_en (visited on 11/12/2022).

[6] *Alternative Fuels Data Center: How Do All-Electric Cars Work?* URL: https://afdc.energy.gov/vehicles/how-do-all-electric-cars-work (visited on 11/28/2022).

[7] *Alternative Fuels Data Center: How Do Plug-In Hybrid Electric Cars Work?* URL: https://afdc.energy.gov/vehicles/how-do-plug-in-hybrid-electric-cars-work (visited on 11/28/2022).

[8] *Alternative Fuels Data Center: How Do Hybrid Electric Cars Work?* URL: https://afdc.energy.gov/vehicles/how-do-hybrid-electric-cars-work (visited on 11/28/2022).

[9] Spendiff-Smith, M. *The Different Levels of EV Charging Explained - EVESCO*. Power Sonic. Oct. 17, 2022. URL: https://www.power-sonic.com/blog/levels-of-ev-charging/ (visited on 01/26/2023).

[10] *Environmental impact of energy — European Environment Agency*. 2004. URL: https://www.eea.europa.eu/help/glossary/eea-glossary/environmental-impact-of-energy (visited on 11/16/2022).

[11] Bober, D. and Kaproń, H. NEW MODEL OF ELECTRIC ENERGY CONSUMPTION CONTROL -ITS POSSIBILITIES AND RESULTS OF SIMULATION RESEARCH. *Rynek Energii* 89 (Aug. 2010).

[12] Raffaele, D., Bolwrk, V., Boske, L. en Tjallingii, I. *Peer to peer energy trading | Future of Energy*. Deloitte Netherlands. URL: https://www2.deloitte.com/nl/nl/pages/energy-resources-industrials/articles/peer-to-peer-energy-trading.html (bezocht op 28-11-2022).

[13] Mengelkamp, E., Gärttner, J., Rock, K., Kessler, S., Orsini, L. and Weinhardt, C. Designing microgrid energy markets: A case study: The Brooklyn Microgrid. *Applied Energy* 210 (2018), pp. 870–880. ISSN: 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2017.06.054. URL: https://www.sciencedirect.com/science/article/pii/S030626191730805X.

[14] Tushar, W., Yuen, C., Saha, T. K., Morstyn, T., Chapman, A. C., Alam, M. J. E., Hanif, S. and Poor, H. V. Peer-to-peer energy systems for connected communities: A review of recent advances and emerging challenges. *Applied Energy* 282 (2021), p. 116131. ISSN: 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2020.116131. URL: https://www.sciencedirect.com/science/article/pii/S0306261920315464.

[15] Zhou, Y. and Cao, S. Energy flexibility investigation of advanced grid-responsive energy control strategies with the static battery and electric vehicles: A case study of a high-rise office building in Hong Kong. *Energy Conversion and Management* 199 (2019), p. 111888. ISSN: 0196-8904. DOI: https://doi.org/10.1016/j.enconman.2019.111888. URL: https://www.sciencedirect.com/science/article/pii/S0196890419308702.

[16] Peng, C., Zou, J. and Lian, L. Dispatching strategies of electric vehicles participating in frequency regulation on power grid: A review. *Renewable and Sustainable Energy Reviews* 68 (2017), pp. 147–152. ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2016.09.133. URL: https://www.sciencedirect.com/science/article/pii/S1364032116306426.

[17] Zhang, R., Cheng, X. and Yang, L. Flexible Energy Management Protocol for Cooperative EV-to-EV Charging. *IEEE Transactions on Intelligent Transportation Systems* 20.1 (2019), pp. 172–184. DOI: 10.1109/TITS.2018.2807184.

[18] Kang, J., Yu, R., Huang, X., Maharjan, S., Zhang, Y. and Hossain, E. Enabling Localized Peer-to-Peer Electricity Trading Among Plug-in Hybrid Electric Vehicles Using Consortium Blockchains. *IEEE Transactions on Industrial Informatics* 13.6 (2017), pp. 3154–3164. DOI: 10.1109/TII.2017.2709784.

[19] Aznavi, S., Fajri, P., Shadmand, M. B. and Khoshkbar-Sadigh, A. Peer-to-Peer Operation Strategy of PV Equipped Office Buildings and Charging Stations Considering Electric Vehicle Energy Pricing. *IEEE Transactions on Industry Applications* 56.5 (2020), pp. 5848–5857. DOI: 10.1109/TIA.2020.2990585.

[20] CORPORATIVA, I. *Do you know the key points and the possibilities that self-consumption of electricity offers?* Iberdrola. URL: https://www.iberdrola.com/innovation/self-consumption (visited on 03/10/2023).

[21] *What is collective self-consumption?* Energuide. URL: https://www.energuide.be/en/questions-answers/what-is-collective-self-consumption/2140/ (visited on 03/12/2023).

[22] Valta, J., Kulmala, A., Järventausta, P., Björkqvist, T. and Mäkinen, S. Forming Collective Self-Consumption Models: How the end user sees them?: *CIRED 2021 - The 26th International Conference and Exhibition on Electricity Distribution*. Vol. 2021. 2021, pp. 3234–3238. DOI: 10.1049/icp.2021.1733.

[23] Tushar, W., Yuen, C., Saha, T. K., Morstyn, T., Chapman, A. C., Alam, M. J. E., Hanif, S. and Poor, H. V. Peer-to-peer energy systems for connected communities: A review of recent advances and emerging challenges. *Applied Energy* 282.PA (2021). DOI: 10.1016/j.apenergy.2020.1. URL: https://ideas.repec.org/a/eee/appene/v282y2021ipas0306261920315464.html.

[24] *Electrical grid*. *Wikipedia*. Page Version ID: 1119626090. Nov. 2, 2022. URL: https://en.wikipedia.org/w/index.php?title=Electrical_grid&oldid=1119626090 (visited on 11/23/2022).

[25] *The Electrical Grid: Key Concepts | EnergySage*. Mar. 17, 2020. URL: https://www.energysage.com/electricity/understanding-electrical-grid/ (visited on 11/23/2022).

[26] *Smart grid*. *Wikipedia*. Page Version ID: 1117408382. Oct. 21, 2022. URL: https://en.wikipedia.org/w/index.php?title=Smart_grid&oldid=1117408382 (visited on 11/23/2022).

[27] *House or flat: where do you live?* 21st May 2021. URL: https://ec.europa.eu/eurostat/web/products-eurostat-news/-/ddn-20210521-1 (visited on 23/11/2022).

[28] Trenbath, K., Tuttle, R., Maisha, K. and LeBar, A. Assessing and Reducing Plug and Process Loads in Office Buildings. (), p. 9.

[29] King, J. and Perry, C. Smart Buildings: Using Smart Technology to Save Energy in Existing Buildings. *SMART BUILDINGS* (Feb. 2017), p. 55.

[30] Yi, F., Li, F. and Hu, Z. Cost analysis of individual EV charging in different community networks. *2017 IEEE Transportation Electrification Conference and Expo, Asia-Pacific (ITEC Asia-Pacific)*. 2017, pp. 1–7. DOI: 10.1109/ITEC-AP.2017.8080847.

[31] Moghaddass, R., Mohammed, O. A., Skordilis, E. and Asfour, S. Smart Control of Fleets of Electric Vehicles in Smart and Connected Communities. *IEEE Transactions on Smart Grid* 10.6 (2019), pp. 6883–6897. DOI: 10.1109/TSG.2019.2913587.

[32] Gong, H. and Ionel, D. M. Optimization of Aggregated EV Power in Residential Communities with Smart Homes. *2020 IEEE Transportation Electrification Conference Expo (ITEC)*. 2020, pp. 779–782. DOI: 10.1109/ITEC48692.2020.9161532.

[33] *React – A JavaScript library for building user interfaces*. URL: `https://reactjs.org/` (visited on 02/13/2023).

[34] *Renault Zoe*. *Wikipedia*. Page Version ID: 1135145334. Jan. 22, 2023. URL: `https://en.wikipedia.org/w/index.php?title=Renault_Zoe&oldid=1135145334` (visited on 02/13/2023).

[35] *chartjs-plugin-streaming*. chartjs-plugin-streaming. URL: `https://nagix.github.io/chartjs-plugin-streaming` (visited on 02/06/2023).