

Master Thesis

Communication standards for automated charging of electro and plug-in-hybrid vehicles

Carried out at the Institute of Automotive Engineering
Member of [FSI]

Director:
Univ.Prof. Dr.techn. Peter Fischer

Supervisor:
Associate Prof. Dr.techn. Mario Hirz
Dipl.-Ing. Bernhard Walzel

Graz, February 2018

The logo for FTG, with the letters 'FTG' in a bold, blue, italicized sans-serif font.

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am
(Unterschrift)

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from used sources.

.....
(Date) (Signature)

Acknowledge

I want to say thank you to my family for their never ending love, their trust in me and my skills as well as their support, on which I can always count on.

I would also like to thank my friends who have helped me over the last few years.

This master thesis was carried out in cooperation with the FTG.

Therefore, I would like to thank my supervisors Dipl. Ing. Bernhard Walzel and Assoc.Prof. Dipl.-Ing. Dr.techn. Mario Hirz for their help and support during this thesis.

Abstract

Automation and digitalization are two terms that will continue to shape the 21st century. They not only affect the production cycle, also the automobile itself which supports the development of networked driving and alternative drive systems. The thesis focusses on the communication of automated charging system for electric vehicles with cables. The fact that an electric vehicle needs almost 8 hours to charge provides challenges. One way to improve this situation is communicating between an automated charging process and vehicle or vehicle owner. A goal in the future is that a vehicle will independently drive to a charging station and charges, while it is not being used. This can be done by instructions of the driver or automatically. The return of the vehicle could work in the same way. This could save a lot of time. Because there are different communication possibilities, infrastructure, as well as automobile brands with different hardware and software, there has to be a standardization so that all devices can communicate with each other. Therefore, a concept was created that can establish a communications channel, which subsequently would be suitable for standardization regarding automated conductive charging of electric vehicles. Additionally, the reader gets an impression of the software and hardware, which was used for the conception. Furthermore, the important factors for the selection of the software used are specified. However, this thesis also includes the problems and challenges, which were faced and solved during this research. Finally, the conclusion discusses the results of this work.

Kurzfassung

Automatisierung und Digitalisierung haben den Beginn des 21. Jahrhundert geprägt und werden dies noch weiterhin tun. Diese Trends ziehen auch in die Automobilindustrie ein und beeinflussen dadurch nicht nur den Produktionszyklus, sondern auch das Automobil selbst. Dies, und die Tatsache der schwindenden Erdöl-Reserven treiben die Entwicklung von alternativen Antrieben. Auf Grund des möglichen Kraftstoff-Sparpotentials spielt auch die Vernetzung von Infrastruktur, Fahrzeug und Fahrer eine immer wesentlichere Rolle. Durch diese Vernetzung können Algorithmen errechnet werden, die es dem Fahrer ermöglichen ein optimales Ergebnis für sein Fahrzeug herauszuholen. In dieser Arbeit wird speziell auf die Kommunikation beim automatisierten Laden von Elektroautos mit Kabel eingegangen. Die Tatsache, dass ein Elektroauto mehrere Stunden zum Aufladen benötigt, sorgt für Herausforderungen. Ein potentiell Ziel in der Zukunft ist, dass ein Fahrzeug selbstständig zu einer Ladestation fährt und ladet, während es nicht genutzt wird. Dies kann auf Anweisung des Fahrers erfolgen oder automatisiert. Ebenso könnte das Fahrzeug den Weg zurück tätigen. Auf Grund der verschiedenen Möglichkeiten der Kommunikation und Automobilmarken mit unterschiedlicher Hard- und Software müssen Standards geschaffen werden, um eine reibungslose Kommunikation zwischen den verschiedenen Geräten sicherzustellen. Im Zuge dieser Arbeit wurde daher ein Konzept für einen Kommunikationskanal erstellt, welches in weiterer Folge für eine Standardisierung im Bereich der automatisierten induktiven Ladesysteme geeignet wäre. Im weiteren Verlauf dieser Arbeit bekommt der Leser einen Eindruck von der Soft- und Hardware, welche bei den Tests benutzt wurde. Des Weiteren werden Faktoren beschrieben, welche ausschlaggebend für die Wahl der Software waren. Die Arbeit liefert auch einen Einblick auf die Probleme während der Durchführung dieser Arbeit und wie diese gelöst wurden. Abschließend bietet die Zusammenfassung einen Überblick der Ergebnisse.

Table of content

<i>Eidesstattliche Erklärung</i>	<i>ii</i>
<i>Acknowledge</i>	3
<i>Abstract</i>	4
<i>Kurzfassung</i>	5
<i>Table of content</i>	6
<i>Index of abbreviations</i>	8
1 Introduction	9
1.1 Initial situation	10
1.2 Objective target	12
1.3 Approach and Methodology	12
2 Research	14
2.1 V2X networks	14
2.1.1 Communication types, and –standards	14
2.1.2 Communication structures	17
2.2 Programs	19
2.2.1 Matlab R2016b	19
2.2.2 Node.js 8.9.4 LTS	21
2.2.3 Arduino Genuino 1.6.13	22
2.2.4 Halcon XL Version 13.0.1.1	23
2.2.5 ROS Indigo Igloo	24
2.2.6 Summary	25
3 Infrastructure	26
3.1 Hardware	26
3.1.1 Charging station	26
3.1.2 Robotic arm	29
3.1.3 Vehicles	30
3.1.4 Challenges and problems	31
3.2 Software	32
3.2.1 Wi-Fi	32
3.2.2 Challenges and problems	33
4 Concept for communication	35
4.1 Concept for the communication structure	37
4.1.1 Concept of user data	37
4.1.2 Charging station data	38
4.1.3 Robotic arm data	39
4.1.4 Vehicle data	40
4.1.5 Summary	40
4.2 Concept of safety	41
5 Charging Strategy	43
5.1 Communication standards for charging Strategy	44

5.2	Strategy for “smart vehicle” case	46
5.3	Strategy for the “non-smart” vehicle.....	49
5.4	Summary	51
6	<i>Conclusion and Outlook</i>	52
7	<i>Links</i>	53
8	<i>Table of figures</i>	54
9	<i>List of Tables</i>	56
10	<i>Bibliography</i>	57
12	<i>Appendix</i>	61

Index of abbreviations

AC	Alternating current
API	Application Programming Interface
BSM	Basic Safety Messages
CAD	Computer Aided Design
CAN	Controller Area Network
CO ₂	Carbon Dioxide
DC	Direct current
DSRC	Dedicated Short Range Communication
e.g.	<i>exempli gratia</i> , a Latin phrase that means “for the sake of example.”
etc.	et cetera
EV	Electric Vehicle
EVCC	Electric Vehicle Communication Controller
EVSE	Electric Vehicle Supply Equipment
FTG	Institut für Fahrzeugtechnik Graz (Institute of Automotive Engineering Graz)
GFCI	Ground Fault Circuit Interrupter
GPS	Global Positioning System
HLSV	Hessisches Landesamt für Straßen- und Verkehrswesen
ICG	Institute for Computer Graphics and Vision
IEEE	Institute of Electrical and Electronics Engineers
IGLZ	Integrierte Gesamtverkehrsleitzentrale der Stadt Frankfurt am Main
IP-Address	Internet Protocol Address
ITS/C-ITS	Cooperative-Intelligent Transportation System
MIT	Massachusetts Institute of Technology
OBU	On Board Units
OEM	Original Equipment Manufacturer
OSI	Open Systems Interconnection Model
PHY	Physical Layer
RCD	Residual Current Operated Device
RGB	Red Green Blue colour system
RSU	Road Side Units
ROS	Robot Operating System
SECC	Supply Equipment Communication Controller
STL	Standard Triangle Language
TCP	Transmission Control Protocol
TUG	Technical University of Graz
UDP	User Datagram Protocol
UK/GB	United Kingdom/Great Britain
UMTS	Universal Mobile Telecommunication System
US/USA	United States of America
VANET	Vehicular Ad-hoc Networks
VPN	Virtual Private Network
WAVE	Acronym Wireless Access for the Vehicular Environment
WSMP	Wave Short Message Protocol
ZID	Zentraler Informatik Dienst (Central Data Processing Service)

1 Introduction

The automotive industry will be facing drastic changes in the next years. The graph in figure 1-1 shows that the sales numbers for electric vehicles are growing. Many companies are consequently putting effort and money into the development of electric cars. The sales trend is constantly rising, which points to the fact that in the future more customers will buy electric driven cars.

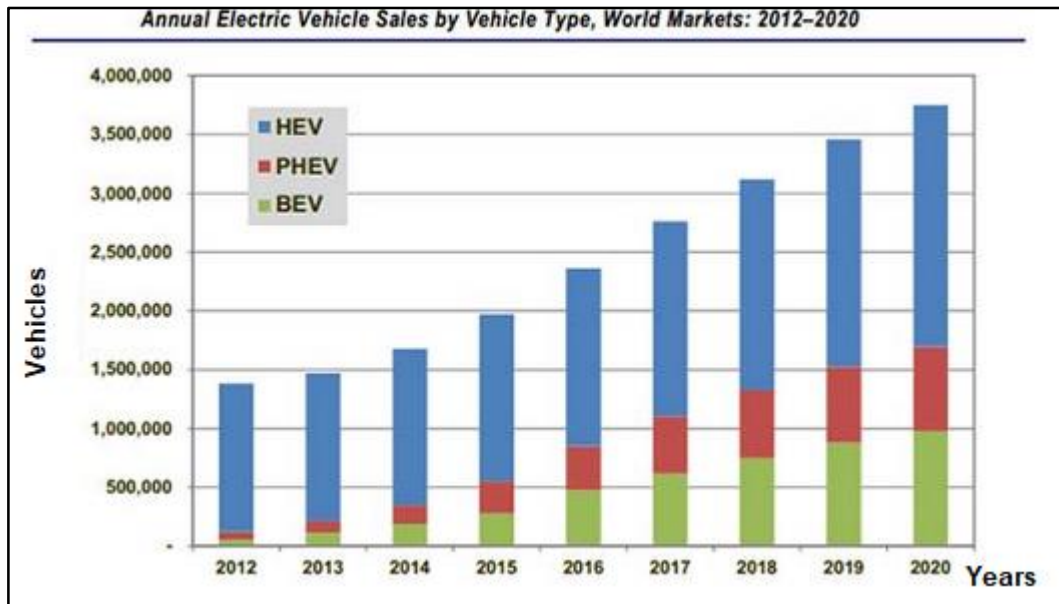


Figure 1-1: Battery electric vehicle sales [1]

However, selling electric cars is not just a matter of alternative power. For an electric driven vehicle, providing infrastructure is a very important issue. Currently, only a handful of countries has a sufficient infrastructure for electric vehicles. A satisfying level of electric infrastructure will not be developed within the next few years.

The needed time is the main difference between refuelling a vehicle with combustion engine and an electric vehicle. While for the gasoline driven car the price per gallon is the most important factor, for the electric driven vehicle it is not only the price, but also charging time. Therefore, an electric vehicle owner has two factors to consider - price and time.

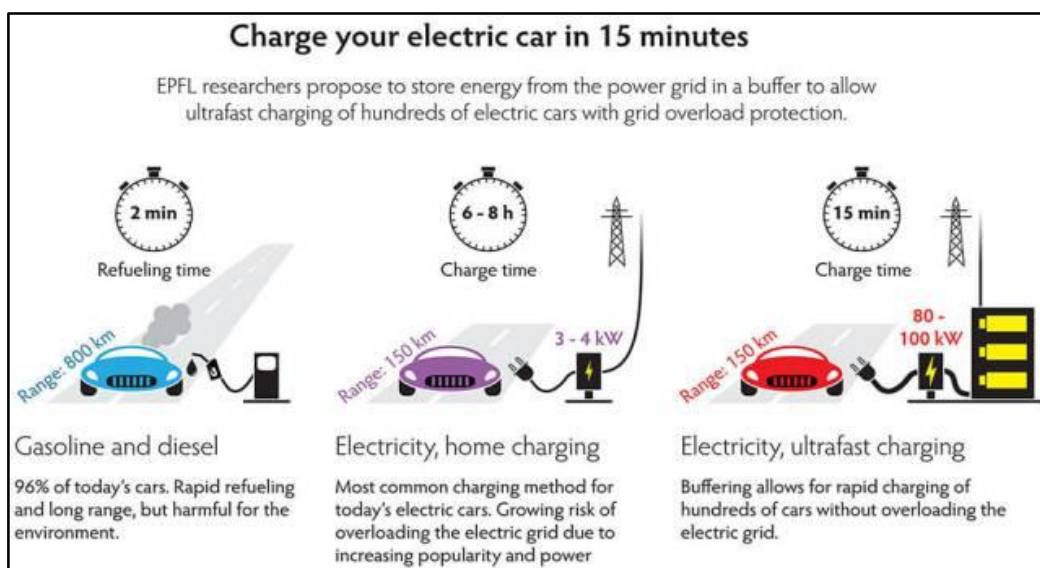


Figure 1-2: Gasoline and diesel refuelling in comparison with electric charging [2]

As we can see in figure 1-2, an average refuelling process requires 2 minutes on average, less compared to the 15 minutes charge time with the rapid charger. This represents a significant difference when the driver needs 7 times more for the recharging process.

Figure 1-3 shows the comparison of the retail price of gasoline and electricity in the last 36 years.

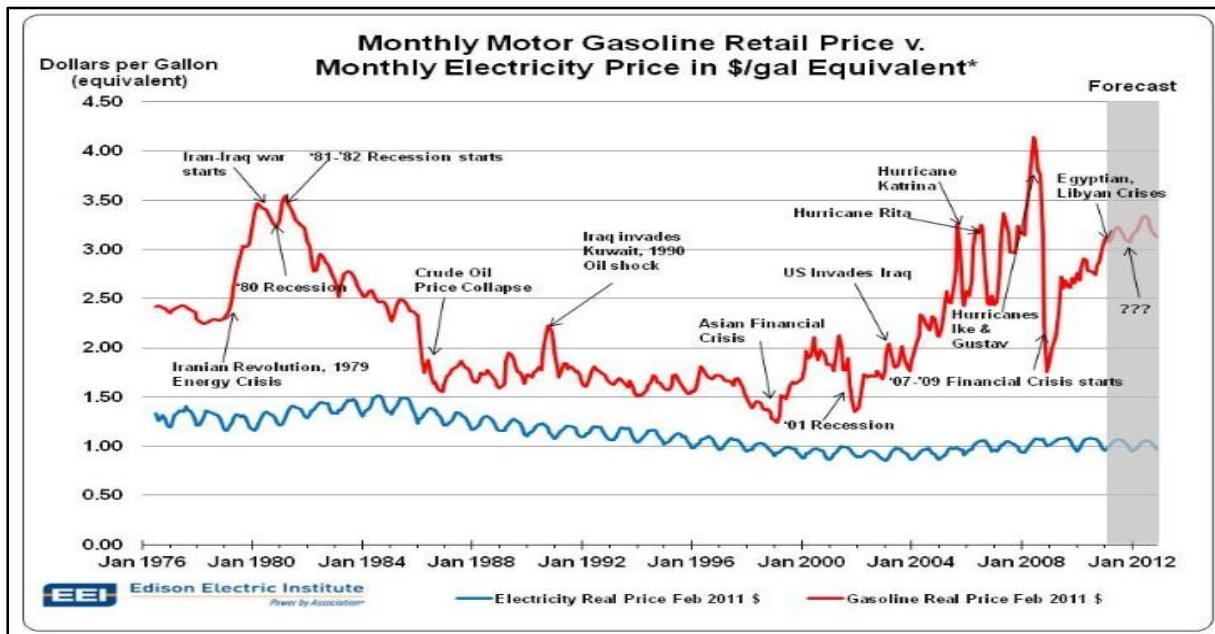


Figure 1-3: Gasoline and electricity price fluctuation [3]

The fluctuation of the electricity price is more stable and is not affected by crises. The oil reserves will not last forever. Therefore, an offer of solutions regarding alternative propulsion systems and a serviceable infrastructure is necessary. Providing these solutions is a big challenge, not only for the entire automotive industry, but also for governments and educational institutions. One of these innovative solutions could be autonomous charging. This solution is not just for solving infrastructure issues; it also supports autonomous driving. The final aim of autonomous charging is that the car drives to a charging station by itself, while the owner is occupied otherwise.

1.1 Initial situation

The goal of this thesis is the development of the communicational part for an automated charging station prototype on the Institute of Automotive Engineering and the formulation of proposals for communication standards. The literature shows various standards and possibilities for communication connections between several devices. This offers many opportunities, but it can also lead to a high complexity and security problems.

The task is to pick the right communication technology from these various options and to give proposals for communication standardization based on existing standards and practical tests. However, before starting with practical tests and the development of standardization proposals there are some basic questions:

- What must the communication of the charging station do?
- What does the surrounding area look like?
- Are other parties ready to implement these standards?

Answering these questions is a crucial step in setting up the objective targets. Therefore, these questions will be answered one by one. The first one is simple to answer. The main task of the communication of the charging station is to transport the commands (e.g. start the process, stop the process, etc.) from the user to the charging station. However, when it comes to the second one, “What does the surrounding area look like?”, things are looking different. In order to be able to set standards, the surroundings have to be as realistic as possible. That means that the practical tests have to be carried out with devices that are available on the market. Figure 1-4 shows a proposal for the communication strategy and the devices for the practical tests.

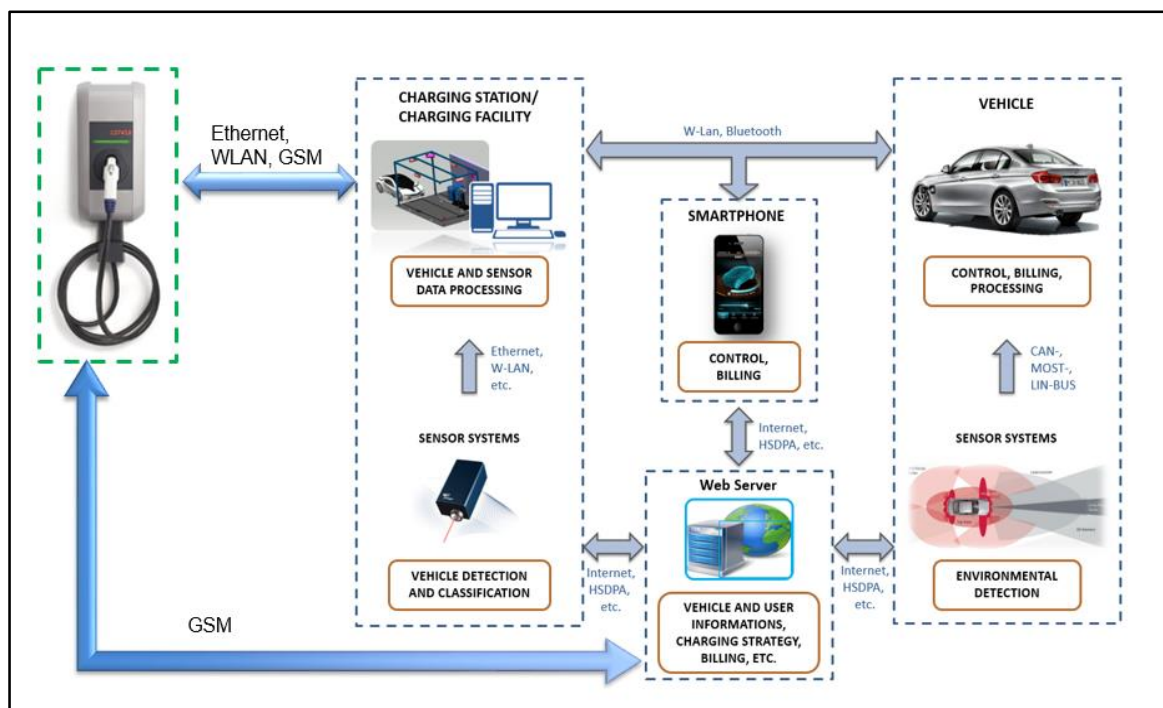


Figure 1-4: Initial situation of the communication strategy [4]

The initial situation therefore is that there are several devices, which need to communicate to each other, tested under realistic surroundings. The used devices has to be available on the market it's communication channels have to be known. Figure 1-4 also shows that the devices already have defined actions like data processing or vehicle detection. The graph shows furthermore potential communication channels, like Bluetooth or Global System for Mobile Communication (GSM).

1.2 Objective target

The main goal of this thesis is to elaborate communication standard proposals for automated conductive charging stations. For this goal, it is important to fulfil the initial conditions and to answer the following questions:

- Which communication standards already exist?
- Which software and hardware can be implemented in the practical tests?

After these questions have been answered, the next step is handling the main task which consists of setting up communication standard proposals based on existing ones and practical tests.

All types of electric vehicles should be able to be automatically charged at grid bound stations. Therefore, a data and communication structure standardization is necessary. The goal of this thesis is to elaborate and draft a standardized communication structure between a robot-based charging station and different types of vehicles. Interests of the various players are playing a major role and have to be considered. Only if these interests are fulfilled, the standardization proposals will be able to be implemented. Therefore, we need to define frame conditions, which are representing these interests:

- The communication has to run on various platforms (e.g. Windows, Linux, Mac, etc.).
- The communication has to be very stable.
- For the automated charging process, the vehicles should stay as they are, without construction or design changes.
- The programs should be written as simple as possible. Program changes or continuing on this work should be possible.
- The programs should provide a good connectivity between each other.
- Unnecessary instructions or downloads should be avoided.
- The communication has to run on programs with low licence costs.

The objective target therefore is to set communication standards that fulfil these frame conditions.

1.3 Approach and Methodology

The first step is to gather as much information as possible. Therefore, a literature research is carried out to gain knowledge about electric vehicle charging and vehicle-to-infrastructure (V2X) communication. The approach to solve the problem is to define different tasks. The first task is to create communication channels between the prototype station devices. Once the communication is established, the second task is to steer and control the devices independently from each other. For example, that a mobile phone acts as a control panel for the charging station. In the next task a user-friendly control unit that enables users an easy steering of the charging station is created. In order to successfully implement this, the control unit software and the design needs to be flexible. That means that the program has to have

the ability to work on various platforms, like Windows, Linux, Apple or Android. The next step is to set up a strategy for the entire communication. The last task includes standardization proposals for the communication.

The methodology for reaching the goals is called selection process. The figure 1-5 shows the steps to reach the goals.

The steps consist of the questions, which needs to be answered combined with the frame conditions. A step can only be concluded if the frame conditions influenced the answer properly.

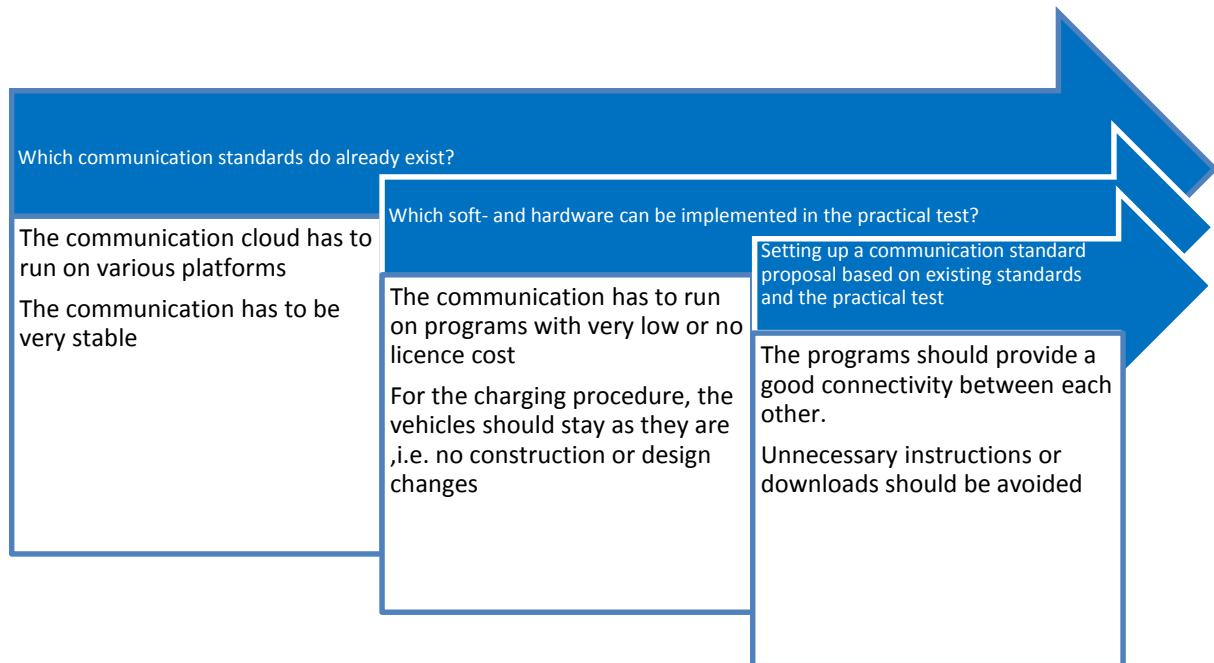


Figure 1-5: Steps of selection

2 Research

This part of the thesis deals with the research of V2X networks and communication. The research focusses on the communication structure and the existing standards.

2.1 V2X networks

V2X networks open new possibilities for traffic efficiency and traffic security. These networks have the potential not just to lower the traffic jam rate, but also help the environment by calculating the optimal trip path with the minimal use of fuel or electrical power. Tools like Bluetooth, Global Positioning System (GPS) or Universal Mobile Telecommunication System (UMTS) use wireless communication. These tools are used to communicate with other vehicles and the infrastructure. [5]

2.1.1 Communication types, and standards

The V2X communication consists of many variants. One type for example works with Wireless Local Area Network (WLAN), another with Universal Mobile Telecommunication System (UMTS). Communication via Bluetooth or infrared is also possible, but the thesis focusses on the first two options. Because V2X communication is important in future, many countries have carried out some field tests. This leads to different standards in this field. Currently, there are two standards that have been defined. These two standards have been implemented in the USA and in Europe. However, there is also an international standard for vehicle to grid communication, which will be mentioned as well.

USA-standard

The focus of V2X communication in the USA is traffic safety and traffic efficiency. The standardization for the US-specific protocol WAVE (Wireless Access for the Vehicular Environment) was carried out by the Institute of Electrical and Electronics Engineers (IEEE). The IEEE 802.11p standard represents the basis of the V2X communication and is the fundament of the European standard TC-ITS G5 as well. Table 2-1 shows the comparison between IEEE 802.11a and IEEE 802.11p standard. [5]

The data rate of the “p” extension is, for example, half of the size than the “a” extension, while the guard time is double that amount. All differences can be seen at the “changes” column.

Parameters	IEEE 802.11a	IEEE 802.11p	Changes
Data rate (Mb/s)	6, 9, 12, 18, 24, 36, 48, 54	3, 4.5, 6, 9, 12, 18, 24, 27	Half
Modulation mode	BPSK, QPSK, 16QAM, 64QAM	BPSK, QPSK, 16QAM, 64QAM	No Change
Code rate	1/2, 2/3, 3/4	1/2, 2/3, 3/4	No Change
Number of subcarriers	52	52	No Change
Symbol duration	4 ms	8 ms	Double
Guard time	0.8 ms	1.6 ms	Double
FFT period	3.2 ms	6.4 ms	Double
Preamble duration	16 ms	32 ms	Double
Subcarrier spacing	0.3125 MHz	0.15625 MHz	Half

Table 2-1: Comparison of PHYs implementations in IEEE 802.11a and IEEE 802.11p [6]

WLAN standards are almost exclusively 802.11 standard. To have compatibility between the equipment of different manufacturers, the Wireless Ethernet Compatibility Alliance (Wi-Fi Alliance) has to be certificated. [7]

Building on the IEEE 802.11p standard, the overlying layers are based on the IEEE 1609.x protocol family. The SAE-J2735 defines the Basic Safety Messages (BSM). The BSMs are messages from a vehicle that contains security relevant information for other road users. [8]

Although the focus of this thesis is on European standards, the standards in the US must still be considered. Figure 2-1 shows the standards of V2X communication in the USA.

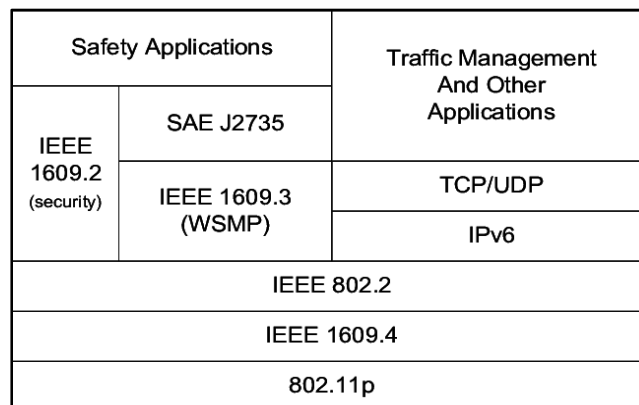


Figure 2-1: Two DSRC protocol stacks, WSMP (left) and TCP/IPv6 (right) [9]

The figure above illustrates the Dedicated Short Range Communication (DSRC) protocol stacks, with the WAVE Short Message Protocol (WSMP), for security relevant communication on the left, and the Transmission Control Protocol/Internet Protocol (TCP/IP), for non-security communication, such as infotainment for example, on the right.

Europe-standard

The focus in Europe lies, in contrast to the US, on infrastructure-less communication. The Car2Car communication consortium and the ETSI are working on the European C2X specification, called Cooperative Intelligent Transportation System (ITS/C-ITS). However, these standards are also considering infrastructure-based communication. [5]

This standard has to be considered in order to make further proposals for standardization regarding automated charging. Therefore, the most important C-ITS layer of the Europe standard is the "Networking and transport" standard as shown in figure 2-2 below.

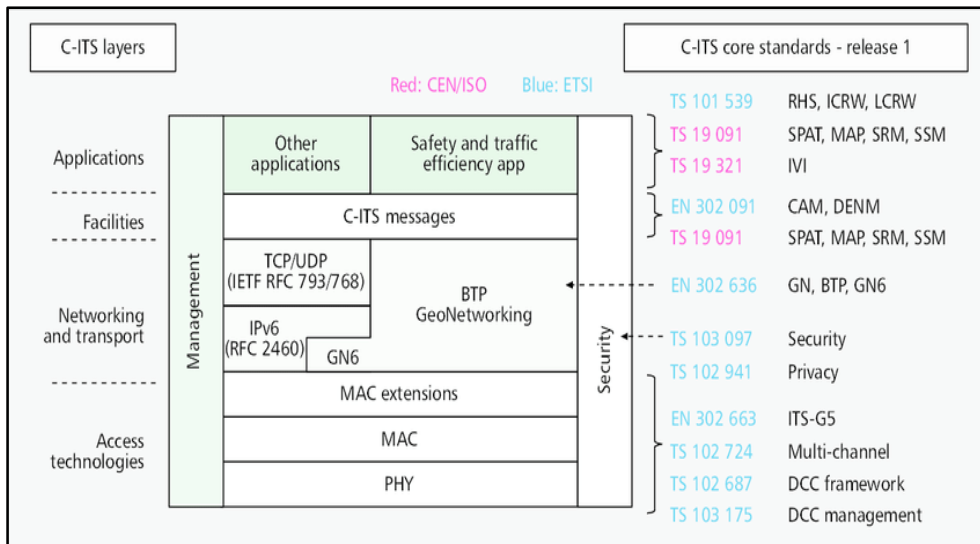


Figure 2-2: Protocol stack and release 1 core standards for C-ITS in Europe [10]

The figure above shows the C-ITS standard compared to the Open Systems Interconnection Model (OSI), which is illustrated left. The right side shows the defined communication from the European Telecommunications Standards Institute (ETSI). For example the number EN 302 663 is a standard number for the profile standard on ITS-G5.

International-standard

The international standard ISO/IEC 15118 defines the vehicle to grid communication and has been fully adopted in spring 2014. It specifies the communication standard between electric vehicle (EV) and the electric vehicle supply equipment (EVSE). Figure 2-3 shows the software modules of this standard.

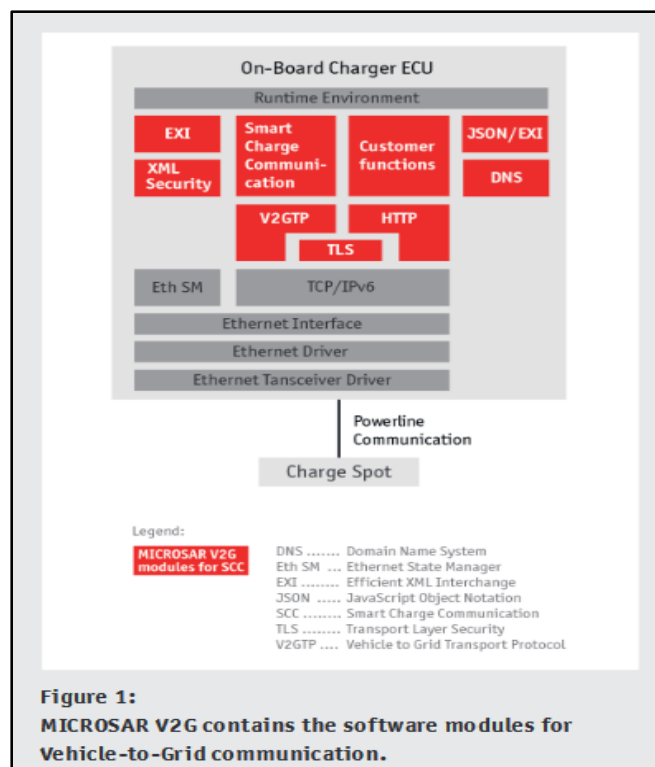


Figure 2-3: Software modules for V2G communication [11]

These standards are the framework for this thesis and the consideration of these norms is necessary in order to give further proposals. The ISO 15118 consists of eight parts.

- Part 1: General information and use-case definition
- Part 2: Network and application protocol requirements
- Part 3: Physical and data link layer requirements
- Part 4: Network and application protocol conformance test
- Part 5: Physical and data link layer conformance test
- Part 6: General information and use-case definition for wireless communication
- Part 7: Network and application protocol requirements for wireless communication
- Part 8: Physical layer and data link layer requirements for wireless communication

Parts one, two and three have already been developed as of 2014. While the other parts are still under development. [12]

However, the parts that are still being developed need to be considered in order to give communication standard proposals.

2.1.2 Communication structures

In this section of the thesis, possible communication structures will be considered. The existing structures are based on the communication standards that have been discussed before. However, it is essential to have knowledge of the structures in order to implement the proposals in one of these.

On Board Unit (OBU)

The interface between driver, vehicle and environment is called OBU. The wireless and bus communication take place on the OBUs. In addition, the hardware platform and the software functionality of the C2X applications are also part of the OBU. Functionality of the OBU depends on two separate platforms, router and host. Router enables wireless communication over different standards like IEEE 802.11p, 802.11b/g and UMTS. To have a better overview the different standards are listed below. [5]

- IEEE 802.11p: Car-to-Car and Car-to-RSU communication.
- IEEE 802.11b/g: Specified for common WLAN in entertainment electronics, serve as performance comparison accord to 802.11p.
- UMTS: Serving as data transfer of security relevant messages across big distances.
- CAN (Controller Area Network): Router needs intelligent CAN-driver to process different CAN-data from different vehicles. Therefore, an interface standardized for programming was used. This enables the usage of Application Programming Interface (API).

The router communicates with the host over ethernet. Software functionality like navigation and C2X applications are already implemented in the host in form of JAVA-applications [5].

Road Side Unit (RSU)

The connection between vehicle and traffic centre is called Road Side Unit. The OBU system architecture is based on the network-, routing- and transport-layer of the Open Systems Interconnection Model (OSI). The IEEE 802.11p standard, in direction to the traffic centre, carries out the communication in direction to vehicles. The transport medium on TCP/IP communication is carried out by stationary RSU glass fibre respectively xDSL. Mobile RSUs use UMTS. [13]

ITS Central Station (ICS)

The connected traffic centres have different tasks; for example in Germany, the IGLZ monitors the inner-city traffic and is able to steer the traffic flow. The HLSV is responsible for the motorways and highways. The outcome is saved in a data bank and transmitted through RSUs. The evaluation of the traffic situation is only possible, when the IGLZ and HLSV are exchanging their data. The OBU can send the requested data to the OBU and can give the driver an overview of the current traffic situation on his route and suggests alternative roads. [5] Therefore, the ICS could send information of the traffic flow close to the charging stations.

2.2 Programs

This chapter describes the software in this project. Furthermore, it will explain the reason why the software programs have been selected and what kind of programs they are.

2.2.1 Matlab R2016b

Matlab [R2016b] is an extensive software package for numerical Mathematics, stands for “MATrix LABoratory” and is widely used. However, although it is very popular and offers many possibilities, the use of the program was not a primary target in this project.

At project start, it was planned to use Matlab as platform for the communication. A big disadvantage of Matlab is the need for additional licences for toolboxes for the TCP communication. For communication with mobile devices such as cell mobile phone or tablet, a Matlab app is needed. However, the program was used later due to the many possibilities in the field of robot control. In order to use the software for the charging station prototype, two network connections at the same time are needed. One connection is linked with the network of the TU-Graz via Virtual Private Network (VPN) -client in order to get the Matlab licence. The second connection is needed to access to the local network of the charging station prototype, in order to communicate with the devices. In this local network, the entire charging station process was tested, for example, data exchange between smartphone and Matlab. Besides the licence requirements and costs, other problems occurred as well. For example, the transmission of data with other software was more difficult than expected. If one program like HALCON (chapter 2.2.4) sends string-based data, Matlab gives it out as numbers.

Figure 2-4 shows the Matlab code for the TCP communication between Matlab and HALCON.

```
%% SendDataToHalcon
%% Create a TCP/IP Connection

IP Address of the Device
IP = '192.168.1.6';
Socket_conn = tcpip(IP, 7090, 'NetworkRole', 'client');
%%
fopen(Socket_conn);
disp('Connected!');

%%

%data=int16(0);
%data5= double(0);
fwrite(Socket_conn, data);
disp('sendet#')

%%
b=fread(Socket_conn);
a=1;

Output of the transferred data
(Coordinates from Halcon)
```

Figure 2-4: Matlab code for the TCP communication

Figure 2-4 shows the Matlab code for TCP communication and the required data input, in order to communicate with other programs. The figure below shows that the Matlab output interprets the HALCON data, which were sent in string format in Matlab double format.

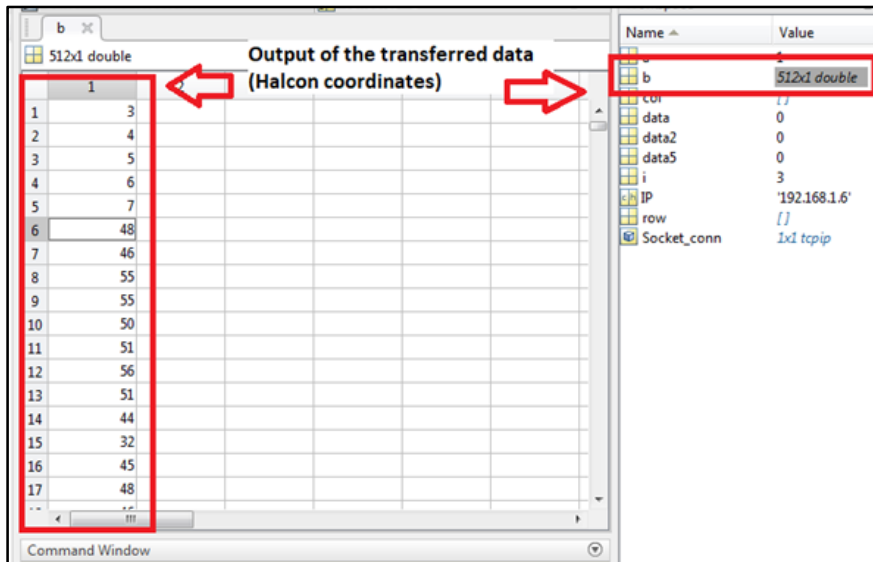


Figure 2-5: Matlab output of transferred data

This creates a major issue, because the sent and received data are not the same. The goal is, that when Matlab receives the coordinates from HALCON, to transform these coordinates in Matlab for the robot control.

However, Matlab offers transformation functions that solves this issues. One function is called "str2num". The principle is that the HALCON data is translated in a character, which is further translated with the command "string2num" from a string to a number. The figure below illustrates this problem.

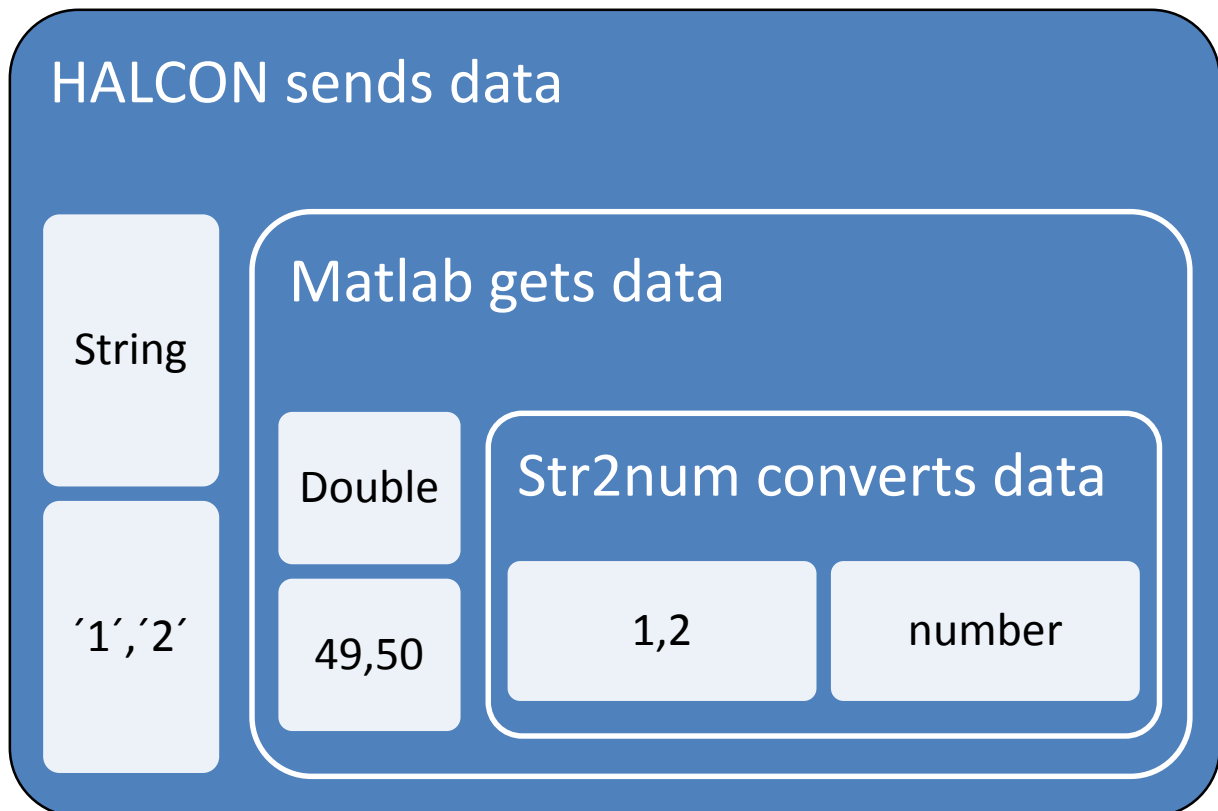
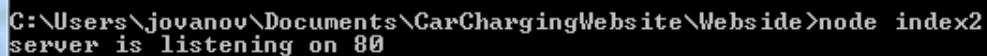


Figure 2-6: HALCON, Matlab data problem

2.2.2 Node.js 8.9.4 LTS

Node.js [8.9.4 LTS] is the program with which the webserver-interface was written. The webserver-interface is used for the connection between the user's mobile phone or a vehicle integrated control panel and the automated charging station. Figure 1-4 in chapter 1.1 shows the units which need to be connected. The webserver-interface connects smartphone, vehicle, wall-box and charging station. Node.js is an open source software for server frameworks. It runs on various platforms such as Microsoft, Linux and is freely available, which makes it a perfect tool for the webserver. The program language is mainly JavaScript. These factors explain why Node.js was used for the webserver and no other programs, like for example Matlab.

For the project, the code for node.js was written in Sublime Build 3143, which is a sophisticated text editor. The code was executed in the downloaded node.js command window. However, it is very important that the npm (node package manager) has to be downloaded as well. Without npm the program cannot run. The program can be activated with the command "node". For starting the webserver, go to the data path where the program is saved via the command "cd". Start the program or programs with the command "node program name".



```
C:\Users\jovanov\Documents\CarChargingWebsite\Webside>node index2
server is listening on 80
```

Figure 2-7: Screenshot "starting the webserver"

The answer line in figure 2-7 shows that the webserver is running and listening for messages on port 80. The User is now able to steer the automated charging process via this homepage. It has to be mentioned, that the code from node.js can also be carried out by the windows command window (cmd.exe). Another advantage of node.js is parallel running of different programs, which is very important for the required topics.

The webserver itself consists of many subprograms that create the design and functionality of the homepage. Furthermore, further programs have to be written in node.js as well, in order to communicate with other software. Two program scripts are needed for communication. The "udp_server" receives the messages from other devices or software, while the task of "udp_client" is mainly to send messages. The website itself runs with the script code known as "Index2". However, this code consists, as mentioned before, of sub-codes. In addition, the difference between subprograms for the website and programs for the communication must be highlighted. The structure of these subprograms can be seen in figure 2-7. The programs "sendChargeMessage" and "sendTCP" are for the communication between the devices. While "sendTCP" is responsible for the TCP messages, the "sendChargeMessage" code is there to send messages in UDP format. The communication to the devices is additionally added to the homepage and does not influence the page itself.

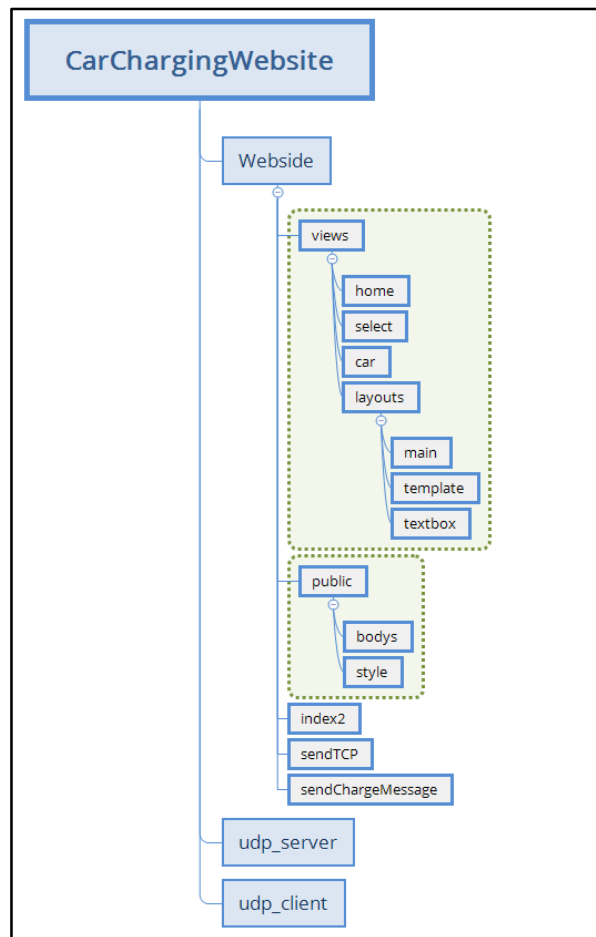


Figure 2-8: Folder structure

The subprograms define the functionality and the design of the homepage. These sub-codes are in the folders “views” and “public”. The folder “views” contains also the programs “car”, “home” and “select”. These codes are defining the various pages of the homepage. Here it can be defined how many pages are required and which functionality they should have.

The folder “layouts” is a subfolder, which contains the programs “main”, “template” and “textbox”. All of these codes define colour, design, text size and aesthetics of the homepage. With this construct, the user-interface of the homepage is defined.

2.2.3 Arduino Genuino 1.6.13

“Arduino [Genuino 1.6.13] consists of both a physical programmable circuit board (often referred to as a microcontroller) and a piece of software, or IDE (Integrated Development Environment) that runs on your computer, used to write and upload computer code to the physical board.” [14]

Almost all Arduino boards are providing digital I/O-Pins (Input/Output-Pins) of the microcontroller for utilization of electric circuits. The microcontroller is steered by the IDE, which runs on various platforms. Arduino is therefore a perfect tool for this project. It is an open platform and can be easily implemented. For testing the Arduino, the first program was written in C#-script in order to steer the actuator. The Actuator is necessary in order to open and close the charging socket cover. Fortunately, the program worked fine and it was possible to steer the Actuator with the user-interface via C#. Later on, the program was written in Matlab. The content of the steering program in C# can be seen in the appendix of this thesis.

Arduino always uses the same principle: define the variables and the command that triggers the variables. For example, if Arduino gets a UDP message from a device with the message 1, the variable, which is defined for this number, fulfils the task.

2.2.4 Halcon XL Version 13.0.1.1

HALCON [XL Version 13.0.1.1] is primarily a bibliotheca of programs with algorithms for image processing from cameras and it is one of the few licence-bound programs included in this thesis. It has to be mentioned that open source image processing software is generally not easy to find. Furthermore, this part of software is very crucial because the cameras are very important for the position detection of the car and positioning and moving the robot arm. This leads to the conclusion that this software must be very advanced. Poor picture processing can lead to major issues. HALCON has also pre-defined packages and data-formats for the communication with other software. The figure below shows the HALCON program window.

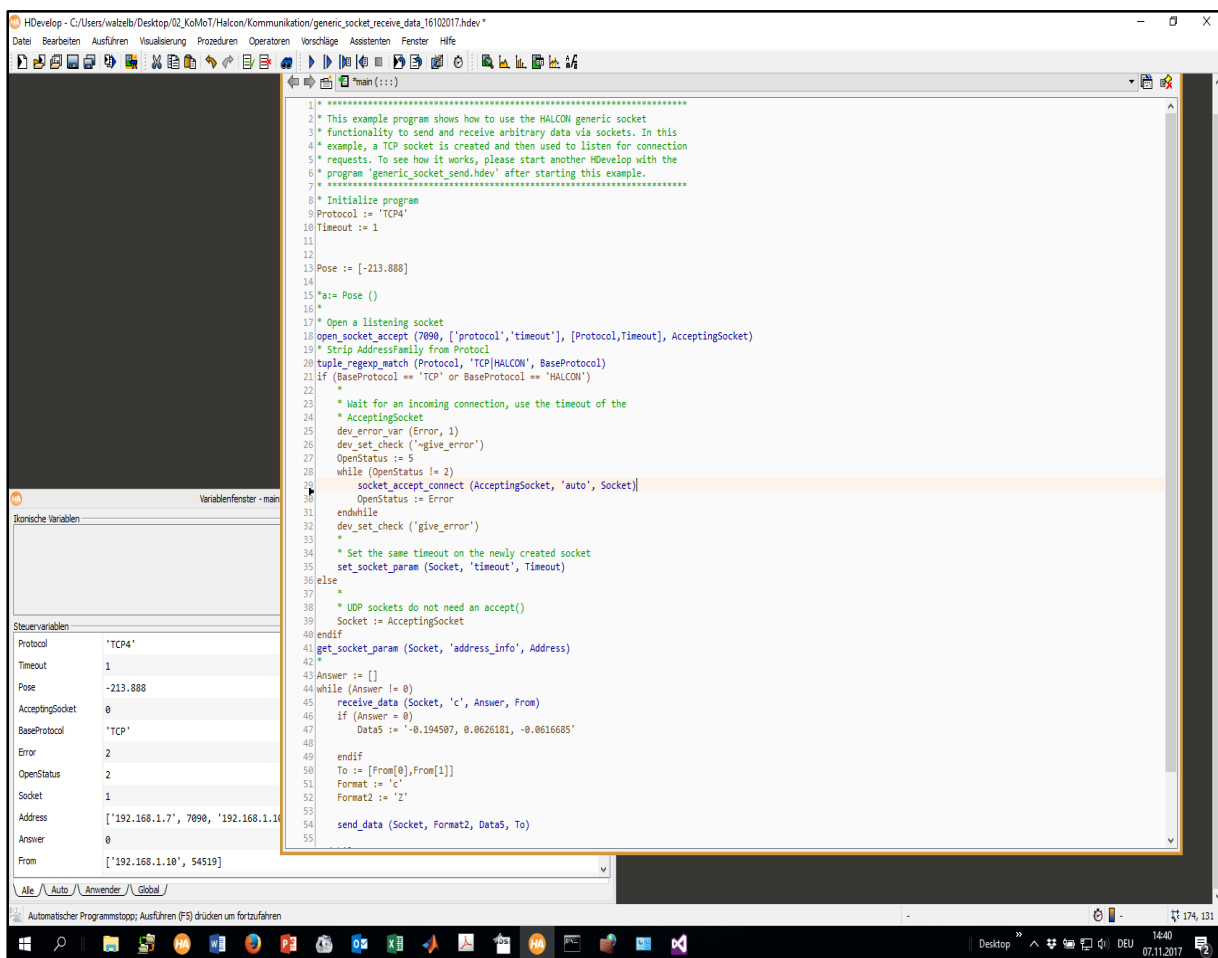


Figure 2-9: Halcon program window

The next figure shows the code for the communication with the TCP protocol. As mentioned before, HALCON has pre-defined packages and the "Socket-Program" is one of them. With this package, a TCP connection can be established and data can be sent to other devices. Therefore, HALCON is responsible for the image processing of the cameras, which is crucial for the robot arm positioning.

```

1 * *****
2 * This example program shows how to use the HALCON generic socket
3 * functionality to send and receive arbitrary data via sockets. In this
4 * example, a TCP socket is created and then used to listen for connection
5 * requests. To see how it works, please start another HDevelop with the
6 * program 'generic_socket_send.hdev' after starting this example.
7 * *****
8 * Initialize program
9 Protocol := 'TCP4'
10 Timeout := 1
11
12
13 Pose := [-213.888]
14
15 *a:= Pose ()
16 *
17 * Open a listening socket
18 open_socket_accept (7090, ['protocol','timeout'], [Protocol,Timeout], AcceptingSocket)
19 * Strip AddressFamily from Protocol
20 tuple_regexp_match (Protocol, 'TCP|HALCON', BaseProtocol)
21 if (BaseProtocol == 'TCP' or BaseProtocol == 'HALCON')
22 *
23 * Wait for an incoming connection, use the timeout of the
24 * AcceptingSocket
25 dev_error_var (Error, 1)
26 dev_set_check ('~give_error')
27 OpenStatus := 5
28 while (OpenStatus != 2)
29     socket_accept_connect (AcceptingSocket, 'auto', Socket)
30     OpenStatus := Error
31 endwhile
32 dev_set_check ('give_error')
33 *
34 * Set the same timeout on the newly created socket
35 set_socket_param (Socket, 'timeout', Timeout)
36 else
37 *
38 * UDP sockets do not need an accept()
39 Socket := AcceptingSocket
40 endif
41 get_socket_param (Socket, 'address_info', Address)
42 *
43 Answer := []
44 while (Answer != 0)
45     receive_data (Socket, 'c', Answer, From)
46     if (Answer = 0)
47         Data5 := '-0.194507, 0.0626181, -0.0616685'
48
49         endif
50         To := [From[0],From[1]]
51         Format := 'c'
52         Format2 := 'Z'
53
54         send_data (Socket, Format2, Data5, To)
55

```

Figure 2-10: TCP-protocol for Halcon

2.2.5 ROS Indigo Igloo

The Robot Operating System (ROS) [Indigo Igloo] is a software framework for robot control. It is an open source program and it fits to the conditions of usage for this thesis. However, during the tests it was discovered that this software was not as user friendly as expected. Many problems occurred during the code writing and especially during the download of the ROS packages. In addition, packages, which should be downloaded and updated without any troubles could not perform as they should. These and other issues led to the consequence that the robot control was realized with Matlab.

Although this framework does not have licence costs, the main problem is that even ROS professionals have troubles with this software. Despite consultation with an ROS expert, the struggles with this program could not be solved. It is recommended that in case of using this framework, involving a ROS-expert is necessary.

2.2.6 Summary

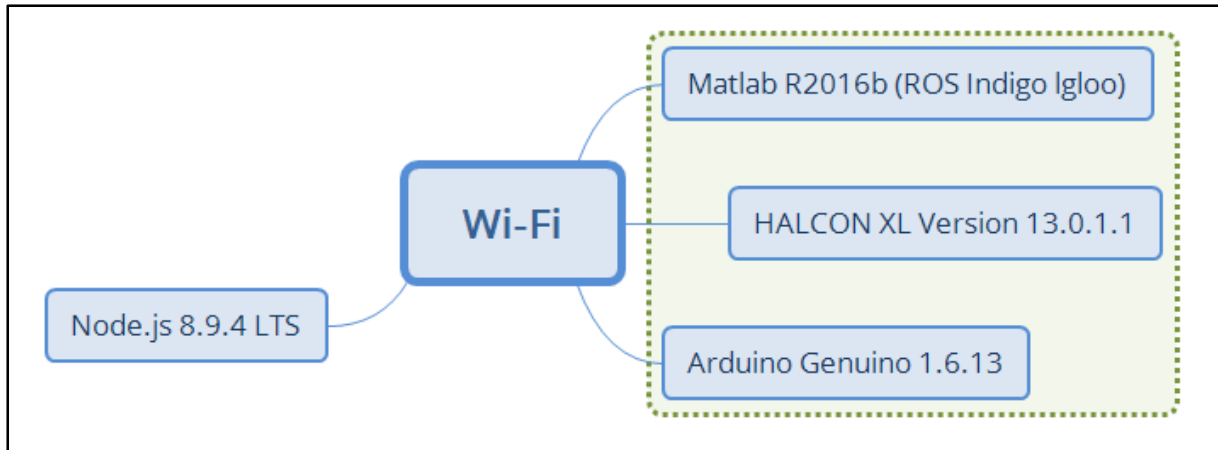


Figure 2-11: Program formation

Figure 2-11 illustrates how the programs were used in the charging station system. The right side shows the charging station programs, such as HALCON, Arduino and Matlab. The left side shows the homepage, which was executed with Node.js. The green box illustrates the internal communication between these programs, while the programs outside of this box, such as Node.js, illustrates the external communication. Furthermore, the green box symbolizes the charging station, while Node.js is responsible for the user data. The exact amount of data transmission and communication is handled later in this thesis in chapter four and five.

All programs communicated with each other and were implemented in the field test and furthermore in the charging strategy. It has to be mentioned that Matlab R2016b does not necessary fulfil the selection criteria from the methodology, however the program was used to bypass the ROS problem, which is described in detail in chapter 3.2.2.

3 Infrastructure

This chapter shows the infrastructure, which was used for the prototype tests. It is very important to split the infrastructure into basic elements. In this chapter the infrastructure is divided into hardware and software.

3.1 Hardware

This chapter describes the hardware used for the project. It clarifies which factors and requirements are important in order to guarantee the functionality. Furthermore, the necessary specifications will be mentioned as well, in order to standardize the communication.

3.1.1 Charging station

The charging station is an essential element for the charging task. The station does not just provide the vehicle with electricity, but also maintains and steers the charging power. There are two possibilities to charge an electric vehicle; conductive and inductive. First topic is the conductive charging. This technique needs to fulfil the general requirements, which are defined in the DIN EN 61851 standard.

According to the literature, tractive and torsional forces is necessary of the charging cable should be avoided. To change the alternating current (AC) from the public grid to direct current (DC) for the vehicle, there has to be a charger. In electric-bikes and scooters, the charger is mostly off-board, whereas in other vehicles it is on-board. [15]

“Vehicles that are loaded with an off-board charger have usually a DC-connection. Vehicles with an on-board charger an AC-connection. Partly vehicles have additionally a DC-rapid connector.” [15]

There are different charging modes because every charging possibility has to be considered. There is a difference if a car is charged at a public or private location. Furthermore, the type of the electrical source is also important.

Mode 1: *“Home charging from a standard power outlet with a simple extension cord, without any safety measures. Although this is what many (private) EV conversions use today, “Mode 1” has been outlawed in several countries and Australia is likely to follow”.*

Mode 2: *“Home charging from a standard power outlet, but with a special in-cable EVSE (EV Supply Equipment), aka “occasional use cable”, usually supplied with an EV from the manufacturer.*

Mode 3: *“Wired-in AC charging station, either in public places or at home, allowing a higher power level than Mode 2. The safety protocol is identical to Mode 2.”*

Mode 4: *“Wired-in DC charging station, either in public places or at home. In DC charging stations, the charger is part of the charging station, not part of the car”.*

*“The standard charging system for wire-conducted charging of electric and plug-in hybrid vehicles is the **Combined Charging System (CCS)**, which is mandatory for all future fast charging systems in the European Union.”* [15]



Figure 3-1: Charging systems [16]

The figure above shows the different charging systems, which are available now. Stations, which are designed to charge a battery of an electric vehicle delivering an electric output ranging between 50 kW – 120 kW are called fast charging systems. [17]

Therefore, the thesis focuses on the charging modes 3 and 4. These modes can be reached with the Combined Charging System (CCS), which is illustrated in figure 3-1 (Combined AC/DC charging system/Type 2) and 3-2 (Vehicle inlet/System C, Combo 2). At the Hannover trade show the “PHOENIX connector” (Combo 2) with an integrated cooling system for the 350 kW, the so-called CCS plus was displayed.

CHAdeMO is the Japanese standard, which in 2017 currently offers a charging station with 150 kW. The connector for the 150 kW will stay the same. If the market demands it, Japan plans to increase the power to 350 kW in 2018.

	System A CHAdeMO (Japan)	System B CATARC (PRC)	COMBO1 (US; System C)	COMBO2
Connector				
Vehicle Inlet				
Communication Protocol	CAN		PLC	

Figure 3-2: Communication protocols for charging systems [18]

Figure 3-2 shows furthermore the communication protocols for the different connector standards. The figure shows mainly two protocols, CAN (Controller Area Network) and PLC (Programmable Logic Controller). The CAN system is a vehicle bus standard, which allows electronic control units and devices to communicate with each other without using a host computer [19].

The PLC controls a wide array of applications. These systems perform many functions, including various communication protocols and the PLC’s are programmed for specific tasks. [20]

Both protocols manage the communication between Electric Vehicle (EV) and EVSE (Electric Vehicle Supply Equipment). The second possibility is charging via electromagnetic induction. This technique has to fulfil the DIN EN 61980-1. [21]




	Actual Wire System	Actual Inductive System with 100 mm gap	FastInCharge System with 100 mm gap
Type:			
Efficiency	90-94%	>80%	87-92%
Power rate	50kW	3kW	40kW
Charging time for 90% of battery:	20-30 min	6-8 hours	~ 30min(if stationary)
Weight Added of Vehicles	/	<4.5 kg	~ 20kg

Figure 3-3: Inductive charging [22]

Contact charging is more efficient and therefore cheaper for charging, while the inductive charging is safer and no cables are required. However, to have an efficient charging procedure, the inductive charging requires a special spot, which depends on the vehicle. Since inductive charging is inconvenient the conclusion is that, this way of charging is not an option. [23]

The charging station used for the project was the KEBA KeKontakt Master P30. [24]

The P30 is a smart charging station, which creates the possibility of communication with other devices via UDP-protocols. The commands are already defined and had to be implemented in the communication strategy of the prototype. The charging station has to fulfil the guideline for charging infrastructure. These requirements are important. Otherwise, the risk of dis-functionality increases drastically.

Guideline requirements:

The first thing which has to be taken into consideration is the connector standard. There are mainly two elements, which are important for the safety conditions of a socket. Therefore, before a charging station goes into operation, the thermal and mechanical situation have to be checked (e.g. type of socket, for example type 63 is better than type 13). After fulfilling the socket or connector standards, the next step is the installation of the charging station.

Each vehicle has to have a ground fault circuit interrupter (GFCI common name in US) or residual current operated device (RCD common name in UK). A certificated worker should do the installations. It is recommended that every new- or reconstruction has to be built by an empty conduit 2 x M25, in public space M80. An expert should then check the installations. The suggestive height for the socket is about 130 cm from the ground. The connector should be close to the vehicle. Pavements between vehicle and socket as well as tractive and torsional forces at the connector have to be avoided. Due to the possibility of overheating it is important to prevent cable reels. [15]

“Note that it is important that the placement of the charging station in an existing accessible parking space should allow adequate space (minimum of 36 inches or 914,4 mm) for a wheelchair to pass the vehicle wheel stop” [25].

3.1.2 Robotic arm

This section considers the robotic arm. The robotic arm cannot run by itself. Therefore, there are additional elements, like control box, switchboard, cables and software, which the robot needs in order to run properly. The robot was steered by the Matlab program. However, the robot itself had additionally a control panel with which the robot was steered manually. For this project the Universal Robot UR10-CB3 [26] was used. Knowledge of the crucial parameters is essential if the functionality of the robot is fully given. Table 3-1 shows the crucial parameters.

UR 10 technical specifications	
Weight	28,9 kg
Maximal load	10 kg
Range	1300 mm
Rotation of joints	+/- 360° on all joints
Speed	foot and shoulder joint: 120°/S. Elbow and hand joint 1-3: 180°/s
Communication	TCP/IP 100 Mbit: IEEE 802.3u, 100 BASE-TX
Footprint	Ø 190 mm
Number of joints	6 hinges
Switchboard size	475 mm x 423 mm x 268 mm (WxHxD)
Material	Aluminium, PP plastic
Power consumption	ca. 350 Watt
Power supply	100-240 VAC, 50-60 Hz
Temperature	0-50 °C

Table 3-1: Technical specification for UR10-CB3 [27]

It is important that the robot fulfil the safety requirements for industrial robots ISO-EN 10218. As it is illustrated in the communication column, the robot arm uses the TCP-format for communication. The parameters are showing that the vehicle is not allowed to park further than 1300 mm and that the charging cable together with the robotic arm tools should not weight more than 10 kg.

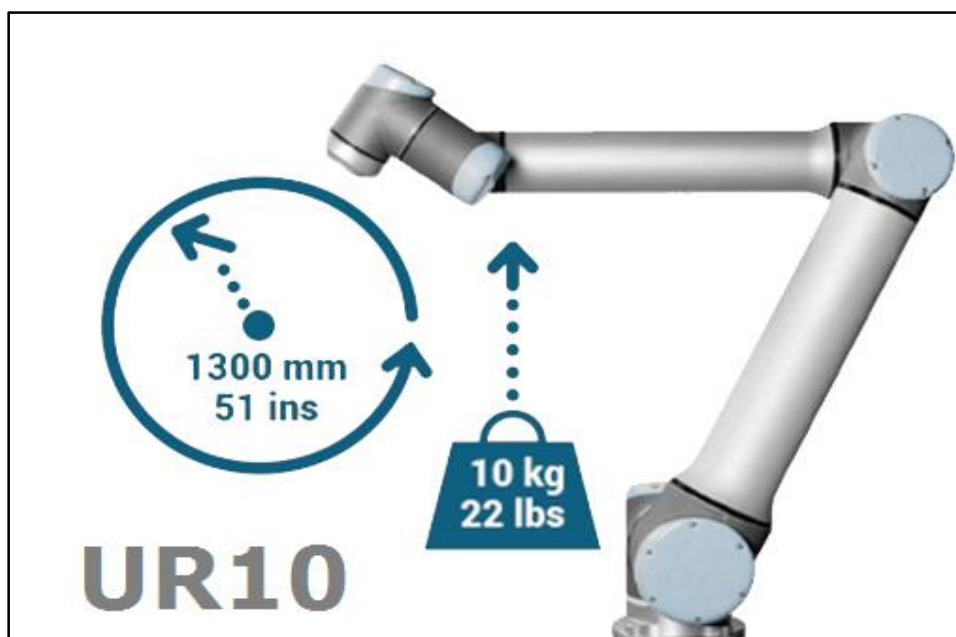


Figure 3-4: UR 10 illustrated [30]

3.1.3 Vehicles

This section explains details about the two vehicles that were used in the experiments. A Peugeot ion [28] and a BMW i3. [31]

Both cars are purely electric cars. While the Peugeot ion is an electric car without communication possibilities, the BMW i3 offers UMTS or Bluetooth and the potential to add software for online services into the car. In other words, the BMW is a “smart” car the Peugeot is not. The main reason for having two different types of cars was testing. The charging station needed to be tested in two different scenarios. A charging test with just one type of car would not be conclusive enough, because the station might fulfil the requirements for one specific type of car but not for others. This leads into two different communication strategies, which will be explained later in this thesis. Therefore, it is important to know the technical specifications of the vehicles, which can be seen in the following figures and tables.

Peugeot ion	
Engine type	Permanent magnet synchronous motor
Total max. power	47 kW
Battery type	Lithium-ion
Battery capacity	16 kWh
Battery voltage	330 V
Battery range	150 km
Top speed	130 km/h
Acceleration 0-100 km/h	15,9 s

Table 3-2: Peugeot ion technical data [28]

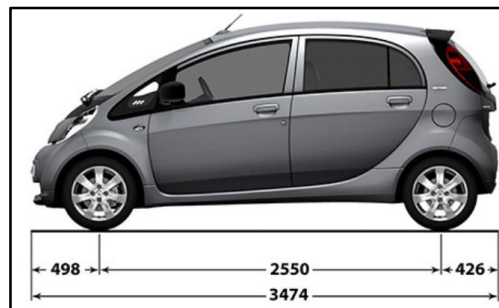


Figure 3-5: Peugeot ion longitudinal dimensions [29]

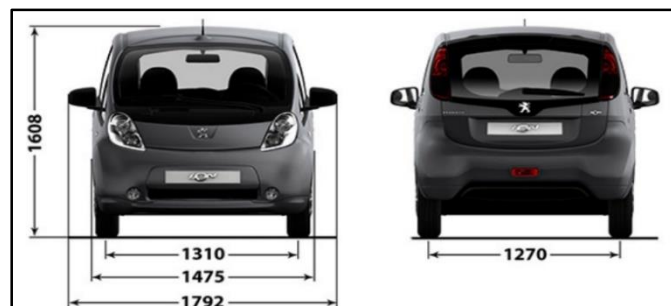


Figure 3-6: Peugeot ion width dimensions [29]

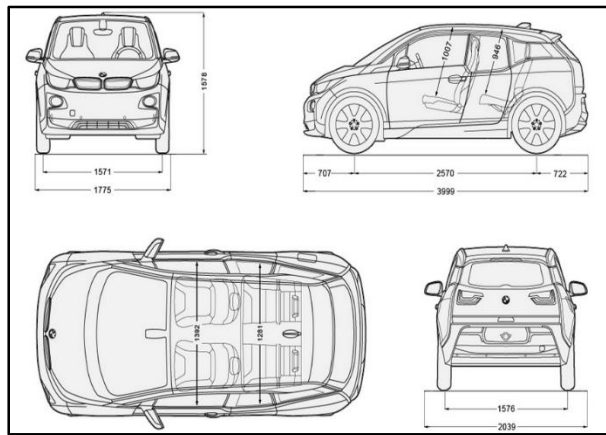


Figure 3-7: BMW i3 dimensions [31]

BMW i3	
Engine type	Hybrid synchronous motor with integrated power electronics, charger and generator mode for recuperation
Total max. power	125 kW
Battery type	Lithium-ion
Battery capacity	33,2/27,2 kWh
Battery voltage	353 Volt
Battery range	190 km
Top speed	150 km/h
Acceleration 0-100 km/h	7,2 s

Table 3-3: BMW i3 technical data [31]

Technical data from both vehicles is essential in order to get an overview of the battery specifications (e.g. battery capacity, voltage, etc.). Additionally, the dimensions have to be considered, when it comes to the parking space, as well as the parking distance for the robot.

3.1.4 Challenges and problems

This section will describe some of the problems that arose during the test phase in the factory hall. The Frank Stronach Institute (FSI) building was under construction for the additional third floor during the time of testing. This restricted the available space and caused noise problems. Parking the cars, making room for other cars or tools was not just disruptive but also affected the work space. Another issue that was faced were the changing conditions. For an effective work, the cameras and sensors should be tested under unchanged and constant conditions. This however was not the case, not only because of the construction works, but also by changing weather conditions, for example changing light conditions. However, it could be argued that these conditions simulated realistic surroundings, for example a roofed public area. Regardless the argumentation points, these conditions still caused many challenges. Another minor issue was that the internet connection via local area network (LAN) cable, did not work consistently.

3.2 Software

This chapter describes the software-infrastructure of the prototype. The software should fulfil minimal requirements in order to be implemented successfully. As mentioned before in the objective targets, the software should be flexible, easy to implement and, if possible, without licence cost. Therefore, it is important to see which layer of the open systems interconnection (OSI) protocols is involved.

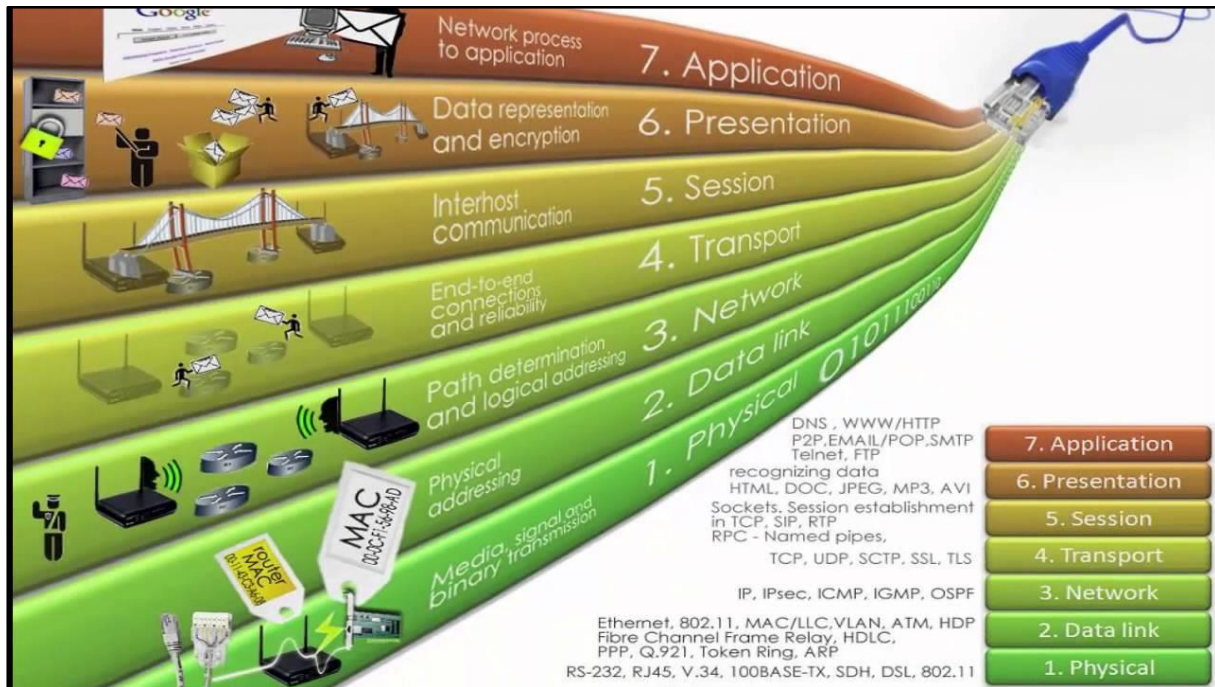


Figure 3-8: OSI-protocols [32]

Figure 3-8 shows that the commands from the user to the charging station happen mainly between the transport and session level, where Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) protocols are used. The main function of the software communication between the devices is either to transport the commands (UDP→Transport level) or to establish a communication path (TCP→Session level). That means that every software, which is used for the project, has to have at least these levels in order to be implemented.

3.2.1 Wi-Fi

In order to realize the communication between the devices an isolated network was implemented, which was not connected to the internet. There were two reasons for this decision. First, it was much more secure if the network was not connected to the World Wide Web. In addition to the possibility of hacking, there was also the issue with the firewall. During the tests, the firewall of the system software was deactivated from time to time in order to communicate with other devices. Deactivating the firewall however was more a safety measure, because it could be possible that the firewall blockades the communication from other devices.

Secondly, it was easier to manage the connected devices in an isolated network. Defining static ports and static Internet Protocol (IP)-addresses are a crucial step for the communication. The charging station for example was only readable by User Datagram Protocol (UDP) with the defined ethernet port 7090. Only this port would allow a communication to the KEBA

station. However as mentioned before, because of the problems regarding the ROS-program, the VPN-client had to be used, in order to be able to work with the Matlab program. The VPN-client makes it possible to establish an external connection to a restricted network (e.g. company network). Therefore, an employee can set up a connection to the company network, without being physically in the company. However, the VPN-gateway allows only assigned users to establish a connection. Figure 3-9 illustrates the principle of a VPN environment.

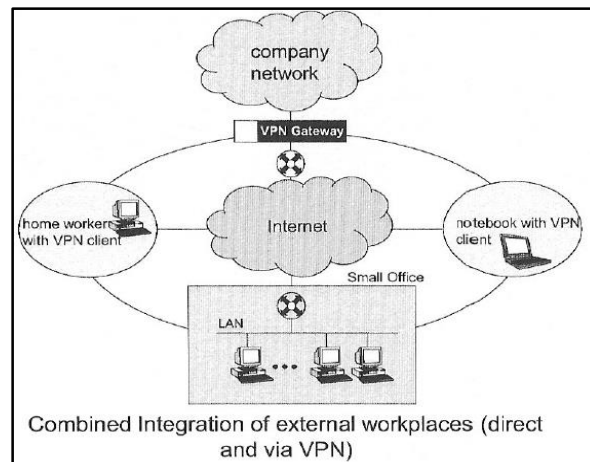


Figure 3-9: VPN function [33]

3.2.2 Challenges and problems

There were truly many problems that arose during this project.

One of the first problems was related to the Media Access Control (MAC). The TUG network is a closed network and therefore does not allow a device without registration a connection. Every communication with a device needed an approval from the ZID. The reason for that is that the TUG-network is a part of the Austrian Academic Computer Network (ACO-net). Graph 3-10 illustrates this network.

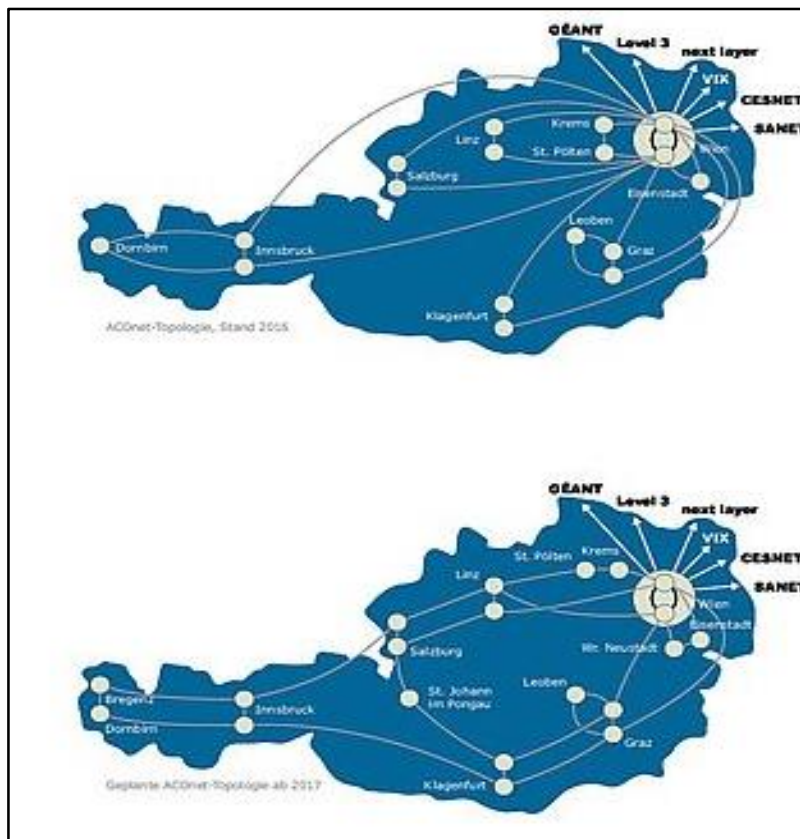


Figure 3-10: Austrian Academic Computer Network (ACO-net) Topology [34]

The next problem occurred with regards to the licence. As mentioned before in the chapter 2.2.1 Matlab, a licenced program not only is more expensive than open source ones, but it also has to have a connection to the license server in order to get the licence. This issue could be solved by bypassing this via adjusting the VPN-client connection properties. However, starting the communication process could only work with the licence. The entire process of getting the licence slowed the work down.

This third problem happened to the ROS program. Despite the fact that the ROS was fully installed, it was impossible to install some additional packages and therefore ROS could not fully communicate with other devices. Despite my research efforts and consultations with an ROS-Expert, the ROS software was not able to work properly. The effort to establish a basic communication between the ROS and the user-interface was unsuccessful because ROS could not react to the message. It seems that the ROS program had trouble with the version. However, the reason for these problems stayed unknown. The solution was by using Matlab for the communication between the robot control and the user-interface. This changed not just the use of the software but also scrapped the use of the Linux system, which was used for ROS.

Most of the communication problems could be solved. However, it is recommended to use ROS instead of Matlab for the robot control. Since it is an open source software, it offers much more possibilities in this field and has no special license costs.

4 Concept for communication

In this chapter, the concept for communication is illustrated. The focus is on the data and the data protocols and as well as what, how many, and which type of data has to be transferred. However, before the communication concept can be established, a clarification of how the data is sent is necessary. As mentioned before in chapter 3.2 Software, the work happens mainly in the transport layer with TCP and UDP protocols. Therefore, in order to create a concept, a basic understanding of how these two data protocols are working is necessary. The main difference is the so-called “Handshake Paradigm”. The TCP cannot start without establishing the connection first. In other words, the TCP protocol needs a handshake before it can transfer data. The picture below illustrates this difference.

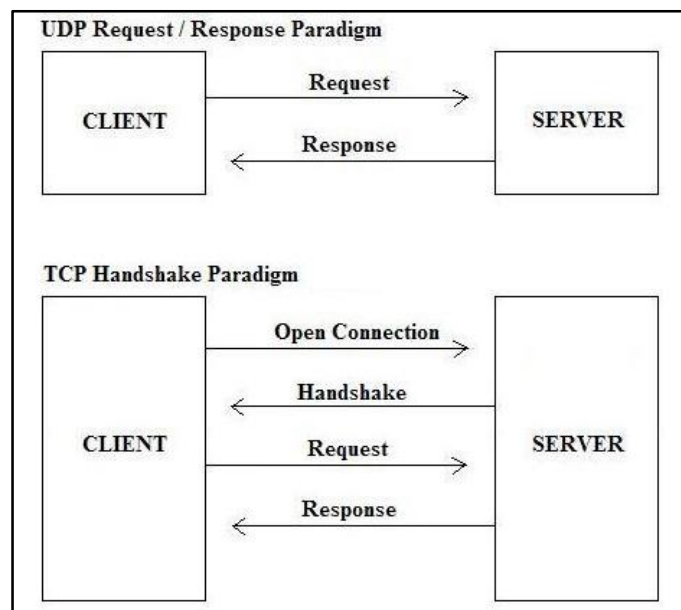


Figure 4-1: Difference between UDP and TCP [35]

As mentioned before the software needs to fulfil the European standard, more precisely the C-ITS protocol stack and the networking and transport layer (figure 2-2). The communication, which was established during the project, could be integrated in the OBU. The OBU communicates then with the RSU in order to get information about the charging infrastructure. These communication paths can be seen image 4-2. The figure shows the Vehicular ad hoc network (VANET) components in detail. Furthermore, the interactions between these components is visualized.

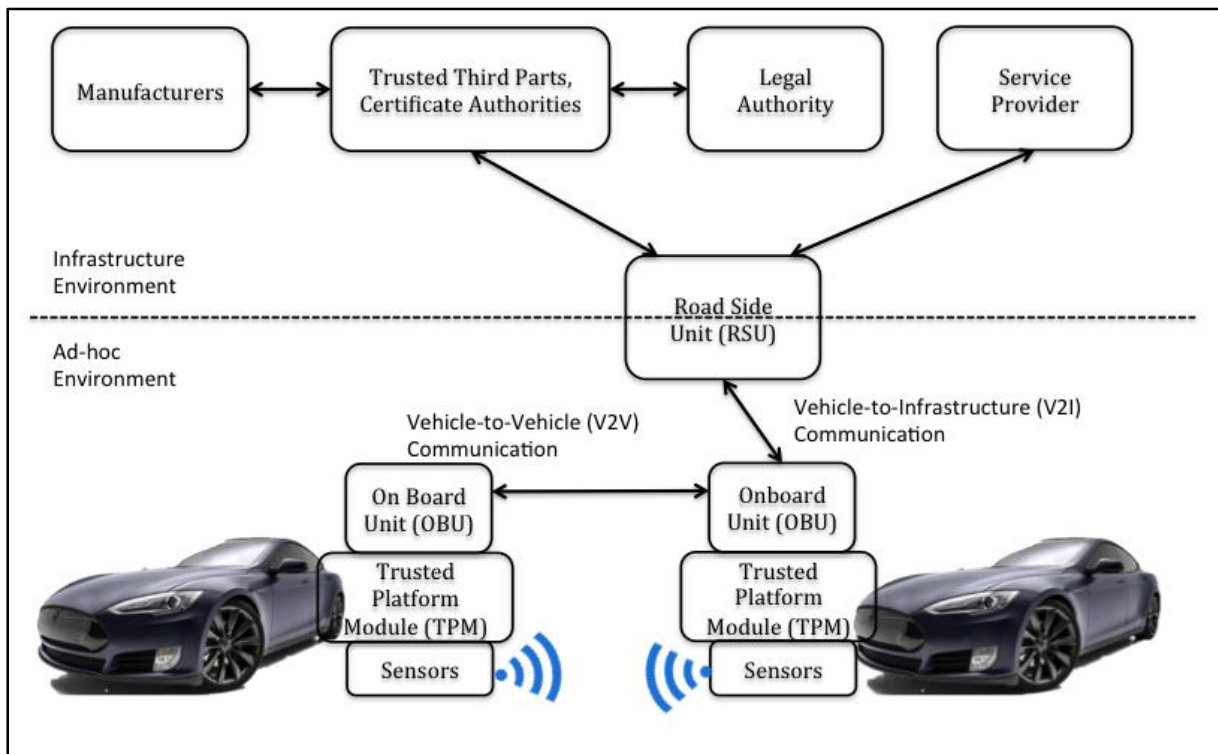


Figure 4-2: VANET components and communication model [36]

This means that the Web-server is located as a service provider, the communication can be executed via RSU and OBU. However, it should be mentioned that a communication between provider and user should be also possible without these units. The big advantage of OBUs is that it can also communicate with other devices as it is illustrated in graph 4-3.

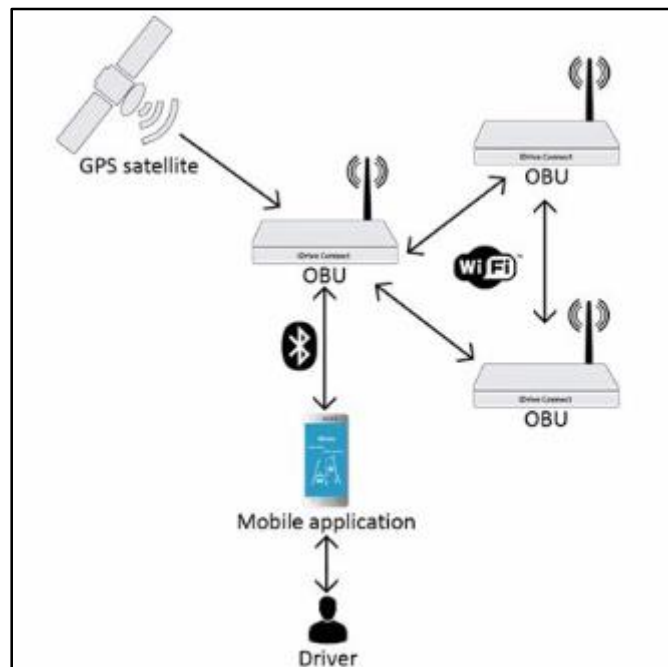


Figure 4-3: OBU System architecture [37]

4.1 Concept for the communication structure

After clarification of the data transport, a general concept for the communication can be created. Figure 4-4 shows the overview of how the devices should interact with each other.

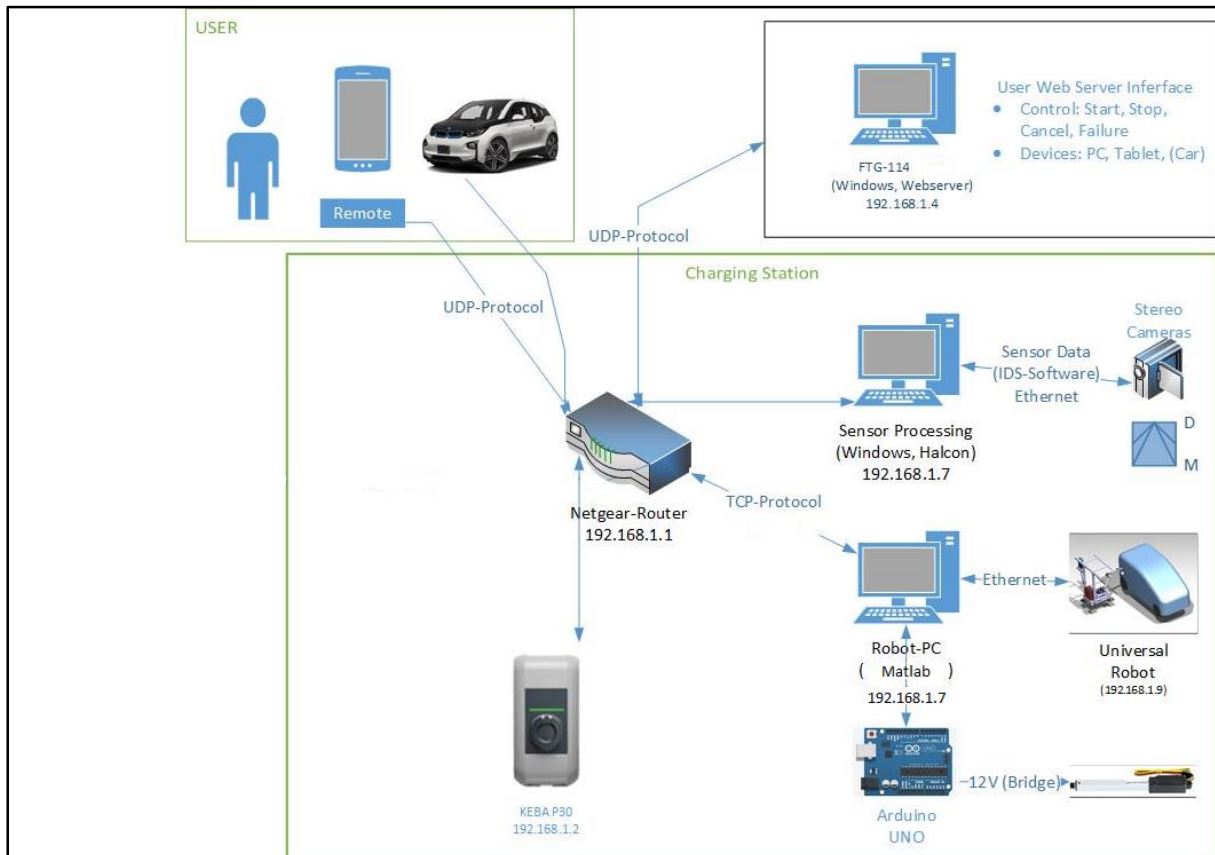


Figure 4-4: General concept

Furthermore, the figure illustrates the communication and data transportation between the units. It is important for the concept to provide a good connection between the devices. The computers, servers and sensors should be connected via wire. Connection via Wi-Fi, Long Term Evolution (LTE) or GSM network cannot guarantee data transportation with high amount of data speed. [38] [39]

The user has to have a wireless connection in order to be independent from the location. In order to guarantee functionality, the data interaction between user and charging station should be as less as possible. High amount of data should be executed via wire connection. [40]

Therefore, the data interaction between cameras, sensors and robotic arm should just happen internally. This internal communication is illustrated with a green frame in figure 4-4. The frame symbolizes a closed unit.

4.1.1 Concept of user data

After establishing the general concept for the communication structure, a detailed look in the data of the individual units is necessary. The first focus of the concept of data is the user. The main reason for that is that the webservice-interface has to be as user friendly as possible. It would not make sense to create a complex platform, and then no potential customer wants to use it. Therefore, a very simple platform is essential, not just for the customer but also for companies. Another reason why the focus lies on the user data is that the user steers the

charging process. The customer decides when the process should start and when the process should end. In that case, the data that comes from the user is very limited. In fact, the only things that is important to know from the user is his identity, if he wants to take the parking place, when he wants to start, the charging procedure and when it should stop. However, this is not as simple as it may sound. This information from the user can only be used in the best-case scenario. If the parking lot is already occupied, or if the charging procedure stops without any active command from the user, this data is not enough. Apparently, additional information is necessary. Therefore, collecting more data from the customer is necessary. In addition to the already mentioned data, the user can be requested for his personal data. This step could be executed with different approaches. The user could type either his e-mail address, phone number, or both. In addition, a registration platform or an app could be an option. Regardless which data the user has to provide, the charging station has to inform the customer on this registration platform.

4.1.2 Charging station data

This chapter considers the charging station data. Please note that the term charging station does not refer exclusively to the charging box. The definition extends to include the server for the web-interface. This means that the charging station includes communication between user, charging box and the webserver. However, the communication between the user and charging station is very simple. The charging station only waits for the approval from the customer. In this case, the assumption is that the user already gave the permission to start the charging process. Therefore, it is necessary to distinguish between webserver and charging box.

Charging box

As mentioned in chapter 3.1.1 the charging box which was used for this project is the KeKontakt P30 Master from KEBA. [24]

This box has already defined UDP commands from the manufacturer, which is illustrated in the table below.

UDP command	KeContact P20	KeContact P30	BMW wallbox
Broadcast Messages	✓	✓	✓
Command "i"	✓	✓	✓
Command "failsafe"	✓	✓	✓
Command "report"	✓	✓	✓
Command "report lxx"	x	✓	✓
Command "ena"	✓	✓	✓
Command "curr"	✓	✓	✓
Command "currtime"	x	✓	✓
Command "setenergy"	x	✓	✓
Command "output"	✓	✓	x
Command "start"	x	✓	✓
Command "stop"	x	✓	✓
Command "display"	x	✓	x
Command "unlock"	✓	✓	✓

Table 4-1: UDP commands for P20, P30 and BMW wallbox [41]

These defined inputs lead to already defined outputs. Furthermore, the table shows that the UDP commands are used in other wallboxes such as the P20 or the BMW wallbox. In order to test the communication, these commands were implemented in the communication concept. Due to infrastructural limitations and safety reasons, the charging box was just adjusted for currents with 230 Volt. Naturally, this limited also the UDP commands. Codes like “start” or “stop” were impractical. However, sending one command and receiving the defined output would be enough to establish the communication. After successfully testing several commands such as “i” or “report”, the communication was established with the command “report 1”. That means that when the user desires to start the charging process, the command that the charging box receives is “report 1” instead of “start”. The output of this command is illustrated in table 4-2.

UDP command:	report 1
Reply:	<pre>{ "ID": "1", "Product": "KC-P20-ES240010-000", "Serial": "15017355", "Firmware": "KEBA P20 v 2.01m11 (140610-073512)" }</pre>
Description:	<p>"ID" = ID of the retrieved report.</p> <p>"Product-ID" = Model name (variant)</p> <p>"Serial" = Serial number</p> <p>"Firmware" = Firmware version</p>

Table 4-2: Defined respond on UDP command “report 1” [41]

Webserver

The second component for the charging station control is the webserver. The webserver controls and establishes the communication between user and charging station. Furthermore, the webserver sends the commands from the user to the station and receives the answers from the charging station. The output from table 4-2 for example, was received and illustrated in the webserver. The task of the webserver is to show if the charging station is free, change its status if not and to transmit the commands from the user, such as start and stop charging. The interaction between box and server happens with the UDP-format. However, for the robotic arm control, the webserver has to communicate with the TCP-format, in order to communicate with Matlab.

4.1.3 Robotic data

This section describes the robotic data. The robotic arm includes the cameras and sensors, for the detection of objects and the control of the robot arm for the movement of the charging cable. The data transfer here happens mainly internal and has very little communication with other devices.

The data transfer between the robot devices is not just very large one, it also transfers many different data types. The data, which is send internal between the cameras, the computer for image processing and robot control and the robot arm, are for example pictures, CAD-data and string commands. The exact data types and necessary steps will be shown in more detail in chapter five, charging strategy.

4.1.4 Vehicle data

Considering the recent trends in the automotive industry, communication between vehicles and charging stations has to be taken into consideration. Therefore, the car must be able to receive the webserver. Smartphones or tablets are able to do so, and even if not, a simple download of a mobile device App solves the problem. This Thesis has to consider both possibilities. This leads into two different communication strategies. One in which the car is “smart” and therefore able to communicate with the web-browser by GSM, UMTE, LTE or WLAN. The other strategy when the car is not able to communicate and therefore the user communicates with the charging station via smartphone or tablet. The question is which data comes from the vehicle. Figure 4-4 illustrates the vehicle in the same position as the smartphone. Therefore, the user communication is similar to the vehicle communication. However, the difference occurs when it comes to autonomous driving. In this case, the vehicle data has additional information, which are important in order to drive automated to the charging station.

4.1.5 Summary

This chapter summarizes the data of the defined units. This data interaction is illustrated in figure 4-5. The communication between user and charging station is controlled by the webserver. User and vehicle have almost the same data transmission, except the orange marked information illustrated in the diagram. The internal communication within the charging station will be described more detailed in chapter 5.

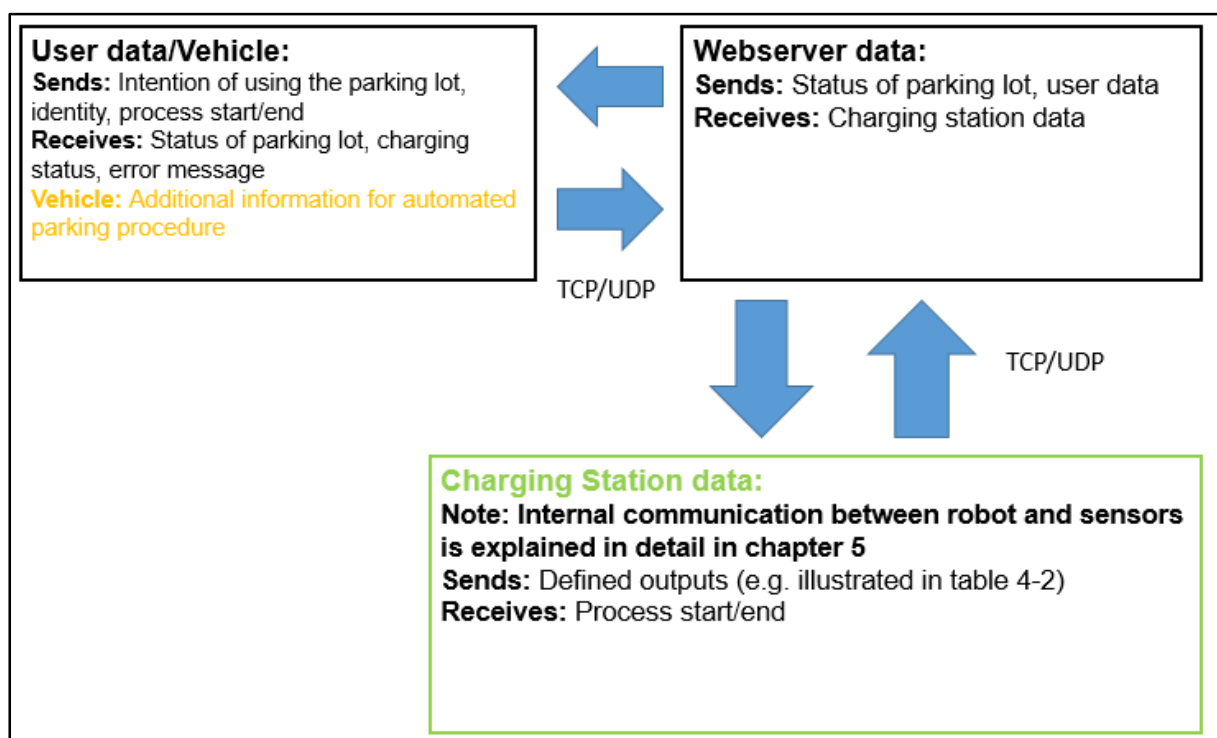


Figure 4-5: Block diagram of data interaction

4.2 Concept of safety

Considering chapter 4.1.1, concept of user data, it is stated that additional personal information is necessary. This automatically raises security requirements. Future customer will not want to provide his or her private information without personal approval. Furthermore, the question of payment will be an issue. This leads to the creation of the security concept, since there is no specified standard for registration and payment via wireless communication. Like the concept of user data, the same principles has to be executed, which are user friendly and simple. The idea is that the platform offers a framework and that the user gets some keywords from this platform. The framework sends the required keywords to the customer's provided e-mail address. When the customer wishes to complete the charging process, it is necessary to type the received keyword from the webserver. The most important point of this concept is that the customer does not forget the password. Another possibility would be to let the customer create their own password. Regardless of which concept is put into use, the platform must provide appropriate information in case of emergency. Therefore, the homepage has footnote with the necessary data, which to show an example in the case of this project is the data from the FTG institute. The figures below illustrate the security concept.

The screenshot shows a web interface with a yellow background. At the top, a banner reads "The parking lot is occupied". Below it, a dropdown menu is labeled "Choose a parking space" with "Parking Spot 1" selected. A text prompt says "Please give us you fully name and E-Mail address, so you can charge your car". There are two input fields: "Name" with "Jane Doe" and "Email" with "jane.doe@example.com" and a "placeholder" label. A "Send invitation" button is below the email field. A red circle highlights the name and email fields, with a red arrow pointing to a text box that says "Input of personal data. With the e-mail address and the send invitation button, the user gets the keywords on his mail address". Below the form are two buttons: "Charge" (green) and "Abort" (red). At the bottom, a red-bordered box contains emergency contact information for FTG, TU Graz, including address, phone number, and email. A red arrow points from this box to a text box that says "Information in case of emergency".

Figure 4-6: Necessary data for security, page one

Figure 4-6 shows the first page, were the user has to type his name and e-mail address. Figure 4-7 shows the necessary password in order to communicate with the web-browser

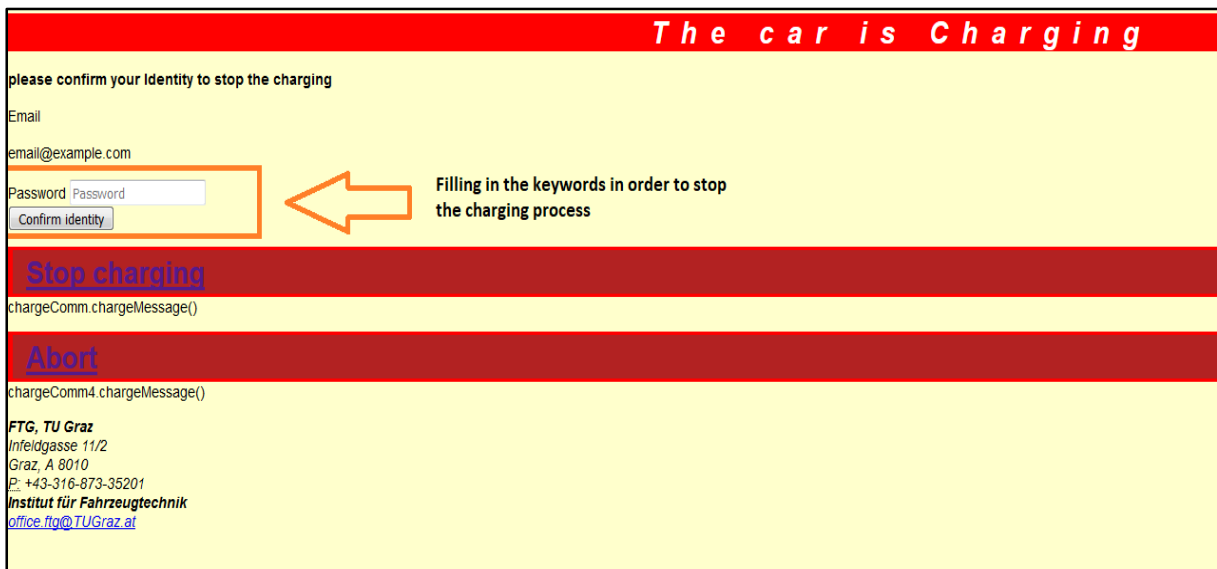


Figure 4-7: Necessary data for security, page two

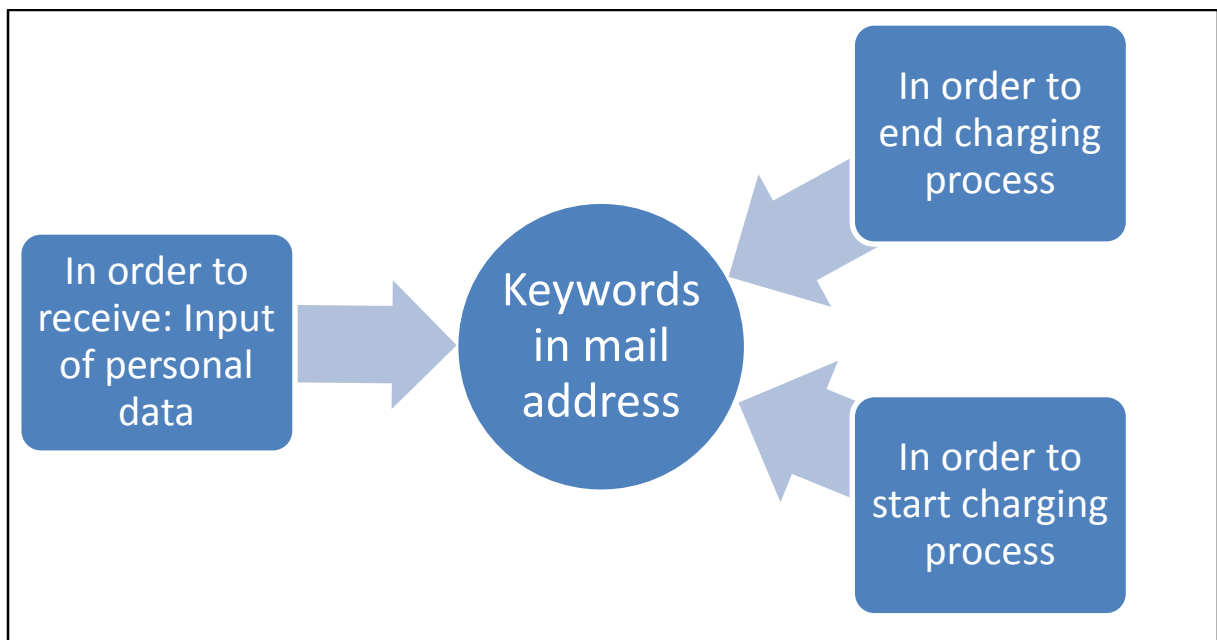


Figure 4-8: Security concept procedure illustrated

Figure 4-8 illustrates the procedure of the security concept. In this figure, the left block and the right block in the bottom are indicating page one, while the right block above is illustrating page two. The essential element of this concept are the keywords, which are received from the inserted personal data. In this case name and e-mail. As mentioned before in order to get the keywords, the user has to send his personal data (page one, insert name and e-mail and press invitation button). Every time when the user wants to steer the charging process via homepage, the user needs to fill in the keywords and therefore confirm his identity.

5 Charging strategy

After finishing all the steps in the previous chapters, the communication standard proposal can finally be defined. The proposal was executed through the charging strategy. Automated conductive charging with heavy standardized charging cables is a complex and sensitive working process. The vehicle has to park in the correct position where the robotic arm can reach the charging point of the car. It has to be mentioned that almost every car has its individual position of the charging socket. Regardless of this situation, this work has to find communication standards for the vast majority of electric cars.

The first things which need to be defined are the preconditions. These conditions have to be fulfilled in order for the communication strategy to work. If just one of these requirement is not met, the entire charging process cannot be executed. Necessary preconditions are:

- Vehicle has enough power to drive to the charging station.
- There is no damage on vehicle and functionality is fully given.
- Charging box is not damaged and functionality is fully given.
- Robotic arm and sensors are not damaged and functionality is fully given.
- Charging station is not blocked and has enough space.
- Driver has to execute registration process before parking

The general case is that the vehicle charging socket with the CCS-Type-2 standard fulfils the ISO 15118 norm. This standard is used in the area of the EU. [42]

It has to be mentioned that combined versions with the Japanese standard CHAdeMO are possible, and is offered at the moment. [43]

Therefore, it is assumed that all vehicles for the charging strategy fulfil these requirements.

The next step in order to establish a charging strategy, is to list the steps for the entire procedure. The table below shows the steps that are defined for the communication strategy. These are summarized in 17 steps.

Step	Target/goal
1	Driver/user wants to start charging process
2	Registration and identification
3	Process start- vehicle drives to the parking lot
4	Vehicle stands at parking lot
5	Vehicle-identification
6	Vehicle- pose-identification
7	Charging socket-pose-identification
8	Opening the charging socket
9	Charging socket-pose-identification
10	Attach procedure
11	Plug procedure
12	Charging procedure (start/end)
13	Charging plug returns
14	Closing the charging socket
15	Robot goes to starting point
16	Charging procedure finished
17	Vehicle leaves parking lot

Table 5-1: Steps for automated charging

5.1 Communication standards for charging strategy

In this chapter, the communication protocols and data formats for the charging strategy are defined. The strategy depends on the 17 steps, which were defined before. As it is visible in the table below, the communication consists of the protocol, and data as well as data formats, which needed to be transferred.

Step	Communication-protocols	Data	Data formats
1	TCP	Take it, which parking lot is free (if more possibilities)	String
2	UDP/TCP	User: name, e-mail, user data, vehicle: charging station position	String
3	-	GPS-data, vehicle position	GPS
4	TCP	Pictures, videos	RGB
5	TCP	Pictures, videos, CAD-data, user data, vehicle data	RGB, string
6	TCP	Pictures, videos, CAD-data, user data, vehicle data	RGB, STL, string
7	TCP	Pictures, videos, CAD-data, user data, vehicle data	RGB, string
8	TCP	Pictures, data for actuator	RGB, string
9	TCP	Pictures, data for actuator	RGB, String
10	TCP	Robot steering data	Position data
11	UDP+TCP	Charging-data, state	String
12	UDP	Charging-state, order	String

13	TCP	Pictures, videos, CAD-data	RGB, String
14	TCP	Pictures, data for actuator	RGB, String
15	TCP	Pictures, videos, CAD-data, robot steering data	RGB, String
16	UDP+TCP	Parking lot free	String
17	-	GPS-data, user position	GPS

Table 5-2: Data, data formats and communication protocols for the charging steps

Table 5-2 also shows the complex communication between the steps. It illustrates the fact that a big amount of data and various data formats are sent between the devices. As it can be seen in the table, the communication protocols are mainly UDP and TCP. The UDP protocol occurs mainly when the charging box is evolved or when a simple message is sent to the customer, for example that the parking lot is free. However, step two can be done either with the TCP or with the UDP protocol. Since both protocols occur in the strategy, the company, which provides the customer data, can decide which one has to be used. Furthermore, it is visible that many different data are communicated between the devices. This leads naturally to many different data formats. As the chart shows, there are string, RGB, STL and position data in order to get the automated charging done. It is helpful to mention that STL (Standard Tessellation Language) is a file format for CAD systems and that RGB (Red, Green, Blue colour system) is a standard for colour and video signals. These data formats are basically communicated inside the charging station. The programs, which are mainly transferring the RGB and STL formats, are HALCON and Matlab. A detailed illustration of this exchange is represented in table 5-3. As mentioned before the international standard ISO 15118 (chapter 2.1.1) is the framework for this thesis. Therefore, it is necessary to compare the defined steps of automated charging prototype test with the steps of the standard. The figure 5-1 shows a sequence illustration of the ISO 15118-6 in case of conductive power transfer.

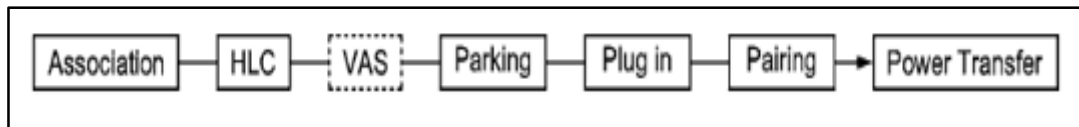


Figure 5-1: Sequence illustrated by ISO 15118-6

The figure shows the steps from the association to the power transfer. As already mentioned in chapter 2.1.1, part 6 of the ISO 15118 is still not implemented as standard yet. However, a consideration of the charging sequence proposed by this draft is necessary since the implementation to a standard is likely to happen. The abbreviations HLC (High Level Communication) and VAS (Value-Added Services) are not specified in the draft. There are also other sequences listed, but the figure 5-1 shows that the steps for automated charging are partially backed up by the standard proposal. However, the charging strategy of this thesis cannot be illustrated entirely by the draft. Therefore, the illustrated sequence from figure 5-1 needs to be adjusted to the charging strategy. Figure 5-2 shows a standard proposal for automated conductive charging leading up until the power transfer.

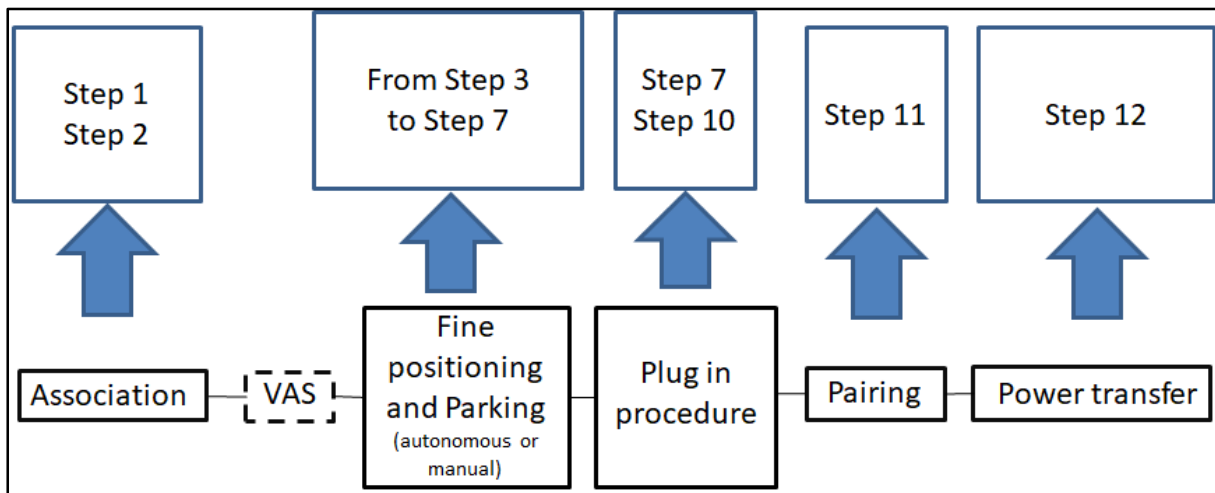


Figure 5-2: Sequence proposal for automated charging till power transfer

The charging strategy doesn't end here and the automated charging process includes 17 steps, meaning 5 steps are still missing. It has to be noted that the ISO 15118 does not provide any specific guidelines for the required steps after the charging process has finished. Most of the illustrated sequences with standardized steps end with the power transfer. However, in this thesis a standard proposal for the detachment process was to be delivered. Figure 5-3 shows the chronological sequence of the charging process.

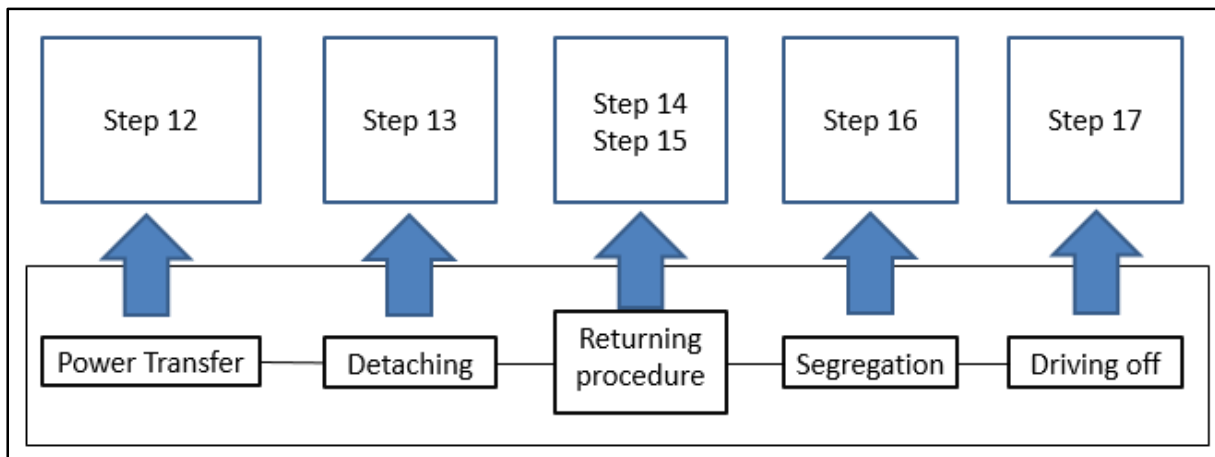


Figure 5-3: Sequence proposal for automated charging after power transfer

5.2 Strategy for “smart vehicle” case

After the defined steps and communication standards, the definition of the strategy for charging a “smart” car can finally happen. It has to be mentioned that the ISO 15118 expressly states that driver manipulation is forbidden in some countries. [44]

Therefore, the user should not, for example be able to choose between 20 possible parking lots, since this would likely violate the driver manipulation reference. The car should offer just the closest stations to the driver. As stated before in chapter 3.1.3, there have been two vehicles available. In this situation, the BMW i3 was simulated as “smart vehicle”. That means that the user communicates through the car and that the vehicle was able to drive autonomously to the charging station. The table below shows an abstract of this strategy.

Step	Target/Goal	Execution/Realization	Program-communication
1	Driver/User wants to start charging process	User reserves parking lot with the "TAKE IT" button	User-webserver-charging station
2	Registration and identification	Input of user data	User-webserver-charging station
3	Process start- vehicle drives to the parking lot	Autonomous driving functions of the car	User-server
4	Vehicle stands at parking lot	Parking lot-object recognition	Halcon-server
5	Vehicle-identification	Car recognition and classification through sensors/cameras and stored server data, which car is in the parking lot, does this car belongs to the user	Halcon-Matlab(ROS)-server
6	Vehicle-pose-identification	Recognition of vehicle type and estimated vehicle position (comparing data between stored data and sensor/camera data)	Halcon-Matlab(ROS)-server
7	Loading cover-Pose-Identification	Comparison of the charging-socket-geometry with the vehicle pictures (comparing data between stored data and sensor/camera data)	Halcon-Matlab (ROS)-server
8	Opening the loading cover	Charging-socket opens (either car opens it automatically or via actuator from robot)	Halcon-Matlab (ROS)-Arduino
9	Charging socket-pose-identification	Comparison of the charging-socket-geometry with the camera pictures	Halcon-Matlab(ROS)
10	Attach procedure	Robot puts charging cable	Halcon-Matlab(ROS)-robot
11	Plug procedure	Charge control (communication between EV and EVSE according to the ISO 15118)	Halcon-Matlab(ROS)-charging box
12	Charging procedure (Start/End)	User stops process/ battery fully charged	User-webserver-charging box
13	Charging plug returns	Robot pulls back	Matlab (ROS)
14	Closing the loading cover	Charging-socket closes	Matlab (ROS)-Arduino
15	Robot goes to starting point	Robot pulls back to starting point	Matlab(ROS)
16	Charging procedure finished	Parking lot is free	Webserver-user
17	Process end- vehicle leaves the parking lot	Autonomous driving functions of the car	User-server

Table 5-3: Targets, execution and program communication for the smart car steps

In the first step, the user connects with the charging infrastructure and takes the free parking lot. Therefore, the user has to press the "TAKE IT" button under the condition that the parking lot is free. With the command, "TAKE IT" the customer signalizes to the charging station that a car is coming. The image below shows the webserver page.



Figure 5-4: User interface page 1

The red arrow shows the “TAKE IT” button and the yellow one shows the possibility of choosing a parking lot if one is already full. This possibility only exists if there are more than just one parking lot.

The next step is the input of user-data. The figure below shows the placeholder for the user-information marked with the red arrow.

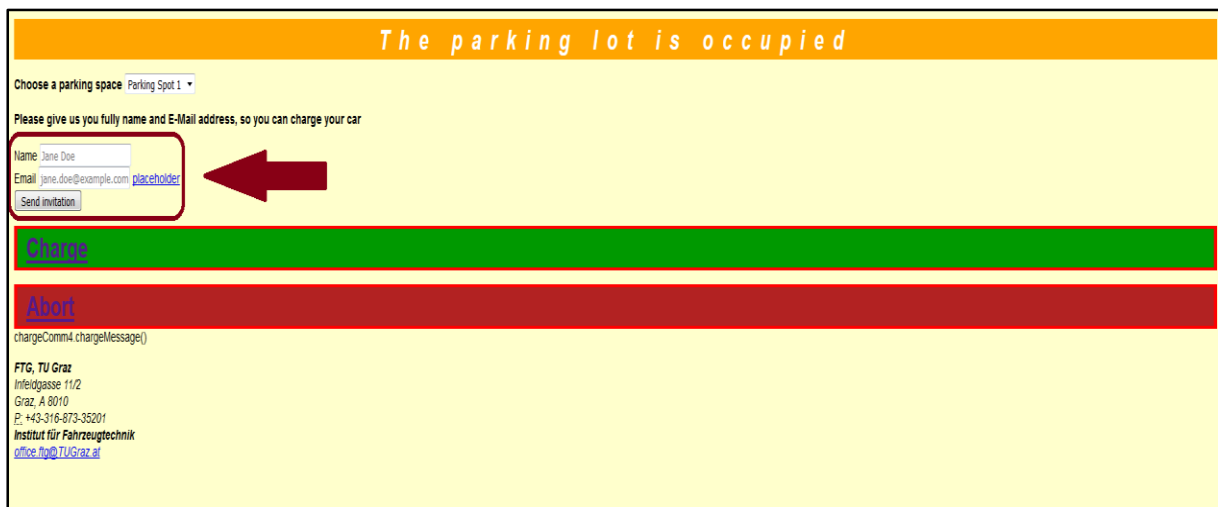


Figure 5-5: User interface page 2

After the second step, the automated process starts. The user does not need to do anything until process step twelve. In this step, the user is informed via e-mail that the charging process can start. In this case, the user needs to press the “Charge” button, in order to start the charging process. This step is necessary for the user, to know that everything is working fine and that the station does not have any problems. The distinction between infrastructure and charging box is necessary because it is possible that the car may not be charged despite the connection to a power grid system. There can be many reasons for this, for example that every charging box needs to have a power managing software in order to end the charging process if the grid system is too busy. In addition, the customer is then in control of the charging process and can start and end the process whenever he wants. However, before he can do that, the customer needs to confirm this identity via password, as shown in the figure below, marked also with a red arrow.

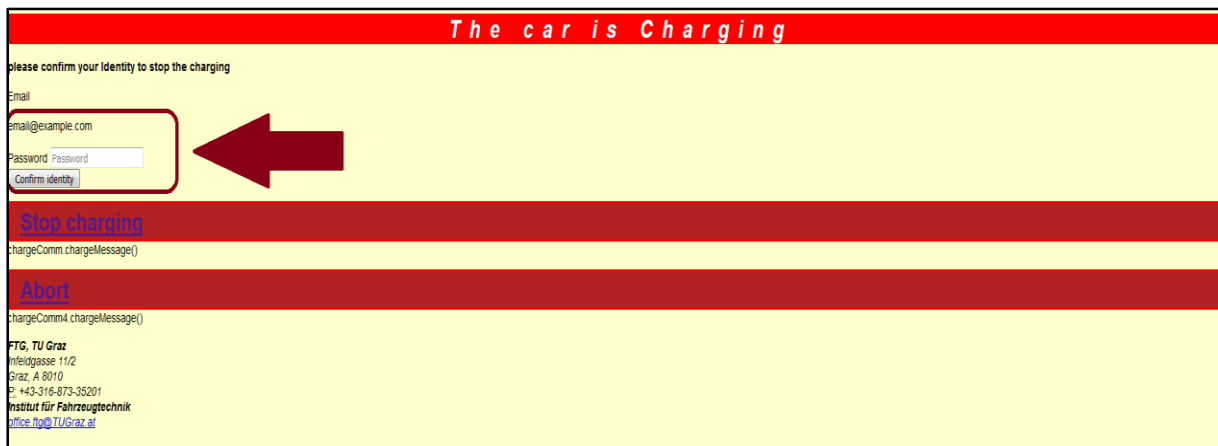


Figure 5-6: User interface page 3

This measure is necessary since this homepage is open to everyone. That means that everyone can see the state of the charging station and is furthermore able to steer it. In order to avoid abuse this step is inevitable.

When the car is fully charged, the user receives an e-mail indicating that the charging process is finished and that he has to move the vehicle or he has to give the permission for the car to move itself. Theoretically, the car could move automatically, in order to make space for the next car to be charged. This would be a solution to speed up the charging procedure of electric cars in general. However, from a legal point of view it would be very risky to move a car autonomously without informing the owner. Regardless of the choice of the finishing procedure, the vehicle has to be moved as quickly as possible and the owner must be informed about the transportation of the car. The parking lot then signals that it is free again.

5.3 Strategy for the “non-smart” vehicle

This chapter looks at the case if the user does not have a “smart” car. This is the case of the Peugeot ion. As it is illustrated in the table below, in comparison to the strategy for the “smart” car, two-steps are missing here. The car is not able to drive in or out of the parking lot itself. This means that the user has to drive. In addition, the car is not able to communicate with the charging station. This is possible via a smartphone. The only condition that needs to be fulfilled is that the smartphone needs to have a web-browser.

Other crucial steps are 7 and 13. Here the user needs to be sure that the loading cover can be opened, so that no mechanical locks are blocking. If a special locking device exists, the user has to ensure a clear path to the charging socket. Everything else is similar to the “smart” car strategy.

Step	Target/Goal	Execution/Realization	Program-communication
1	Driver/User wants to start charging process	User reserves parking lot with the TAKE IT button	User-webserver-charging station
2	Registration and identification	Input of user data	User-webserver-charging station
3	Vehicle stands at parking lot	Parking lot-object recognition	Halcon-server
4	Vehicle-identification	Car recognition and classification through sensors/cameras and stored server data, which car is in the parking lot, does this car belongs to the user	Halcon-Matlab(ROS)-server
5	Vehicle-pose-identification	Recognition of vehicle type and estimated vehicle position (comparing data between stored data and sensor/camera data)	Halcon-Matlab(ROS)-server
6	Loading cover-pose-identification	Comparison of the charging-socket-geometry with the vehicle pictures (comparing data between stored data and sensor/camera data)	Halcon-Matlab(ROS)-server
7	Opening the loading cover	Charging-socket opens (either car opens it automatically or via actuator from robot)	Halcon-Matlab(ROS)-Arduino
8	Charging socket-pose-identification	Comparison of the charging-socket-geometry with the camera pictures	Halcon-Matlab(ROS)
9	Attach procedure	Robot puts charging cable	Halcon-Matlab(ROS)-robot
10	Plug procedure	Charge control	Halcon-Matlab(ROS)-charging box
11	Charging procedure (Start/End)	User stops process/ Battery fully charged	User-webserver-charging box
12	Charging plug returns	Robot pulls back	Matlab(ROS)
13	Closing the loading cover	Loading cover closes	Matlab(ROS)-Arduino
14	Robot goes to starting point	Robot pulls back to starting point	Matlab(ROS)
15	Charging procedure finished	Parking lot is free	Webserver-user

Table 5-4: Targets, execution and program communication for the “non-smart” car steps

5.4 Summary

This chapter gives an overview of implemented, proposed and considered standards that are taken into consideration from drafts. Table 5-5 illustrates the overview by comparing the proposal from the thesis with existing standards and draft standards, which influenced the proposals. Furthermore, the table shows the steps which has been carried out in the field test. It has to be mentioned that not every step was checked. Due to lack of possibilities step 3,5 and 6 were assumed. Other steps like 3, 12 and 17 were simulated due to lack of infrastructure (e.g. charging procedure).

Step	Target/goal	Field test	Existing standard	Existing draft standard	User interface
1	Driver/user wants to start charging process	Tested	No	Yes	Page 1
2	Registration and identification	Assumption	No	Yes	Page 2
3	Process start- vehicle drives to the parking lot	Simulated	No	No	Page 2
4	Vehicle stands at parking lot	Tested	No	Yes	Page 2
5	Vehicle-identification	Assumption	No	Yes	Page 2
6	Vehicle-identification	Assumption	No	Yes	Page 2
7	Charging socket-pose-identification	Tested	No	Yes	Page 2
8	Opening the charging socket	Tested	No	Yes	Page 2
9	Charging socket-pose-identification	Tested	No	Yes	Page 2
10	Attach procedure	Tested	No	Yes	Page 2
11	Plug procedure	Tested	No	Yes	Page 2
12	Charging procedure (start/end)	Simulated	Yes	Yes	Page2/Page 3
13	Charging plug returns	Tested	No	No	Page2
14	Closing the charging socket	Tested	No	No	Page2
15	Robot goes to starting point	Tested	No	No	Page2
16	Charging procedure finished	Tested	No	No	Page2
17	Vehicle leaves parking lot	Simulated	No	No	Page 1

Table 5-5: Charging strategy summary

The chart also shows how the user interface interacts with the strategy. It is essential to mention that the user gets informed via his personal data (e.g. in this strategy e-mail) that the identification and attach procedure is finished and that the user can start the charging procedure.

6 Conclusion and Outlook

This thesis dealt with communication standards for automated conductive charging of electro and plug-in-hybrid vehicles. After the research of V2X networks, the thesis looked into the already used and implemented standards. Based on the research and the field test results, which has been carried out at the Institute of Automotive Engineering proposals for communication standards were developed through the charging strategy

The field tests were executed, considering the mentioned objective targets from chapter 1.2. While carrying out the tests, this thesis looked into the variety of soft- and hardware, mentions the advantages as well as disadvantages, and gives an answer as to why certain programs and hardware were used. The infrastructure that was used for this project is also listed. Before an elaboration of a standard proposal a concept for the prototype communication was necessary. In this concept, the data from the different units needed to be considered. Additionally, a concept for communication safety was required, since the entire communication process has to be freely available. Furthermore, the proposed standard was compared with already established norms and considered norms still in development. The communication standard proposal was then implemented and executed in charging strategies for a “smart” and a “non-smart” vehicle. There are still obstacles that will occur such as legal and insurance issues. That includes issues caused by autonomous driving like legal problems, such as the Vienna convention [46] or insurance problems. [47]

However, dealing with these obstacles was not goal of this thesis.

The future of the drive technology in the automobile industry is uncertain. However, electric cars will play a big role in the near future. It is expected that the number of electric vehicles sold will continue to grow. Additionally, more and more cities will ban fossil fuel cars in order to lower their carbon dioxide emissions, which should motivate people to buy alternatively driven vehicles. [45]

Providing sufficient infrastructure is therefore crucial. This leads to the conclusion that automated conductive charging is inevitable. Automatization and therefore communication, which supports this process will play a major role in the industry.

7 Links

<https://nodejs.org/api/dgram.html>
<http://therevproject.com/doc/2012-EVcharging-s.pdf>
https://vector.com/portal/medien/cmc/press/Vector/EV_EMobility_Standards_HanserAutomotive_201610_PressArticle_EN.pdf
<http://htmldog.com/guides/html/beginner/>
<http://www.erg.abdn.ac.uk/users/Gorry/course/inet-pages/udp.html>
<http://www.hacksparrow.com/node-js-udp-server-and-client-example.html>
https://github.com/abhinavjain241/comm_tcp
<http://brandonalexander.com/rosnodejs/>
[https://msdn.microsoft.com/de-de/library/system.net.sockets.socket\(v=vs.110\).aspx](https://msdn.microsoft.com/de-de/library/system.net.sockets.socket(v=vs.110).aspx)
[https://msdn.microsoft.com/de-de/library/system.net.sockets.tcpclient\(v=vs.110\).aspx](https://msdn.microsoft.com/de-de/library/system.net.sockets.tcpclient(v=vs.110).aspx)
<https://msdn.microsoft.com/de-de/library/bb979208.aspx>
<https://store.codesys.com/communication-with-halcon.html>
<http://www.mvtec.com/de/produkte/halcon/arbeiten-mit-halcon/programmierung/>
<http://www.mvtec.com/de/produkte/halcon/arbeiten-mit-halcon/hdevenge/>
http://wiki.ros.org/common_msgs
<http://wiki.ros.org/ROS/Tutorials/CreatingPackage>
<https://blog.risingstack.com/your-first-node-js-http-server/>
<http://www.erg.abdn.ac.uk/users/Gorry/course/inet-pages/udp.html>
<https://forum.unity3d.com/threads/simple-udp-implementation-send-read-via-mono-c.15900/>
<https://stackoverflow.com/questions/13817732/udp-in-c-sharp-works-on-windows-but-not-linux>
<https://gist.github.com/tedmiston/5935757>
<https://nodejs.org/en/>
<http://www.itwissen.info/Sicherheitsprotokoll-security-protocol.html>
http://ec.europa.eu/eurostat/statistics-explained/index.php/Electricity_price_statistics
http://festag-net.de/doc/2014_CommMag_C-ITS_festag.pdf
<https://ec.europa.eu/transport/sites/transport/files/themes/its/news/doc/c-its-platform-deployment/draft-work-programme.pdf>

8 Table of figures

Figure 1-1: Battery electric vehicle sales [1]	9
Figure 1-2: Gasoline and diesel refuelling in comparison with electric charging [2].....	9
Figure 1-3: Gasoline and electricity price fluctuation [3].....	10
Figure 1-4: Initial situation of the communication strategy [4].....	11
Figure 1-5: Steps of selection	13
Figure 2-1: Two DSRC protocol stacks, WSMP (left) and TCP/IPv6 (right) [9].....	15
Figure 2-2: Protocol stack and release 1 core standards for C-ITS in Europe [10].....	16
Figure 2-3: Software modules for V2G communication [11].....	16
Figure 2-4: Matlab code for the TCP communication	19
Figure 2-5: Matlab output of transferred data.....	20
Figure 2-6: HALCON, Matlab data problem	20
Figure 2-7: Screenshot “starting the webserver”	21
Figure 2-8: Folder structure	22
Figure 2-9: Halcon program window	23
Figure 2-10: TCP-protocol for Halcon	24
Figure 2-11: Program formation.....	25
Figure 3-1: Charging systems [16].....	27
Figure 3-2: Communication protocols for charging systems [18].....	27
Figure 3-3: Inductive charging [22]	28
Figure 3-4: UR 10 illustrated [30].....	29
Figure 3-5: Peugeot ion longitudinal dimensions [29].....	30
Figure 3-6: Peugeot ion width dimensions [29]	30
Figure 3-7: BMW i3 dimensions [31].....	31
Figure 3-8: OSI-protocols [32]	32
Figure 3-9: VPN function [33]	33
Figure 3-10: Austrian Academic Computer Network (ACO-net) Topology [34].....	34
Figure 4-1: Difference between UDP and TCP [35]	35
Figure 4-2: VANET components and communication model [36].....	36
Figure 4-3: OBU System architecture [37]	36
Figure 4-4: General concept.....	37
Figure 4-5: Block diagram of data interaction	40
Figure 4-6: Necessary data for security, page one	41
Figure 4-7: Necessary data for security, page two	42
Figure 4-8: Security concept procedure illustrated.....	42
Figure 5-1: Sequence illustrated by ISO 15118-6	45
Figure 5-2: Sequence proposal for automated charging till power transfer	46
Figure 5-3: Sequence proposal for automated charging after power transfer.....	46
Figure 5-4: User interface page 1	48
Figure 5-5: User interface page 2	48
Figure 5-6: User interface page 3	49
Figure 12-1: Main web-browser program [48].....	62
Figure 12-2: UDP Client [49].....	63
Figure 12-3: UDP Server [49]	63
Figure 12-4: TCP-Socket [50].....	64
Figure 12-5: First page of user interface	65
Figure 12-6: Second page of user interface.....	66

Figure 12-7: Second page of user interface.....	67
Figure 12-8: General design of pages [48].....	68
Figure 12-9: Design of page writings [48]	69
Figure 12-10: Size of borders [48]	70
Figure 12-11: Main class	71
Figure 12-12: Program to set act	71
Figure 12-13: Method of importing the message.....	72
Figure 12-14: Import of message.....	72
Figure 12-15: Output of string message	72
Figure 12-16: Arduino part.....	73
Figure 12-17: Loop function.....	73
Figure 12-18: Exit function.....	73

9 List of tables

Table 2-1: Comparison of PHYs implementations in IEEE 802.11a and IEEE 802.11p [6]...	15
Table 3-1: Technical specification for UR10-CB3 [27]	29
Table 3-2: Peugeot ion technical data [28]	30
Table 3-3: BMW i3 technical data [31]	31
Table 4-1: UDP commands for P20, P30 and BMW wallbox [41]	38
Table 4-2: Defined respond on UDP command “report 1” [41]	39
Table 5-1: Steps for automated charging	44
Table 5-2: Data, data formats and communication protocols for the charging steps.....	45
Table 5-3: Targets, execution and program communication for the smart car steps.....	47
Table 5-4: Targets, execution and program communication for the “non-smart” car steps ...	50
Table 5-5: Charging strategy summary	51
Table 12-1: Charging strategy part 1.....	74
Table 12-2: Charging strategy part 2.....	74

10 Bibliography

Angermann, M. Beuschel, M. Rau, and U. Wohlfarth, *MATLAB - Simulink - Stateflow: Grundlagen, Toolboxen, Beispiele*, 6th ed. Berlin/Boston: De Gruyter; De Gruyter Oldenbourg, 2009.

Austria Standards Institute, „EN ISO 15118-1 Road vehicles – Vehicles to grid communication interface, Part 1: General Information and use-case definition, Vienna, Edition 2016-01-15

Austria Standards Institute, „EN ISO 15118-2 Road vehicles – Vehicles to grid communication interface, Part 2: Network and application protocol requirements, Vienna, Edition 2016-01-15

Austria Standards Institute, „EN ISO 15118-3 Road vehicles – Vehicles to grid communication interface, Part 3: Physical data and link layer requirements, Vienna, Edition 2016-09-01

Wagner, Henning; Maier, Reinhard; Schubert, Jürgen (Hg.) (2015): *Alternative Antriebe - E-Mobilität. Wie wird man Fachkundiger für Arbeiten an Hochvolt-Systemen im Kraftfahrzeug? ; Hybridfahrzeuge, Elektrofahrzeuge, Brennstoffzellenfahrzeuge*. 2., überarb. und korrigierte Aufl. Konstanz: Christiani.

Meroth, Ansgar; Tolg, Boris (Hg.) (2008): *Infotainmentsysteme im Kraftfahrzeug*. 1. Aufl. s.l.: Vieweg (Vieweg Praxiswissen).

Mitchell, William John; Borroni-Bird, Chris; Burns, Lawrence D. (Hg.) (2010): *Reinventing the automobile. Personal urban mobility for the 21st century*. Cambridge, Mass: Massachusetts Institute of Technology.

Lienkamp, Markus (Hg.) (2012): *Elektromobilität. Hype oder Revolution?* Berlin, Heidelberg: Springer (VDI-Buch).

Koubaa, Anis (Hg.) (2016): *Robot Operating System (ROS). The Complete Reference (Volume 1)*. 1st ed. 2016. Cham, s.l.: Springer International Publishing (Studies in Computational Intelligence, 625).

11 References

- [1] Electric cars report, *Electric Vehicle Sales to Reach 3.8 Million Annually by 2020*. [Online] Available: <https://electriccarsreport.com/2013/01/electric-vehicle-sales-to-reach-3-8-million-annually-by-2020/>.
- [2] A.-M. Brouet, *Charging an electric car as fast as filling a tank of gas*. [Online] Available: <http://www.technologist.eu/charging-an-electric-car-as-fast-as-filling-a-tank-of-gas/>.
- [3] J. Voelcker, *Electric Cars Equal \$1/Gallon Gas For Life + \$1,200 Cash A Year*. [Online] Available: http://www.greencarreports.com/news/1076450_electric-cars-equal-1-gallon-gas-for-life-1200-cash-a-.
- [4] B. Walzel, C. Sturm, J. Fabian, and M. Hirz, "Automated robot-based charging system for electric vehicles," Fahrzeugtechnik Institut, Technische Universität Graz, Graz, 2017.
- [5] Payerl CH, "Integration von Car-to-X Kommunikation in die E/E-Architektur von Fahrzeugen," Master-Thesis, Technische Universität Graz, Graz, 2013.
- [6] *IEEE 802.11P: Wireless Access in Vehicular Environment*. Challenges and Future Trends in Vanet. [Online] Available: <https://gauravpathak07.wordpress.com/>.
- [7] Wi-Fi Alliance, *Wi-Fi CERTIFIED™*. [Online] Available: <https://www.wi-fi.org/certification>. Accessed on: February 2018.
- [8] L. A. Svenson, G. Peredo, and L. Delgrossi, *DEVELOPMENT OF A BASIC SAFETY MESSAGE FOR TRACTOR-TRAILERS FOR VEHICLE- TO-VEHICLE COMMUNICATIONS*. [Online] Available: <https://www-esv.nhtsa.dot.gov/proceedings/24/files/24ESV-000379.PDF>. Accessed on: February 2018.
- [9] G. Corser *et al.*, *Privacy-by-Decoy: Protecting Location Privacy Against Collusion and Deanonymization in Vehicular Location Based Services*. [Online] Available: https://www.researchgate.net/figure/269293897_Fig-1-Two-DSRC-protocol-stacks-WSMP-left-and-TCP-IPv6-right. Accessed on: Dec. 21 2017.
- [10] A. Festag, *Cooperative Intelligent Transport Systems Standards in Europe*. [Online] Available: https://www.researchgate.net/figure/273395691_fig1_Figure-1-Protocol-stack-and-Release-1-core-standards-for-C-ITS-in-Europe.
- [11] D. Grossmann and H. Hild, *Smart Charging-A key to successful E-Mobility: Development and testing of charging electronics conforming to standards*. [Online] Available: https://vector.com/portal/medien/cmc/press/Vector/EV_Smart_Charging_ElektronikAutomotive_201407_PressArticle_EN.pdf.
- [12] *Road vehicles -- Vehicle to grid communication interface: ISO 15118*.
- [13] A. Hinsberger, J. Vogt, S. Weber, and H. Wieker, *MANAGEMENT OF ROADISDE UNITS FOR THE sim TD FIELD TEST (GERMANY)*. [Online] Available: http://www.simtd.de/index.dhtml/object.media/enEN/5925/CS/-/backup_publications/Vortrge/simTD_20090921 ITS2009_HTW_Hinsberger.pdf.
- [14] Sparkfun, *What is an Arduino?* [Online] Available: <https://learn.sparkfun.com/tutorials/what-is-an-arduino>.
- [15] Opi2020, AGVS, Schweizer Forum Elektromobilität, VSEI and infovel. Supported from EnergieSchweiz, Elektrizitätsunternehmen Repower, EBM (Genossenschaft Elektra Birseck, Münchenstein), BKW and the Elektrizitätswerken des Kantons Zürich (EKZ), "e'mobile, Electrosuisse, VSE: Merkblatt Ladeinfrastruktur Elektrofahrzeuge," Bern, 2011.
- [16] K. O. Zehle, *Und ich dachte immer AC/DC sei eine Australische Hard-Rock-Band*.

- [Online] Available: <http://www.klausolafzehle.de/teslablog/tag/combined-charging-system/>.
- [17] Z. McDonald, *A Simple Guide to DC Fast Charging*. [Online] Available: <https://www.fleetcarma.com/dc-fast-charging-guide/>. Accessed on: February 2018.
- [18] M. Pottorff, *As of June 25 how many SAE combo fast charging stations are there in the continental United States?* [Online] Available: <https://www.quora.com/As-of-June-25-how-many-SAE-combo-fast-charging-stations-are-there-in-the-continental-United-States>.
- [19] K. Mucevski, *Automotive CAN Bus System Explained*. [Online] Available: <https://www.linkedin.com/pulse/automotive-can-bus-system-explained-kiril-mucevski>.
- [20] H. Wright, *Introduction to Programmable Logic Controllers (PLCs) and the Operational Function of Main System Modules*. [Online] Available: <https://www.maximintegrated.com/en/app-notes/index.mvp/id/4701>.
- [21] *Electric vehicle wireless power transfer systems (WPT) - Part 1: General requirements (IEC 69/236/CD:2012): DIN EN 61980-1*, 2013.
- [22] A. Vanoost and M. Ponsar, *Innovative fast inductive charging solution for electric vehicles: Research Progress - Information for scientists*. [Online] Available: <http://www.fastincharge.eu/scientist.php>.
- [23] M. Luciano, *The Inconvenient Truths About Wireless Charging*. [Online] Available: <https://www.wirelessdesignmag.com/blog/2017/02/inconvenient-truths-about-wireless-charging>. Accessed on: February 2018.
- [24] The mobility house, *Ladestationen-KeBa*. [Online] Available: https://shop.mobilityhouse.com/de_de/ladestationen/l/keba.html?msclkid=cf50d148c9441ed97ca48bcf24b767c4&utm_source=bing&utm_medium=cpc&utm_campaign=AT_Ladestation%20Hersteller&utm_term=KEBA%20P30&utm_content=KEBA%20P30%20%2B%20Generic. Accessed on: February 2018.
- [25] *Electric Vehicle Charging Infrastructure Deployment Guidelines Electric Vehicle Charging Infrastructure Deployment Guidelines for the Greater San Diego Area: DE-EE0002194*, 2010.
- [26] Universal Robots, *Universal Robots UR10*. [Online] Available: <https://www.universal-robots.com/products/ur10-robot/>. Accessed on: February 2018.
- [27] Universal Robots, *UR 10 Technical data*. [Online] Available: https://www.universal-robots.com/media/1514624/101081_199917_ur10_technical_details_web_a4_art03_rls_de.pdf.
- [28] Cars-data, *Peugeot Ion 2010 - 2016*. [Online] Available: <http://www.cars-data.com/en/peugeot-ion-specs/36634>.
- [29] Peugeot, *ION Electric 5-Türer*. [Online] Available: <http://www.peugeot.at/modellreihe/alle-peugeot-modelle/ion/technische-daten.html>.
- [30] Y. Zhang, *Meet Universal Robots-Simple, flexible, affordable*. [Online] Available: <https://www.rnaautomation.com/blog/meet-universal-robots-simple-flexible-affordable/>. Accessed on: February 2018.
- [31] BMW, *Technical data*. [Online] Available: https://legacy.bmw.com/com/de/newvehicles/i/i3/2016/showroom/technical_data.html.
- [32] C. Bennett, *OSI Model and Protocols*. <http://techreviewsandhelp.com/>. [Online] Available: <https://www.youtube.com/watch?v=lmFFc8ih244>.
- [33] Siegfried Vössner, "Engineering-and Business Informatics," Maschinenbau- und Betriebsinformatik, Technische Universität Graz, Graz, Summer Term 2014.
- [34] Zentraler Informatikdienst, *Übergeordnete Netzwerkstrukturen*. [Online] Available: <https://tugnet.tugraz.at/struktur/>.

- [35] National Instruments, *Building Networked Applications with the LabWindows™/CVI UDP Support Library*. [Online] Available: <http://www.ni.com/white-paper/6723/de/>.
- [36] M. Li, *Security in VANETs*. [Online] Available: https://www.cse.wustl.edu/~jain/cse571-14/ftp/vanet_security/index.html.
- [37] S. Heath, *Intelligent Transport Systems*. [Online] Available: <http://slideplayer.com/slide/9088269/>. Accessed on: Dec. 21 2017.
- [38] Rysavy Research, *EDGE, HSPA, LTE: The Mobile Broadband Advantage*. [Online] Available: https://web.archive.org/web/20091007091901/http://developer.att.com/devcentral/tools_technologies/network/docs/DataCapabilities_GPRS__to_HSDPA.pdf. Accessed on: February 2018.
- [39] Surf-Stick.net, *LTE & UMTS Speedtest – Geschwindigkeit messen*. [Online] Available: <http://www.surf-stick.net/umts-speedtest.html>. Accessed on: February 2018.
- [40] International Telecommunication Union, *G.989 : 40-Gigabit-capable passive optical networks (NG-PON2): Definitions, abbreviations and acronyms*. [Online] Available: <https://www.itu.int/rec/T-REC-G.989/en>. Accessed on: February 2018.
- [41] Keba, “KeContact P20/P30 UDP Programmers Guide,” Linz, 2013-2015.
- [42] *RICHTLINIE 2014/94/EU DES EUROPÄISCHEN PARLAMENTS UND DES RATES*, 2014.
- [43] ABB AG, *Ladeinfrastruktur für Elektrofahrzeuge: DC-Ladestation Terra 53 – konform mit vielen Standards*. [Online] Available: [http://www02.abb.com/global/atabb/atabb104.nsf/0/74c33c5b0d0cdc26c1257b36006601ef/\\$file/4EVC204301-LFDE_Terra53+Ladestation.pdf](http://www02.abb.com/global/atabb/atabb104.nsf/0/74c33c5b0d0cdc26c1257b36006601ef/$file/4EVC204301-LFDE_Terra53+Ladestation.pdf). Accessed on: February 2018.
- [44] *Road vehicles — Vehicle to grid communication interface — Part 6: General information and use-case definition for wireless communication*, 2015.
- [45] F. Harvey, *Four of world's biggest cities to ban diesel cars from their centres*. [Online] Available: <https://www.theguardian.com/environment/2016/dec/02/four-of-worlds-biggest-cities-to-ban-diesel-cars-from-their-centres>.
- [46] *CONVENTION ON ROAD TRAFFIC*, 1968.
- [47] E. Gurdus, *Buffett has an interesting theory about why self-driving cars will hurt the insurance industry*. [Online] Available: <https://www.cnbc.com/2017/02/27/buffett-self-driving-cars-will-hurt-the-insurance-industry.html>. Accessed on: February 2018.
- [48] G. Nemeth, *Node Hero - Your First Node.js HTTP Server*. [Online] Available: <https://blog.risingstack.com/your-first-node-js-http-server/>. Accessed on: February 2018.
- [49] Hack Sparrow, *Node.js UDP Server and Client Example*. [Online] Available: <http://www.hacksparrow.com/node-js-udp-server-and-client-example.html>. Accessed on: February 2018.
- [50] T. Miston, *nodejs-tcp-example.js*. Node.js TCP client and server example. [Online] Available: <https://gist.github.com/tedmiston/5935757>. Accessed on: February 2018.

12 Appendix

Appendix 1: Main web-browser program.....	62
Appendix 2: UDP client and server	63
Appendix 3: TCP-socket	64
Appendix 4: Code for the first page of the user interface.....	65
Appendix 5: Code for the second page of the user interface.....	66
Appendix 6: Code for the third page of the user interface	67
Appendix 7: General design of the pages	68
Appendix 8: Design of the page writings	69
Appendix 9: Size of the writings	70
Appendix 10: Arduino C# program for actuator open and close via webserver.....	71
Appendix 11: Charging strategy chart	74

Appendix 1: Main web-browser program

```
File Edit Selection Find View Goto Tools Project Preferences Help
index2.js
1
2 const path = require('path')
3 const express = require('express')
4 const exphbs = require('express-handlebars')
5 const port = 80
6 // This is to send the charging message
7 var chargeComm = require('./sendChargeMessage')
8 var chargeComm2 = require('./sendChargeMessage2')
9 var chargeComm3 = require('./sendChargeMessage3')
10 var chargeComm4 = require('./sendChargeMessage4')
11 var chargeComm5 = require('./sendTCP')
12 const app = express()
13 app.engine('.hbs', exphbs({
14   defaultLayout: 'main',
15   extname: '.hbs',
16   layoutsDir: path.join(__dirname, 'views/layouts')
17 }))
18 app.set('view engine', '.hbs')
19 app.set('views', path.join(__dirname, 'views'))
20 app.get('/', (request, response) => {
21   chargeComm4.chargeMessage()
22
23   response.render('home', {
24     name: 'Free'
25   })
26 })
27 app.get('/car', (request, response) => {
28   chargeComm3.chargeMessage() // Sending out the charging message when the page is called
29   response.render('car', {name: 'Charging'})
30
31
32 })
33 app.get('/select', (request, response) => {
34   chargeComm2.chargeMessage()
35   chargeComm5.chargeMessage()
36   response.render('select', {park1: 'Parking Spot 1', park2: 'Parking Spot 2'})
37 })
38 app.use(express.static('public'))
39
40 app.listen(port, (err) => {
41   if (err) {
42     return console.log('something bad happened', err)
43   }
44
45   console.log(`server is listening on ${port}`)
46 })
```

Defined Port where the Web-browser listens

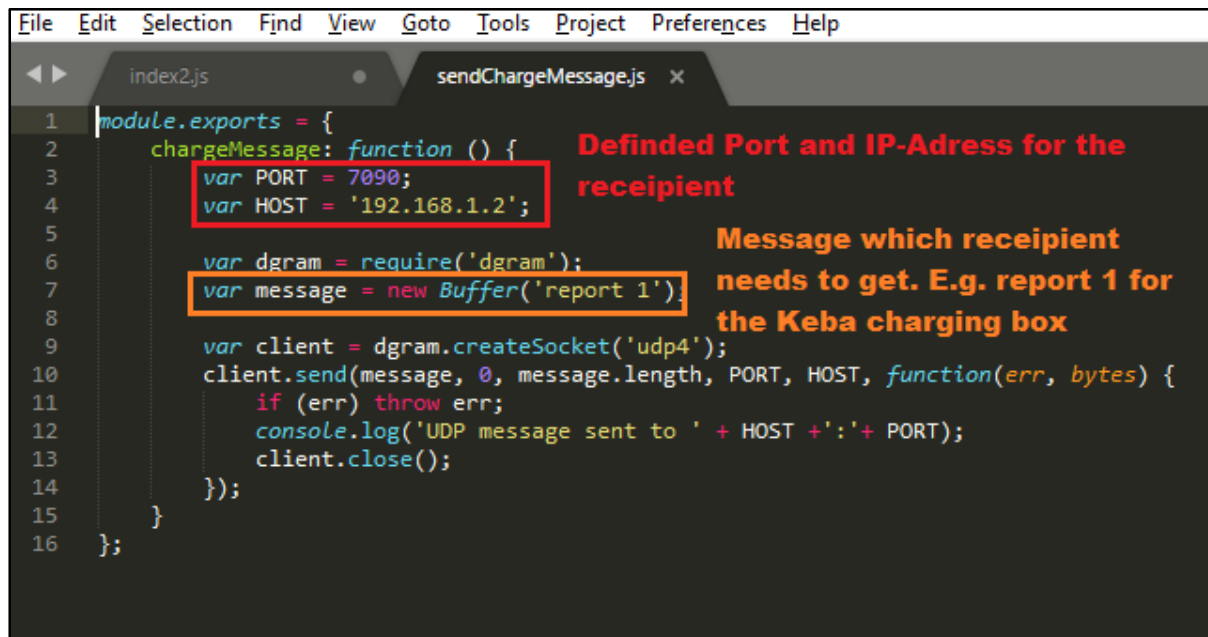
Messages which are send, when the page changes

Defined pages and the message, which is send out when the page is called

Figure 12-1: Main web-browser program [48]

Index 2 is the main page for the web-browser. In this program, we can define the number of pages, the number of messages and the port where the web-browser listens.

Appendix 2: UDP client and server



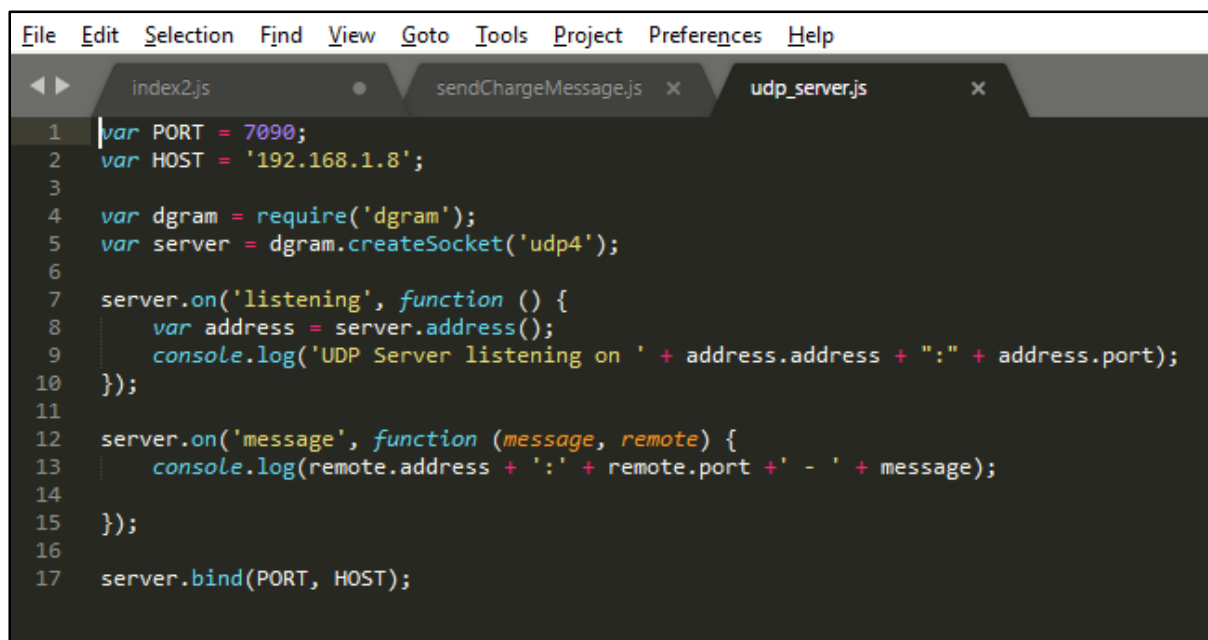
```
File Edit Selection Find View Goto Tools Project Preferences Help
index2.js sendChargeMessage.js x
1 module.exports = {
2   chargeMessage: function () {
3     var PORT = 7090;
4     var HOST = '192.168.1.2';
5
6     var dgram = require('dgram');
7     var message = new Buffer('report 1');
8
9     var client = dgram.createSocket('udp4');
10    client.send(message, 0, message.length, PORT, HOST, function(err, bytes) {
11      if (err) throw err;
12      console.log('UDP message sent to ' + HOST + ':' + PORT);
13      client.close();
14    });
15  }
16 };
```

Definded Port and IP-Adress for the recipient

Message which receiptent needs to get. E.g. report 1 for the Keba charging box

Figure 12-2: UDP Client [49]

This is an example of a UDP-message. This program is also called UDP-client. The figure below shows an example of a UDP-server.



```
File Edit Selection Find View Goto Tools Project Preferences Help
index2.js sendChargeMessage.js x udp_server.js x
1 var PORT = 7090;
2 var HOST = '192.168.1.8';
3
4 var dgram = require('dgram');
5 var server = dgram.createSocket('udp4');
6
7 server.on('listening', function () {
8   var address = server.address();
9   console.log('UDP Server listening on ' + address.address + ":" + address.port);
10 });
11
12 server.on('message', function (message, remote) {
13   console.log(remote.address + ':' + remote.port + ' - ' + message);
14 });
15
16
17 server.bind(PORT, HOST);
```

Figure 12-3: UDP Server [49]

The main difference between these two programs is that the client is just sending messages while the server just gets or listens to messages. The UDP-server is not able to send any messages.

Appendix 3: TCP socket

```
File Edit Selection Find View Goto Tools Project Preferences Help
index2.js sendChargeMessage.js x udp_server.js x sendTCP.js x
1 module.exports = {
2   chargeMessage: function () {
3     var net = require('net');
4     var client = new net.Socket();
5     client.connect(8001, '192.168.1.3', function() {
6       console.log('Connected');
7       client.write('2');
8     });
9     var i = 0;
10    client.on('data', function(data) {
11      console.log('Received: ' + data);
12      i++;
13      if(i==50)
14        client.destroy();
15    });
16    client.on('close', function() {
17      console.log('Connection closed');
18    });
19  }
20 }
21 };
```

Defined Port and IP-Address of the recipient

Message which recipient needs to get

Figure 12-4: TCP-Socket [50]

This is the example of the TCP-message. In this example the socket sends number “2”. Compared to the UDP-Client, the TCP-Client is able to send and receive messages. As already explained in chapter 4, the TCP format opens the connection and waits until he receives the “handshake”, better known as the already mentioned handshake paradigm. The connection stays open until either a limit is programmed or an error happens.

Appendix 4: Code for the first page of the user interface

```
File Edit Selection Find View Goto Tools Project Preferences Help
index2.js home.hbs sendChargeMessage.js udp_server.js
1 <h2>hello the parking lot is <strong>{{name}}</strong></h2>
2
3 <p><a href="select"><h4>Take it</h4></a></p>
4
5
6
7 <p>
8 <strong>Choose a parking space</strong>
9 <select>
10 <option>{{park1}}</option>
11 <option>{{park2}}</option>
12 <option value="third option">Parking lot3</option>
13 </select>
14 </p>
15
16 <p>
17 <a href="select" class="btn btn-primary btn-lg active" role="button">Link</a>
18 </p>
19
20
```

This line defines at which button the page switches. E.g. when you press the TAKE IT button, the page switches to the "select" page

This line is programmed to choose more parking lots

Figure 12-5: First page of user interface

This is the code for the first page of the user interface. As we can see in the red box, the page switches when the button "TAKE IT" is pressed. The orange box is programmed if the customer wants to choose a special spot. Furthermore in the very first sentence you can see that there is a variable in the sentence "hello the parking lot is..". This was done in order to be able to change this variable. For example, if the parking lot is under construction for maintenance reasons the customer would see then the sentence "hello the parking lot is under construction", or if an error occurs like "hello the parking lot is not available".

Appendix 5: Code for the second page of the user interface

```
File Edit Selection Find View Goto Tools Project Preferences Help
index2.js home.hbs select.hbs sendChargeMessage.js udp_server.js
1 <h1>The parking lot is occupied</h1>
2 <p>
3 <strong>Choose a parking space</strong>
4 <select>
5   <option>{{park1}}</option>
6   <option>{{park2}}</option>
7   <option value="third option">Option 3</option>
8 </select>
9 </p>
10
11
12 <p>
13 <strong>Please give us you fully name and E-Mail address, so you can charge your car</strong>
14 <form class="form-inline">
15   <div class="form-group">
16     <label for="exampleInputName2">Name</label>
17     <input type="text" class="form-control" id="exampleInputName2" placeholder="Jane Doe">
18   </div>
19   <div class="form-group">
20     <label for="exampleInputEmail2">Email</label>
21     <input type="email" class="form-control" id="exampleInputEmail2" placeholder="jane.doe@example.com">
22     <a href="mailto:#">placeholder</a>
23   </div>
24   <button type="submit" class="btn btn-default">Send invitation</button>
25 </form>
26 </p>
27
28 <p><a href="car"><h4>Charge</h4></a></p>
29
30
31
32 <p><a href="/"><h5>Abort</h5></a>
33   chargeComm4.chargeMessage()</p>
34
```

Main Text of the page. This page occurs after the TAKE IT button was pushed

Line for the required customer information. In this case E-Mail

The button charge switches to the next page

This line aborts the procedure and switches back to the main page

Figure 12-6: Second page of user interface

This is the code for the second page of the user interface. The red box gives the customers the information that the parking lot is occupied. In this page, the customer needs to fill in the required information in order to continue the charging process, which is marked with the orange box. When he registers himself, the customer is able to continue the process and other users are not able to take this parking lot anymore.

The line in the blue box is responsible for switching to the next place. It should be mentioned that this button is independent from the required customer information. This was done because there was no information pool where the web-browser could get the information in order to continue. However, it is recommended to maybe add the dependency before it can continue to the next page. The dark red box shows the “abort” button, if the customer changes his mind. With this button, the parking lot is free again.

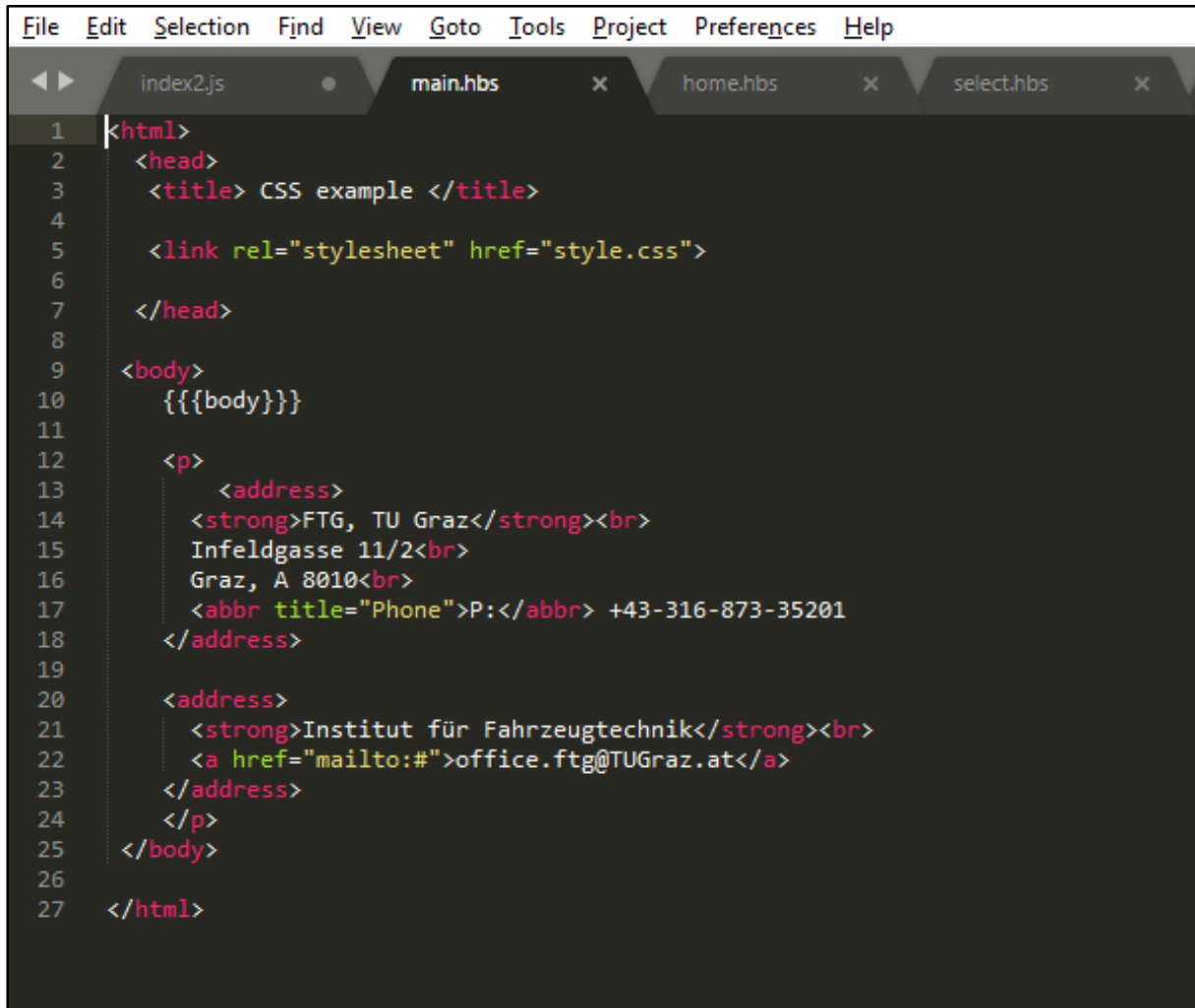
Appendix 6: Code for the third page of the user interface

```
1 | <h3>The car is {{name}} </h3>
2 |
3 | <p>
4 |   <strong>please confirm your Identity to stop the charging</strong>
5 |   <form class="form-inline">
6 |     <div class="form-group">
7 |       <label class="sr-only">Email</label>
8 |       <p class="form-control-static">email@example.com</p>
9 |     </div>
10 |    <div class="form-group">
11 |      <label for="inputPassword2" class="sr-only">Password</label>
12 |      <input type="password" class="form-control" id="inputPassword2" placeholder="Password">
13 |    </div>
14 |    <button type="submit" class="btn btn-default">Confirm identity</button>
15 |  </form>
16 | </p>
17 |
18 | <p><a href="select"><h5>Stop charging</h5></a>
19 |   chargeComm.chargeMessage()
20 | </p>
21 |
22 | <p><a href="/"><h5>Abort</h5></a>
23 |   chargeComm4.chargeMessage()</p>
24 |
```

Figure 12-7: Second page of user interface

This is the code for the third and the last page of the user interface. Similarly to the page before the red box marks the main sentence of this page. As you can see here, the sentence also consists of a variable. This was also written in order to be able to change the word. At the moment the message “*The car is charging*” appears. However, when a data pool of customers is established it can change to the specific car that belongs, according to the information, to the customer. That means that the sentence can change to “*The car is BMW i3*”. The customer can check if the information is right and the maintenance technician can see if everything works properly. The orange box marks the security of this concept. The customer is not able to steer the interface unless he types the password. The blue box shows the line for the manual charging stop, caused by the customer. The violet box marks the abort button, which aborts the entire process and goes back to the first page.

Appendix 7: General design of the pages

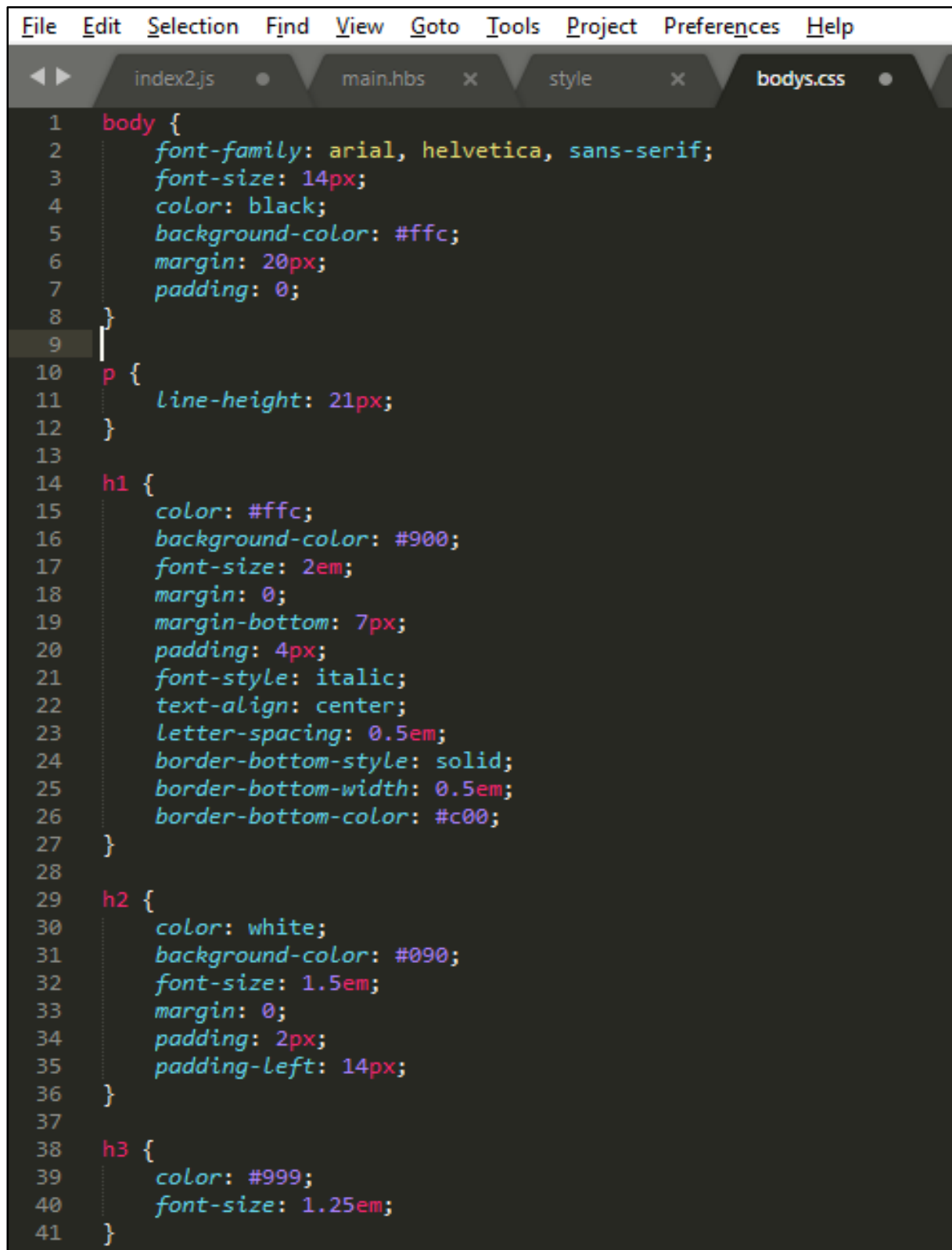
A screenshot of a code editor window with a dark theme. The menu bar at the top includes File, Edit, Selection, Find, View, Goto, Tools, Project, Preferences, and Help. The tab bar shows four tabs: index2.js, main.hbs (selected), home.hbs, and select.hbs. The main editor area displays HTML code for a page template, with line numbers 1 through 27 on the left. The code defines the structure of an HTML document, including a head section with a title and a link to a stylesheet, and a body section containing two address blocks. The first address block includes the text 'FTG, TU Graz' and a phone number. The second address block includes the text 'Institut für Fahrzeugtechnik' and an email address. The code uses standard HTML tags like <html>, <head>, <title>, <link>, </head>, <body>, <p>, <address>, ,
, <abbr>, </abbr>, , , </p>, </body>, and </html>.

```
1 <html>
2   <head>
3     <title> CSS example </title>
4
5     <link rel="stylesheet" href="style.css">
6
7   </head>
8
9   <body>
10    {{{body}}}
11
12    <p>
13      <address>
14        <strong>FTG, TU Graz</strong><br>
15        Infeldgasse 11/2<br>
16        Graz, A 8010<br>
17        <abbr title="Phone">P:</abbr> +43-316-873-35201
18      </address>
19
20      <address>
21        <strong>Institut für Fahrzeugtechnik</strong><br>
22        <a href="mailto:#">office.ftg@TUGraz.at</a>
23      </address>
24    </p>
25  </body>
26
27 </html>
```

Figure 12-8: General design of pages [48]

This page defines all other pages in general. This can be seen especially with the address lines. As you can see the address of the FTG Institute is written, which automatically is shown in all three pages.

Appendix 8: Design of the page writings

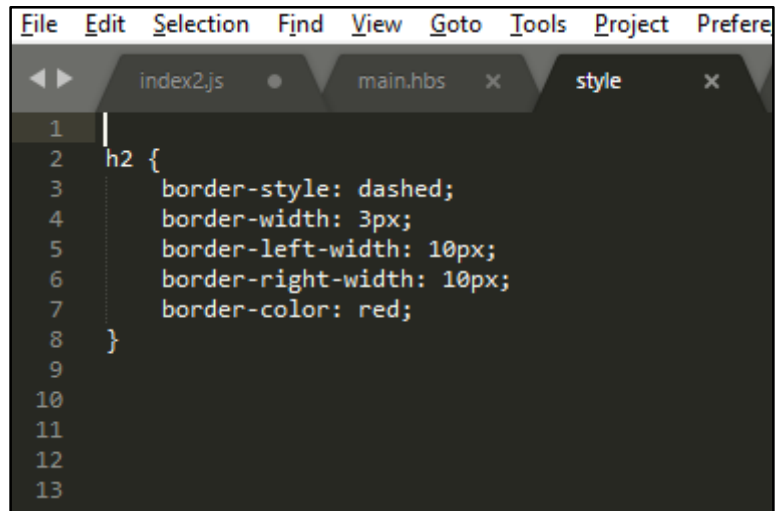
A screenshot of a code editor window with a dark theme. The menu bar at the top includes 'File', 'Edit', 'Selection', 'Find', 'View', 'Goto', 'Tools', 'Project', 'Preferences', and 'Help'. Below the menu bar, there are four tabs: 'index2.js', 'main.hbs', 'style', and 'bodys.css'. The 'bodys.css' tab is active, showing CSS code for styling the body and various heading elements. The code is as follows:

```
1  body {
2      font-family: arial, helvetica, sans-serif;
3      font-size: 14px;
4      color: black;
5      background-color: #ffc;
6      margin: 20px;
7      padding: 0;
8  }
9
10 p {
11     line-height: 21px;
12 }
13
14 h1 {
15     color: #ffc;
16     background-color: #900;
17     font-size: 2em;
18     margin: 0;
19     margin-bottom: 7px;
20     padding: 4px;
21     font-style: italic;
22     text-align: center;
23     letter-spacing: 0.5em;
24     border-bottom-style: solid;
25     border-bottom-width: 0.5em;
26     border-bottom-color: #c00;
27 }
28
29 h2 {
30     color: white;
31     background-color: #090;
32     font-size: 1.5em;
33     margin: 0;
34     padding: 2px;
35     padding-left: 14px;
36 }
37
38 h3 {
39     color: #999;
40     font-size: 1.25em;
41 }
```

Figure 12-9: Design of page writings [48]

This program defines the size and colour of the writings of the pages. The variables h1, h2 and h3 for example had to be written at the sentence start and the end of the sentence. Therefore, the entire sentence can be designed with this program. You can see these variables in the appendixes above.

Appendix 9: Size of the border

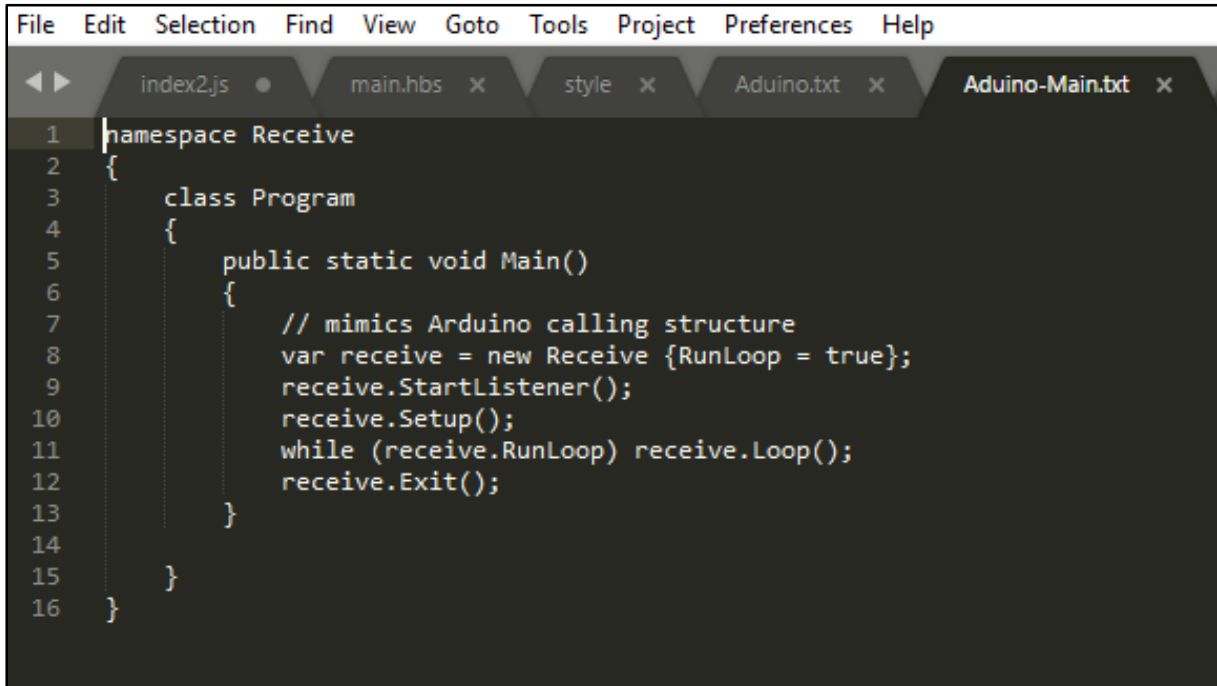
A screenshot of a code editor window with a dark theme. The menu bar at the top includes 'File', 'Edit', 'Selection', 'Find', 'View', 'Goto', 'Tools', 'Project', and 'Prefere'. The tab bar shows three tabs: 'index2.js', 'main.hbs', and 'style'. The 'style' tab is active, displaying CSS code for an 'h2' element. The code is as follows:

```
1  
2 h2 {  
3     border-style: dashed;  
4     border-width: 3px;  
5     border-left-width: 10px;  
6     border-right-width: 10px;  
7     border-color: red;  
8 }  
9  
10  
11  
12  
13
```

Figure 12-10: Size of borders [48]

This program defines the box size or the border size of the buttons. These lines are more for fashion purposes than for practical ones.

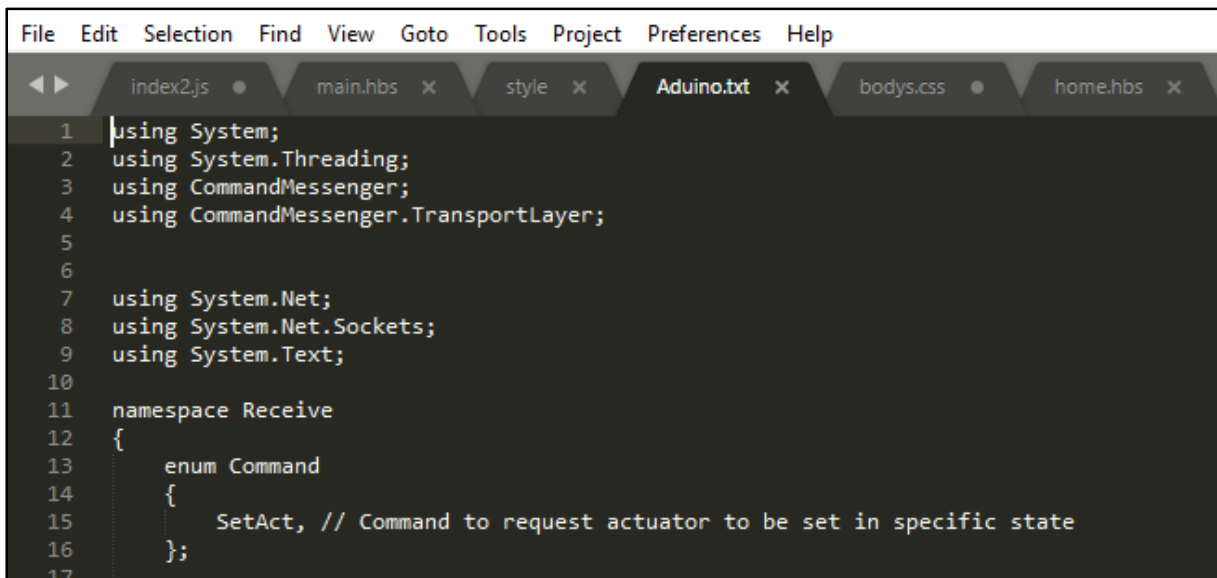
Appendix 10: Arduino C# program for actuator open and close via webservice



```
File Edit Selection Find View Goto Tools Project Preferences Help
index2.js main.hbs x style x Aduino.txt x Aduino-Main.txt x
1 namespace Receive
2 {
3     class Program
4     {
5         public static void Main()
6         {
7             // mimics Arduino calling structure
8             var receive = new Receive {RunLoop = true};
9             receive.StartListener();
10            receive.Setup();
11            while (receive.RunLoop) receive.Loop();
12            receive.Exit();
13        }
14    }
15 }
16 }
```

Figure 12-11: Main class

As you can see, this program exists in order to call the defined commands. The next graph shows the defined framework which the program need to have, also it steers the actuator to be in a defined state, so you can always start the actuator from the same position.



```
File Edit Selection Find View Goto Tools Project Preferences Help
index2.js main.hbs x style x Aduino.txt x body.css home.hbs x
1 using System;
2 using System.Threading;
3 using CommandMessenger;
4 using CommandMessenger.TransportLayer;
5
6
7 using System.Net;
8 using System.Net.Sockets;
9 using System.Text;
10
11 namespace Receive
12 {
13     enum Command
14     {
15         SetAct, // Command to request actuator to be set in specific state
16     };
17 }
```

Figure 12-12: Program to set act

The figure below shows how the message is imported and the resulting actions.

```

public class Receive
{
    public bool RunLoop { get; set; }
    private SerialTransport _serialTransport;
    private CmdMessenger _cmdMessenger;
    private int _ActState;
    private string input;

    // Teil: UDP Message
    private const int listenPort = 7090;

    public void StartListener() // Method to import the Message
    {
        bool done = false;

        UdpClient listener = new UdpClient(listenPort);
        IPEndPoint groupEP = new IPEndPoint(IPAddress.Any, listenPort);

```

Figure 12-13: Method of importing the message

```

try
{
    while (!done)
    {
        byte[] bytes = listener.Receive(ref groupEP);

        input = Encoding.ASCII.GetString(bytes, 0, bytes.Length); // Import of Message

        if (input == "1") // if 1 is imported -> Aktuator is pulling in and Loop is left
        {
            break;
        }
        if (input == "2") // if 2 is imported -> Aktuator is pulling out and Loop is left
        {
            break;
        }
    }
}
catch (Exception e)
{
    Console.WriteLine(e.ToString());
}
finally
{
    listener.Close();
}
}

```

Figure 12-14: Import of message

The figure below shows the end of the loop and the readout.

```

}
catch (Exception e)
{
    Console.WriteLine(e.ToString());
}
finally
{
    listener.Close();
}
}

```

Figure 12-15: Output of string message

The next figure shows the Arduino part. This section sends the commands to Arduino and this is the crucial step for the steering.

```
// Teil: Arduino
public void Setup()
{
    // Create Serial Port object
    _serialTransport = new SerialTransport();
    _serialTransport.CurrentSerialSettings.PortName = "COM3"; // Set com port!!!
    _serialTransport.CurrentSerialSettings.BaudRate = 9600; // Set baud rate!!!
    _serialTransport.CurrentSerialSettings.DtrEnable = false; // For some boards (e.g. Sparkfun Pro Micro) DtrEnable may need to be true.

    // Initialize the command messenger with the Serial Port transport layer
    _cmdMessenger = new CmdMessenger(_serialTransport);

    // Tell CmdMessenger if it is communicating with a 16 or 32 bit Arduino board
    _cmdMessenger.BoardType = BoardType.Bit16;

    // Attach the callbacks to the Command Messenger
    AttachCommandCallbacks();

    // Start listening
    _cmdMessenger.StartListening();
}
```

Figure 12-16: Arduino part

```
// Loop function
public void Loop()
{
    _ActState = Convert.ToInt32(input); // eingelesener Wert wird in int umgewandelt
    // Create command
    var command = new SendCommand((int)Command.SetAct, _ActState);

    // Send command
    _cmdMessenger.SendCommand(command);

    StartListener(); // Listener is start again
}
```

Figure 12-17: Loop function

The last figure shows the end of the function and the exit of the program.

```
// Exit function
public void Exit()
{
    // We will never exit the application
}

/// Attach command call backs.
private void AttachCommandCallbacks()
{
    // No callbacks are currently needed
}
}
```

Figure 12-18: Exit function

Appendix 11: Charging strategy tables

Step	Target/Goal	Program-communication	Communication-protocols
1	Driver/User wants to start charging process	User-webserver-charging station	TCP
2	Registration and Identification	User-Webserver-Charging station	UDP/TCP
3	Process start- Vehicle drives to the parking lot	User-server	-
4	Vehicle stands at parking lot	Halcon-server	TCP
5	Vehicle-identification	Halcon-Matlab(ROS)-server	TCP
6	Vehicle-Pose-Identification	Halcon-Matlab(ROS)-server	TCP
7	Loading cover-Pose-Identification	Halcon-Matlab(ROS)-server	TCP
8	Opening the loading cover	Halcon-Matlab(ROS)-Arduino	TCP
9	Charging socket-Pose-Identification	Halcon-Matlab(ROS)	TCP
10	Attach procedure	Halcon-Matlab(ROS)-robot	TCP
11	Plug procedure	Halcon-Matlab(ROS)-charging box	UDP+TCP
12	Charging procedure (Start/End)	User-webserver-charging box	UDP
13	Charging plug returns	Matlab(ROS)	TCP
14	Closing the loading cover	Matlab(ROS)-Arduino	TCP
15	Robot goes to starting point	Matlab(ROS)	TCP
16	Charging procedure finished	Webserver-user	UDP+TCP
17	Vehicle leaves parking lot	User-server	-

Table 12-1: Charging strategy part 1

Data	Data formats
Take it, which parking lot is free(if more possibilities)	String
User: name, e-mail, user data, vehicle: charging station position	String
GPS-data, current vehicle position	GPS
Pictures, videos	RGB
Pictures, videos, CAD-data, user data, vehicle data	RGB, STL, String
Pictures, videos, CAD-data, user data, vehicle data	RGB, STL, String
Pictures, videos, CAD-data, user data, vehicle data	RGB, STL, String
Pictures, data for actuator	RGB, STL, String
Pictures, data for actuator	RGB, STL, String
Robot steering data	Position data
Charging- data, state	String
Charging- state, order	String
Pictures, videos, CAD-data	RGB, STL, String
Pictures, data for actuator	RGB, STL, String
Pictures, videos, CAD-data, robot steering data	RGB, STL, String
Parking lot free	String
GPS-data, current vehicle position	GPS

Table 12-2: Charging strategy part 2