Charging Station for ISO / IEC 15118 Protocol

*Building working smart networked charging station with support for both ISO 15118 and IEC 61851*

Bachelors Project

**Presented By**

Jiztom Kavalakkatt Francis

Nivas Gokul Manimurugesan

Raphael Scholz

**Supervisor** Prof. Dr.-Ing. Ansgar Meroth, Hochschule Heilbronn

University of Heilbronn

Faculty of Mechanics and Electronics

Automotive Engineering

**content**

**Table Directory**

**List of Figures**

1. **introduction**

In this chapter a short summary of this work is described. In addition, the task description as well as an overview of the following chapters and their contents are given.

* 1. **short version**

The present project work serves to set up a demonstrator for the current standardization of the vehicle-column communication according to ISO / IEC 15118.

The built-up demonstrator consists of two interconnected boards. Each communication user is represented by a circuit board.

The software used is derived from an existing stack and adapted for a defined application of ISO / IEC 115118.

According to the ISO variable parameters are listed as macros and documented, so that a change is possible at any time. If parameters are selected so that the resulting requirements can no longer be met by the respective other subscriber, a fault message is output.

* 1. **Work position**

The project is expected to result in a structure that demonstrates the communication staple of ISO / IEC 15118.

For this purpose an existing stack for the ISO is put into operation. The freely available stack is written so far for demonstration purposes of the software. For use at a demonstrator, it must be separated, reordered and modularized so that a stack for the vehicle as well as the load column side is created. The initially defined load parameters are defined as macros, so that a change is possible at any time. Furthermore, incorrect error messages are to be output in the event of incorrect inputs or not supporting functions of the charging column or the vehicle.

At the same time, a development environment to be used for demonstration purposes is to be established. First of all, hardware requirements must be defined. From the resulting specifications, approaches for implementation have to be found. The resulting results are evaluated and selected according to the previously defined viewpoints.

As soon as the selected hardware is ready for operation, it must be commissioned and all functions required for commissioning must be checked.

Finally, the functional stack is flashed to the development environment and tested for functionality.

The resulting project results are defined as follows:

* Integration into vehicle charging columns Communication according to ISO / IEC 15118
* Separation of an existing stack in the vehicle and loading column side
* Commissioning and demonstration of the separate stack
* Selection and commissioning of a hardware suitable for demonstration
* Start up the demonstrator with the previously created communication pack
* documentation of the results
  1. **Chapter overview**

In *Chapter* [1](https://translate.googleusercontent.com/translate_f#_Ref403389681) a summary of this work and the task and this chapter overview are displayed.

*Chapter* [2](https://translate.googleusercontent.com/translate_f#_Ref433123920) provides an overview of previous work at the university as well as documents that have contributed to the incorporation into the subject. Furthermore, hardware and software components used are described.

An overview of the hardware requirements and the commissioning is in *Chapter* [3](https://translate.googleusercontent.com/translate_f#_Ref450304497) to find. In addition, an overview of the work with the Linux terminal is given. This is important for communication with the board, especially in the field of data management, as well as for compiling and starting a program.

In Chapter [4](https://translate.googleusercontent.com/translate_f#_Ref446328597) , the various parts of the created and edited the software will be described. It contains newly created files for communication within the board as well as transferred program parts for communication according to ISO / IEC 15118. Newly created components contain functions which are particularly important for signal generation according to IEC 61851. For the data exchange of the communication users according to ISO / IEC 15118, a connection setup is also necessary, which is also presented in this chapter.

The results achieved by this work are discussed in *Chapter* [5](https://translate.googleusercontent.com/translate_f#_Ref450304510) reproduced. Recently a summary of the work and an outlook of the project in *chapter* [6](https://translate.googleusercontent.com/translate_f#_Ref446328613) reproduced.

Parts of ISO / IEC 15118, together with program schedules in *Chapter* [7](https://translate.googleusercontent.com/translate_f#_Ref450304514) shown. In both cases, this is done in order to get an overview of the requirements according to DIN / ISO and the resulting programs.

1. **State of the art**

The chapter provides an overview of the work and information on which the project is working. This includes, among other things, the previous way of charging a vehicle, as well as the previous information exchange, various vehicle connectors used for charging electronic vehicles. Furthermore, a study work which is to be described in more detail in ISO 15118 as well as a dissertation by Dr.-Ing Marc Mültin, which deals with the electric vehicle as a "flexible consumer and energy storage in the Smart Home".

* 1. **IEC 62196: Vehicle connector**

Connector types and Lademodis of electric vehicles by the International Electrotechnical Commission in IEC 62196 defines [(Wiki\_Stecker, 2016)](https://translate.googleusercontent.com/translate_f#Wiki_stecker) .

The second part of the standard was released in 2011 and includes various connector types. This contains three of the most commonly used charge plugs at this time.

The Type 1 charging plug, which in [Figure 2.1](https://translate.googleusercontent.com/translate_f#_Ref446330588) is shown, takes its specifications from the SAE J1772. This was first published in 1996 by the Society of Automotive Engineers and has since been expanded and maintained by the Society of Automotive Engineers. The disadvantage of this type of plug is found in the contacts, since these do not allow a three-phase charging with alternating current.

Type 2 of the standard charging plug is the currently the most built-up type of charging plug systems and also in [Figure 2.1](https://translate.googleusercontent.com/translate_f#_Ref446330588) to find. The connector finds its origins through a collaboration of the plug manufacturer Mennekes with the power supply RWE as well as the automobile manufacturer Daimler. The name of the Mennekes plug is thus obtained by its manufacturer.

The third plug-in type plugged into the standard, the EV Plug Alliance, was defined by a consortium led by French and Italian companies. Due to the low demand, the further production of the plug was discontinued.

The signal connectors CP (ControlPilot) and PP (ProximityPilot) defined in Type 1 are included for all defined connector types. Which allows a charging process according to IEC 61851.

**Figure 2.1: charging plug systems**[**(Mültin, 2014)**](https://translate.googleusercontent.com/translate_f#Mültin)

The in [Figure 2.1](https://translate.googleusercontent.com/translate_f#_Ref446330588) signal contacts shown are defined as follows:

|  |  |  |
| --- | --- | --- |
| abbr ürzung | Contact | function |
| CP | Control Pilot | Control signals Lades Äule  electric vehicle |
| PP | Proximity Pilot | The presence of a charging cable pr üfen |
| N | Neutral | F or AC charging |
| L1, L2 and L3 | Current conductionphases ührende | For alternating current charging with one (L1 / L) or three (L1, L2, L3) phases |
| PE | Protective Earth | Protective conductor |
| DC +/- | Current conductionphases ührende | For DC charging |

**Table 2.1: Contact names**

* 1. **IEC 61851: Charging process**

Prior to the definition of a charging process according to ISO / IEC 15118, the charging parameters required for the charging process were defined using a PWM signal according to IEC 61851. To determine the required parameters for charging the signals of the in *chapter* [2.1](https://translate.googleusercontent.com/translate_f#_Ref446582293) described contacts Control Pilot (CP), Protective Earth (PE) and Proximity pin (PP) required.

For charging the vehicle, both communication subscribers are first connected to one another. A 1kHz signal with 12V is generated on the CP contact from the side of the charging column. The pulse width of the signal indicates which maximum power can be provided by the charging column. In this case, 10% max. 10A, 25% 16A, 50% max. 32A and 90% quick charge [(Wiki\_Stecker, 2016)](https://translate.googleusercontent.com/translate_f#Wiki_stecker) .

On the vehicle side, resistors are connected between CP and PE or PP and PE. Through different switchable level of tension between the CP and PP contact corresponding [Figure 2.2](https://translate.googleusercontent.com/translate_f#_Ref446585121) as well different charge states are given. Please note that the negative voltage value is permanently -12V, and only the positive values ​​change. A definition of each state is [Table 2.2](https://translate.googleusercontent.com/translate_f#_Ref446585976) .

Lastly, a vehicle-side resistance between the PP and the PE contact indicates the maximum possible charging current of the electric vehicle. The greater the resistance used, the lower the maximum charging current. Specifically, for a 1.5kΩ resistor, a maximum charging current of 13A, a maximum of 20A with a resistance of 680Ω, at 220Ω the maximum charging current 32A and 63A is at 100Ω.

**Figure 2.2: Voltage CP-PE**[**(Lewandowski, 2014)**](https://translate.googleusercontent.com/translate_f#Lewandowski)

|  |  |  |
| --- | --- | --- |
| level | State | Condition description |
| 12 ± 1 V | State A | Electric vehicle not connected |
| 9 ± 1 V | State B | Electric vehicle connected, not ready for charging |
| 6 ± 1 V | State C | Electric vehicle connected, ready for charging |
| 3 ± 1 V | State D | Electric vehicle connected, ready for charging, ventilation necessary |
| 0 ± 1 V | State E | Power problem, PP short circuit against ground |
| -12V | State F | Vehicle not avail ügbar, Error |

**Table 2.2: states CP-PE**[**(Wiki\_Stecker, 2016)**](https://translate.googleusercontent.com/translate_f#Wiki_stecker)

* 1. **ISO / IEC 15118**

The International Organization for Standardization (ISO) and the International Electronic Commission (IEC) in 2009 started to describe the standardization of a "digital IP-based communication protocol" between electric vehicle and charging station [(Mültin, 2014)](https://translate.googleusercontent.com/translate_f#Mültin) . This should be a "plug-and-batch" mechanism for authentication, authorization, billing as well as load control so that the parameters required for the charge release are stored in the vehicle and the user only has to connect the two communication parties together. The individual communication Content will be the level of tension control pin signal of Chapter [2.2](https://translate.googleusercontent.com/translate_f#_Ref448342277) in accordance with [Figure 2.3](https://translate.googleusercontent.com/translate_f#_Ref448342461) associated.

**Figure 2.3: Assignment of the communication contents to the different charging states**

The full schedule of communication stacks for AC or DC charging an electric vehicle according to ISO / IEC 15118 can be found in [Figure 7.1](https://translate.googleusercontent.com/translate_f#_Ref447825795) to [Figure 7.4](https://translate.googleusercontent.com/translate_f#_Ref447825798) . An overview of the variables contained in the messages within the AC communication stack is described together with an overview of the ISO / IEC 15118 in a previous study work [(Barth, 2015)](https://translate.googleusercontent.com/translate_f#Barth) .

* + 1. **Open V2G project**

An already far ahead paced example for implementing a communication stack to ISO / IEC 15118 has already been through the support of Siemens Corporate Technology as an open source project initiated [(OpenV2G, 2016)](https://translate.googleusercontent.com/translate_f#OpenV2G) . Both Ladesäulen- and side of the vehicle are shown in a program code and the message corresponding to [Figure 2.4](https://translate.googleusercontent.com/translate_f#_Ref447829449) generated sequentially checked and the next message is generated. At the current status (version 0.9.3), the sequence of the individual requests and responses as well as the message contents to the direct current and alternating current charge can be derived very well. It is one of the objectives of this thesis to divide this code into a program for every communication user.

**Figure 2.4: Program flow inside the V2A codes**[**(OpenV2G, 2016)**](https://translate.googleusercontent.com/translate_f#OpenV2G)

* 1. **Powerline communication**

General power line communications (PLC) is described as "transmission of data over the power cord" [(El-Ko, 2016)](https://translate.googleusercontent.com/translate_f#PLC_Komp) . In this case, this form of signal transmission has already been used, for example, in intercom systems. The use of this communication variant means that no further pins and lines have to be defined and retrofitted in the existing ladder connectors and cables.

In the case of powerline communication, the data to be transmitted are modulated on the power cable in the high-frequency range and demodulated on the receiver side. In the case of ISO / IEC 15118, the modulation and demodulation of the communication protocol according done [Figure 2.5](https://translate.googleusercontent.com/translate_f#_Ref448349602) on the PWM signal of the CP-conductor.

**Figure 2.5: modulated PWM signal**

* 1. **Automatic state machines**

An automatic state machine consists of states, state transitions and actions. The purpose of these tools is to implement the control of a system that takes into account past, present and future events. Each state is associated with actions that occur when it is entered or exited. A state must be defined at any time during the runtime of the system. A state transition, on the other hand, describes the connections of the individual states to one another as well as the event which must occur in order to switch between the states.

**Figure 2.6: Example of a state machine**[**(Bustamante, 2014)**](https://translate.googleusercontent.com/translate_f#Zustandsautomat)

To illustrate such a state machine, a coffee machine is used in accordance with [Figure 2.6](https://translate.googleusercontent.com/translate_f#_Ref449990101) . The state machine starts with the start state, which in the present example is the switched off state of the coffee machine. Here, a status change is only possible by the switch-on transition. Depending on the user's input, the machine can be set to "Espresso ready", "Steamer ready" or "OFF".

This example shows particularly well the inclusion of different time forms. To be ready for operation, the coffee machine had to be switched on in advance and brought to a defined temperature. Which state is assumed in the further course depends on unforeseeable events. It is also clearly shown that the machine can not activate the individual states at any time.In order to be able to assume a particular state, this must be connected to that of a state transition from the current state. Thus, in the given example, no coffee can be prepared as long as the coffee machine is in the "ON" state.

 The programming of an automatic state machine can be implemented with the switchcase function. A basic state is already defined in advance. As soon as an event that might cause a state change occurs, the function is started. The currently defined state is queried and the state change is defined in the corresponding case in order to reach a new state. An example to the coffee machine [Figure 2.6](https://translate.googleusercontent.com/translate_f#_Ref449990101) shows [Figure 2.7](https://translate.googleusercontent.com/translate_f#_Ref450076151) .

**Figure 2.7: Implementing an automatic state machine**

1. **hardware**

A component of the present work is the construction of a demonstrator. For this reason, a board is required for the hardware on both the vehicle and the load column side, which can generate a signal according to IEC 61851 and at the same time can build up a powerline-based communication.

* 1. **Requirements / characteristics**

A total of two boards are therefore required for the realization of the demonstrator. One is to be used for the construction of a load column detector and the second one for the electric vehicle. Both boards will need to demonstrate in *Chapter* [2.3](https://translate.googleusercontent.com/translate_f#_Ref448685045) pins required for communication processing according to ISO / IEC 15118th Furthermore need lines for controlling the power supply and the response to the locking of the charging plug to prevent withdrawal during charging, be present.

For communication in accordance with IEC 61851, the load column must be able to generate a PWM signal and to change the positive levels of the latter. Furthermore, this communication user must be able to measure the levels to the other pins. The vehicle side needs the positive voltage as well as the pulse width of the vehicle to determine the current charge state to determine the maximum charging current. Furthermore, resistors can be connected between the CP, PP and PE contacts for the feedback signal.

Another important feature of this project is the possibility of powerline communication. This must be supported from the load column and the electric vehicle side in order to modulate the messages according to ISO / IEC 15118 to the signal of the control pin.

* 1. **EVAchargeSE**

The EVAchargeSE board of I2SE was developed for communication according to ISO / IEC 15118. The advantage of the board is that it can be used on both the vehicle and the load column side. This allows two identical boards with different configurations to be used for the demonstration set-up.

On the board an i.MX28 is installed with a Linux operating system. The two GPIOs CP, PP and PE required for communication are connected to a plug. Programs for the communication pack can be transmitted using the existing Ethernet interface. These are stored in the 4GB memory and started from there.

A KL02 microcontroller contained on the board can be addressed by the i.MX28 via UART. In the master / slave connection, the KL02 is addressed as a slave and is responsible for generating and measuring the signals according to IEC 61851. It is also used to address GPIOs, which should start a locking mechanism of the charging cables.

Furthermore, the EVAchargeSE board is PLC-capable. The messages to be sent are sent over TCP / IP sockets. A QCA 7000 chip modulates the transmitted data in the further course to the PWM signal of the KL02.

**Figure 3.1: EVAchargeSE board**[**(I2SE, 2016)**](https://translate.googleusercontent.com/translate_f#I2SE)

* + 1. **construction**

For the commissioning of the circuit boards, they are first set up in the laboratory. The voltage supply is 12V at 0.3A. For communication between the boards EVAchargeSE the terminals of CP, PP and PE according to [Figure 3.2](https://translate.googleusercontent.com/translate_f#_Ref448952340) interconnected. Furthermore, the configuration shows an Ethernet connection to a router and a computer with a Linux operating system. This allows access to the Linux systems that reside on the boards. At this point, this access is useful because it allows programming in an environment like Eclipse on a computer. The programs are first started on the computer using Eclipse to check the runnability. In the further course, the programs for vehicle and loading column sides are flashed, compiled and started from the respective boards. Detailed instructions for this is in Chapter [3.3](https://translate.googleusercontent.com/translate_f#_Ref448952802) to find.

**Figure 3.2: Connection of the information lines**

* + 1. **Create serial\_Programming directories**

The serial\_Programming directory is responsible for the communication between the i.MX 28 and the KL02, which is responsible, inter alia, for the generation of the PWM signal.This makes the settings of the microcontroller important when distinguishing between the electric vehicle and the charging station. As noted in Section [3.2](https://translate.googleusercontent.com/translate_f#_Ref449040722) above, the communication between the two microcontrollers via UART. The i.MX 28 is declared as master and the KL02 as slave. This means that the KL02 waits for and responds to a command request from the ARM processor. The contents of the transmitted messages are organized into frames. This has the advantage of a small amount of data to be transmitted.A frame is in this case according to [Figure 3.3](https://translate.googleusercontent.com/translate_f#_Ref449132046) , both vehicle and charging station side together. The parameters contained therein are in [Table 3.1](https://translate.googleusercontent.com/translate_f#_Ref449982939) defined.

**Figure 3.3: Message frames between i.MX and KL02**[**(I2SE, 2016)**](https://translate.googleusercontent.com/translate_f#I2SE)

|  |  |
| --- | --- |
| Name in the frame | description |
| STX | Displays the start of a new frame |
| LGE | Specifies the number of bytes contained in ADR, Service, Data, and BBC |
| ADR | Issue Zielger DEVICE. |
| service | Describes the durchzuf ührenden service according to [Table 3.2](https://translate.googleusercontent.com/translate_f#_Ref449209247) |
| Data 1 ... n | Senior Citizen contains the required parameters for performing the services or returns Requested values |
| BCC | Pr üfsumme: To avoid transmission errors.  Is calculated by connecting all bytes contained in the frame to each other by a logical XOR. |

**Table 3.1: Description of the parameters contained in the frames**

|  |  |
| --- | --- |
| Service ID | Service description |
| 0x01 | Test the connection. Has R ückgabewert software and H ardwareversion of EVAchargeSE Boards |
| 0x04 | Further test the connection. Contains Senior Citizen as a return value, among other things the last reason a reset. A list of possible reasons can be found in the data sheet. |
| 0x10 | Measures the frequency of the PWM signal |
| 0x11 | Changes the frequency of the PWM signal |
| 0x12 | Defines whether a PWM signal is generated.  This setting is important for the distinction between the electric vehicle and the load column because the signal may only be generated on the column side of the load column. |
| 0x14 | Measures both the positive and negative levels of the CP pin |
| 0x15 | Changes the positive level of the CP signal. This service may only be called up on the column side, since this parameter is important on the vehicle side for the determination of the current charge state. |
| 0x17 / 0x18 | Speaks pins for controlling motors for charging cable locking |
| 0x1A | Indicates the current status of the charging cable latching motors |
| 0x20 | Enables an automatic measurement and R ückgabe the PWM signal |
| 0x31 | Ensures f or an interruption of the connection |
| 0x33 | Initiates a restart of the KL02 |
| 0x50 | Defines the value of the resistor between PP and PE |
| 0x51 | Enables the resistance between PP and PE. This service may only be activated on the vehicle side. This defines the maximum charging current of the electric vehicle. |
| 0x52 | Measures the voltage between PP and PE |

**Table 3.2: Description of the services for communication between i.MX28 and KL02**

* 1. **Communication between computer and EVAchargeSE**

On the board is a Debian operating system installed by the manufacturer of the board I2SE. This "uses the Linux operating system kernel, but the most basic OS tools come from the GNU Project" [(Debian, 2016)](https://translate.googleusercontent.com/translate_f#Debian) . In order to facilitate a communication setup, the Linux system Ubuntu is used exclusively for the work with the EVAchargeSE. The terminal allows the user, after a short training period, to access, quickly, and exchange information to other connected users of the router.

* + 1. **Flashing a program**

Before a program can be started, it must be flashed on the microcontroller, on which it is to be compiled and executed in the future.

The Secure File Transfer Protocol (SFTP) could be used for programs with few files. This protocol runs on the basis of secure shells and establishes a secure connection for the transmission of data. In this case, data can be transferred to the EVAchargeSE boards using SFTP using a Linux terminal.

The command "sftp <Username> @ <IP address>" is used to start an SFTP connection. If a subscriber is accessible with the identification data, a password input is required. Once this is correct, the commands initiated in the terminal are executed by the user of the device of the specified IP address.

Thus, the current directory on the board can be changed by means of the command "cd". If, however, a different directory is to be called on the Linux system of the computer, the command is preceded with an exclamation point before the command ("! Cd").

You can use the "put <file name>" command to copy files from the Linux machine to the board. A copy of a file existing on the board can be written to the computer directory by the command set "get <filename>".

Due to the increasing number of files, such a procedure for transferring the individual files is very time-consuming. For this reason, data will be transferred later in the program using FileZilla. The FTP client allows a simple and clear transmission of several data files. At the same time, the graphical user interface gives the user the possibility to get an overview of the existing folder structures and directories. As a result, local files as well as those of the connected communication partner can be found and transmitted from various folders in a short time.

To transfer data, the program is started and the IP address, username, password, and port of EVAchargeSE boards according [Figure 3.4](https://translate.googleusercontent.com/translate_f#_Ref449382982) registered in the appropriate fields. As soon as a connection is established, next to the folder structure of the local computer, this also appears for the board connected to the computer. By dragging and dropping files can be exchanged in both directions of the link [(FileZilla, 2016)](https://translate.googleusercontent.com/translate_f#FileZilla) .

**Figure 3.4: FileZilla**

* + 1. **Compiling and running a program**

To compile and run a program, you must first establish a connection between the local computer and the respective development board. A secure shell (SSH) is used to make locally available a remote command line available. This is a network protocol that establishes an encrypted network connection [(Wiki\_SSH, 2016)](https://translate.googleusercontent.com/translate_f#Wiki_SSH) .

The function call to bring up such a compound is similar to that in Chapter [3.3](https://translate.googleusercontent.com/translate_f#_Ref448952802) . First, the secure shell is set up with the call "ssh <Username> @ <IP address>" in the Linux terminal. For security reasons, a password request is also carried out in the next step.

Both usernames (root) and passwords (zebematado) are used for both boards. However, the IP addresses must have differences (192.168.37.250 and 192.168.37.251). The password is correct the current date and the last logged in is accordingly [Figure 3.5](https://translate.googleusercontent.com/translate_f#_Ref449468752) appears. The Linux terminal in which this call was executed now represents a command line of the EVAchargeSE board.

**Figure 3.5: Successful SSH connection**

To compile the previously transmitted C-code must first be coordinated into the corresponding directory. Overview of the current directory path and navigation in other directories, the commands "pwd" and "cd" to corresponding [Table 3.4](https://translate.googleusercontent.com/translate_f#_Ref449471226) used. The command "gcc -o <NAME> \* .c" is called for the EVAchargeSE board to compile the sourcefiles in the directory. The variable <NAME> used here can be named as desired and contains the start file. To start the compiled code, the previously defined start file is called by "./ <NAME>". A Linux terminal, which compiled in the serial\_Programming this directory and Starts is in [Figure 3.6](https://translate.googleusercontent.com/translate_f#_Ref449473049) to see.

If an executable file is to be interrupted, this can be done using the key combination "Ctrl" and "C". Once the work in the terminal of the development board is completed, the SSH connection can be terminated by "exit".

**Figure 3.6: Compiling and running a program**

* 1. **Problems encountered and troubleshooting**

There are problems during commissioning of the hardware used, which are not described in the data sheet of the boards. For traceability and error prevention, the most important are described in the following section.

In the first step to commissioning the boards, a connection is established between a PC and a board. The structure of this is accordingly [Figure 3.2](https://translate.googleusercontent.com/translate_f#_Ref448952340) realized with only one board.When attempting to other participants connected to the router using the "ping" command according to [Table 3.4](https://translate.googleusercontent.com/translate_f#_Ref449471226) to address, the Linux computer does not receive a response from the board. The reason for this is found in the IP of the router and the EVAchargeSE-board. Since the basic setting of the router provides an IP address in a different subnet. By changing the IP address of the router, the network users can recognize and the "ping" command gives the computer a response from the board.

A further problem can be found in an undefined response service of the serial communication. In the event of an error in the transmission of the frame from master to slave, the latter returns the non-documented response service 0x99 as a response. In this case, the data block 0x43 can be recognized for an error of the checksum or 0x44 for an unknown error as the reason for the error.

In order to communicate the boards with each other, a mutual ping is initiated via the Powerline communication interface. As the two participants could not find the I2SE, the Council was consulted. By in [Table 3.3](https://translate.googleusercontent.com/translate_f#_Ref449216106) described command "PLCSTAT -i qca0 -t" was not also possible a recognition.By reading the parameter information block both Boards could be determined that both boards have been initialized with an identical MAC address. A change of address is also in [Table 3.3](https://translate.googleusercontent.com/translate_f#_Ref449216106) describes and fixes this bug. This communication partner with a renewed call for display of all available Powerline respond per [Figure 3.7](https://translate.googleusercontent.com/translate_f#_Ref449559028) .

**Figure 3.7: Display of another power line communication system**

* 1. **Directories used commands**

As noted in Section [3.2.2](https://translate.googleusercontent.com/translate_f#_Ref449555254) above, there are various, defined by I2SE shell commands. These are especially needed for configuration and display of the powerline connection. By communicating with I2SE while editing the project overview was in [Table 3.3](https://translate.googleusercontent.com/translate_f#_Ref449216106) are created. It will always be the three main commands "PLCSTAT", "PLCTool" and "modpib" used.The command "PLCSTAT" is responsible for network settings between the two communication partners of the powerline connection. These include settings and display of parameters. "PLCTool" is responsible for the internal settings of the powerline connection. By "modpib", defined parameters are read and changed a pib file.

|  |  |
| --- | --- |
| I2SE commands |  |
| P lcstat | PLC Network Settings |
| P lctool | Local Settings of the PLC communication |
| PLCSTAT - I qca0 -t | He know of available Powerline connections |
| p lctool -i qca0 -p / tmp / local pib | Changing the MAC address |
| modpib -M < MAC> -v / tmp / pib |
| plcwait -R -I qca0 |
| / usr / local / bin / plcboot -N /root/Mac-QCA7000-v1.1.0-01-X-FINAL.nvm -P / tmp / pib -S |
| /root/NvmSoftloader-QCA7000-v1.1.0-01-FINAL.nvm -I qca0 -F |
| PLCTool -I qca0 -p / tmp / local pib | QCA Display Settings |
| m odpib -v / tmp / pib |

**Table 3.3: I2SE commands in Terminal**

To navigate within the Linux terminal commands are in [Table 3.4](https://translate.googleusercontent.com/translate_f#_Ref449471226) summarized the Shell commands most commonly used during processing. The detailed description of a command can also be issued by "man <command>" from the terminal. More support for programming with Nutshells can be found in the bibliography [(Hekman, 1998)](https://translate.googleusercontent.com/translate_f#Hekman_Linux_in_a_Nutshell) .

|  |  |
| --- | --- |
| General commands Nutshell |  |
| ssh [root@192.168.37.250](mailto:root@192.168.37.250) | Building a Secure Shell connection (Chapter [3.3.2](https://translate.googleusercontent.com/translate_f#_Ref449551436) ) |
| s ftp [root@192.168.37.250](mailto:root@192.168.37.250) | Connect an SFTP protocol (sh. Chapter [3.3](https://translate.googleusercontent.com/translate_f#_Ref448952802) ) |
| I fconfig | Configuration and status of all features ügbaren network interfaces |
| cd <folder name> | Change Directory: Change in a subdirectory of the current folder |
| cd .. | In Change Parent directory |
| cd / home / user | Switch to File deposit / home / user |
| vi <filename> | Open a file in the vi editor. To get out of the editor back into the terminal press "ESC", "q" and confirm with Return. |
| gcc -o <NAME> \* .c | Compile all source files as described in section [3.3.2](https://translate.googleusercontent.com/translate_f#_Ref449551436). |
| ./ <NAME> | Start of a program as described in section [3.3.2](https://translate.googleusercontent.com/translate_f#_Ref449551436) . |
| ping <IP address> | Sends data packets to an IP address to the presence of a compound to pr üfen |
| ping -I qca0 <IP address> | Ping command via network interface qca0 |
| to you | Shows all the current directory of the folders and files. |
| p wd | Is the current folder path |
| rm <file> | L lear a file |

**Table 3.4: Linux terminal commands**[**(shell commands, 2016)**](https://translate.googleusercontent.com/translate_f#Shellbefehle)

1. **software**

As noted in Section [1.2](https://translate.googleusercontent.com/translate_f#_Ref449882142) above, an object of the project is technically Software to split an existing communication stack to the ISO / IEC 15118 in vehicles and charging stations side into two parts and to implement the EVAchargeSE boards for demonstration run. For this reason, the establishment of a register for generating the PWM signal, and the IP-based communication structure must be considered additionally.

[Figure **4.1**](https://translate.googleusercontent.com/translate_f#_Ref449882797) shows the procedure for selected White during the project work.

While communication between the i.MX 28 and KL02 must first be prepared for the use of the functions of serial\_Programming directory. The contents of the messages which are communicated in this connection include, in Chapter [3.2.2](https://translate.googleusercontent.com/translate_f#_Ref449555254) defined.

Subsequently, the modulation of the IP-based messages on the PWM signal is realized. For the implementation of such a method of communication are sockets as described in section [4.1](https://translate.googleusercontent.com/translate_f#_Ref449974872) is used.

The final work package is divided into vehicle and charging station, as already mentioned, the existing stack. By connecting all the software packages of the communication stack according to ISO / IEC to be realized 15118th In addition to the split stack belongs to generate or read out of the PWM signal and the communication via TCP / IP.

* 1. **Figure 4 . 1 : the lapse of Software Programming****Communication within the Boards**

For communication between the i.MX 28 and the KL02 a UART interface is used. Since the i.MX 28 controls the program flow in the further course, it is defined as master. For this reason, the KL02 is already taught as slave programming. This means that it only to requests in accordance with Chapter [3.2.2](https://translate.googleusercontent.com/translate_f#_Ref449555254) responds.

To establish the connection to the settings according to the "Board Support Package" document [(I2SE, 2016)](https://translate.googleusercontent.com/translate_f#I2SE) used. Therein a baud rate of 57600Bd at 8 data and 1 stop bit are defined. Further, the modem device via which the data are to be transferred must be known. When you open the ports, the connection is secured in a file handle, so even with several existing compounds a differentiation can take place. For the initialization of such a communication protocol already exist Sources, which are used to change the configurations[(Sweet, 2016)](https://translate.googleusercontent.com/translate_f" \l "ser_prog" \o "ser_prog) .

After initialization of the UART interface messages can be transmitted. To send a message, it is written in an array. Sending <write (filehandle, array, number bytes to be transferred)> the array is sent via UART. The return value of the function indicates how many bytes were sent. By <read (filehandle, array, number to be received bytes)> the data to be received is written to the previously defined array. In both cases, must be known, expected or sent as many bytes. The in [3.2.2](https://translate.googleusercontent.com/translate_f#_Ref449555254) Frames defined thus always written in an array, shipped and received again.

* 1. **IP-based communication structure between EV and EVSE**

To establish a TCP communication widespread LAN interface for network applications, a socket is used. Also in [3.3.1](https://translate.googleusercontent.com/translate_f#_Ref449904153) and [3.3.2](https://translate.googleusercontent.com/translate_f#_Ref449551436) described features SFTP and SSH are applications that use a socket interface. With this protocol, the client-server principle applies, the charging station is assigned to the server side [(ITWissen, 2016)](https://translate.googleusercontent.com/translate_f#sockets) .

In order to devices that are connected to a network reach, and be able to distinguish, an IP address is assigned to them. Because some protocols can be used by multiple users at the same time simultaneously, port numbers to distinguish the processes used in addition. To establish a connection between the two boards EVAchargeSE the port number used both sides must be known. In addition, the client side needs to know the IP address of the server to be accessed.

The socket functions are accordingly for the construction, communication and the termination of the TCP connection [Figure 4.2](https://translate.googleusercontent.com/translate_f#_Ref449905272) uses.

The socket () is responsible for the establishment of a communication end point on. This includes parameters defining the address family, the type of transmission and the protocol to use for transmission, specify.

After both sides will have built a socket communication endpoint of the server port number by using the command bind () assigned. This port number must be known to the client, in order to build a future connection.

On the server side, all settings required for communication are known. For this reason, this communication node is ready to listen () for a connection request on the port previously defined.

Once the client tries to establish a connection using the call connect (), the server side must accept this with accept () to enable communication.

To send or receive now the functions send () and recv () are used. As with the UART connection size needs to be transmitted or the data to be received to be known Pact here in both functions. For this reason structs are used in a given case, their content and content sizes have been defined in advance.

Recently must terminate the communication to be terminated the socket connection to. This is done by calling Close (). This is the port used for communication is free and can again with the listen () - wait function to a connection request.

**Figure 4.2: Typical socket communication**[**(Large, 2015)**](https://translate.googleusercontent.com/translate_f#Groß_Sockets_Fuldau)

* 1. **Creating an ISO / IEC 15118 Stacks**

To have to build a stack to build a demonstrator not completely new, is built on a program for ISO / IEC 15118 from an open source project [(OpenV2G, 2016)](https://translate.googleusercontent.com/translate_f#OpenV2G) . This especially the message contents and their data types are described. Further exemplary processes for loading are presented with direct and alternating current.

In previous versions of the communication stack both vehicle and charging stations page are simulated by a program. This brings a large number of files with them, which additionally contain up to 20,000 lines of code. In addition, the final status of the project is to include two C programs, which enables integration into a demonstrator with two communication subscribers. This makes it imperative for each vehicle and charging stations side to implement a program and thus divide the existing stack.

To validate the two programs they need to communicate. For this reason, in chapter later in [4.2](https://translate.googleusercontent.com/translate_f#_Ref449979968) created library for IP-based communications involved. For the socket connection, the IP address 127.0.0.1 is used during initialization of the client. This leads to a local host which send messages to your own computer by this address is allowed. This is a program start Example White Eclipse allows on a computer, without having to make changes to the IP address. A distinction is made using the macro "CODE\_EXAMPLE". If this parameter specifies the value of the defined macro "CODE\_EXAMPLE\_SOFTWARE", the IP address 127.0.0.1 is used. For establishing communication with the server must "CODE\_EXAMPLE\_HARDWARE" be specified. In this project, this is the IP address of the charging station. When communicating the EVAchargeSE boards a powerline communication is being established. Therefore not the previous IP address of the Ethernet connection is used for this connection. The board, which has been addressed for connection to a computer with the IP address 192.168.37.250, uses the IP address 192.168.38.32 for powerline communications. The other board will tend to powerline connecting the IP address 192.168.38.33. This means that when flashing the boards must be paid to the IP addresses. If a server program flashed on the board with the IP address 192.168.37.250, has this to powerline communications address 192.168.38.32. It follows that the Board 192.168.37.251 is the client side using the IP address 192.168.37.33. When initializing the socket connection now has on the client side, the IP address of the server, so there indicated 192.168.37.32. For a better overview in this regard is [Figure 4.3](https://translate.googleusercontent.com/translate_f#_Ref450470225) .

After the communication stack in accordance with ISO / IEC 15118 is running in the development environment, this also needs to be implemented as hardware in the last step. In order to realize the necessary functions from the serial\_Programming library must be involved. It is important to be careful to distinguish between the vehicle and charging station.These functions are protected by the macro "CODE\_EXAMPLE" activated or deactivated. Such disconnection is the possibility to start the source files in a development platform and to functions that require a return value of other communication partners to bridge.

The resulting program flowchart of the electric vehicle is annexed by [Figure 7.9](https://translate.googleusercontent.com/translate_f#_Ref450261841) and [Figure 7.10](https://translate.googleusercontent.com/translate_f#_Ref450261854) shown.

**Figure 4.3: IP addresses of EVAchargeSE boards**

* + 1. **Extension to a state machine**

As noted in Section [2.5](https://translate.googleusercontent.com/translate_f#_Ref450076185) describes a state machine between its states changes due to entering events. Des Weitern are previous and future states involved in the implementation with.Moreover, such a machine, as [Figure 2.6](https://translate.googleusercontent.com/translate_f#_Ref449990101) shows, are clearly presented. This further brings with it the advantage to identify missing elements of the state machine.

Particularly in the present project, the implementation of such a state machine on the side of the charging station is advantageous. The communication requests are sequentially started by the electric vehicle, as indicated by [Figure 7.9](https://translate.googleusercontent.com/translate_f#_Ref450261841) and [Figure 7.10](https://translate.googleusercontent.com/translate_f#_Ref450261854) is seen. Since the charging station in a large part of the available states have several options of a subsequent state, provides a state machine for reasons of clarity, a good overview of the different branches. Furthermore, the individual states, which can take in this project such a machine, already in the news content in [Figure 7.1](https://translate.googleusercontent.com/translate_f#_Ref447825795) to [Figure 7.4](https://translate.googleusercontent.com/translate_f#_Ref447825798) can be seen.

A state transition is thereby triggered by a signal received from the electric vehicle message. If this message is composed of information that is not defined in any of the state transitions to the next state in accordance with ISO / IEC 15118, an error condition is assumed and the communication is terminated. Last to be considered that a distinction between AC and DC load is necessary from the time of the first level change in charging status C.

When implementing a function first "handshake" is called. This figure already contains the message "supported app Protocol". After the function has been successfully executed, the variable "next\_state" the first state "Session Setup" preset.

An overview of the state machine is in [Figure 4.4](https://translate.googleusercontent.com/translate_f#_Ref450261440) , [Figure 4.5](https://translate.googleusercontent.com/translate_f#_Ref450261460) and [Figure 4.6](https://translate.googleusercontent.com/translate_f#_Ref450261467) to see. To start the states in [Figure 4.4](https://translate.googleusercontent.com/translate_f#_Ref450261440) and [Figure 4.5](https://translate.googleusercontent.com/translate_f#_Ref450261460) , the level of the CP signal has the charging status B match. For a start the states in [Figure 4.5](https://translate.googleusercontent.com/translate_f#_Ref450261460) , however charging status C.

The program flow chart of the charging station and the functions in which the state machine is realized, can be found in the appendix starting with [Figure 7.5](https://translate.googleusercontent.com/translate_f#_Ref450262639) .

**Figure 4.4: State Machine EVSE Part 1 of 3**

**Figure 4.5: State Machine EVSE Part 2 of 3**

**Figure 4.6: State Machine EVSE Part 3 of 3**

1. **Results**

The boards of I2SE were accordingly [Figure 5.1](https://translate.googleusercontent.com/translate_f#_Ref450263002) up and pins, which correspond to those of the charging plug, connected to twisted cables.

**Figure 5.1: Hardware configuration**

Furthermore, a setting up of serial\_Programming library could be achieved. Once a board with a Linux computer is connected, it can be started using a Linux terminal. After starting a menu corresponding to [Figure 5.2](https://translate.googleusercontent.com/translate_f#_Ref450264045) displays the available functions can be started by that.

**Figure 5.2: Display of Linux terminals to start serial\_Programming**

The functions of the ISO / IEC 15118 required from the resulting library are located in the same named subdirectory of the final program. This results in signals on the line of control pins according to the following pictures.

**Figure 5.3: Level of Control pin for charge status A**

**Figure 5.4: Level of Control pin for charge status B**

**Figure 5.5: Level of Control pin for charge status C**

It is shown [in Figure 5.3](https://translate.googleusercontent.com/translate_f#_Ref450289056) the charging status A, which has a level of ± 12 ± 1 volt. According to IEC 61851, this corresponds to an unaffiliated electric vehicle. In the given case, however, the resistors can be set accordingly, so that the levels shown can be achieved while maintaining a connection between the CP, PE and PP consists of the boards.

In [Figure 5.4](https://translate.googleusercontent.com/translate_f#_Ref450289061) , the level is shown at bandaged, but not Ladebereitem electric vehicle. As already stated in chapter [2.2](https://translate.googleusercontent.com/translate_f#_Ref448342277) above, the negative level -12 ± 1V must be maintained while B is a range of 8 ± 1V is predetermined in the positive range in the present state of charge.

[Figure 5.5](https://translate.googleusercontent.com/translate_f#_Ref450289064) shows a limit of the tolerance range during the charging status C, whereby a load-ready compound is indicative. The tolerance zone for the level with min -12 ± 1V and max. 6 ± 1V indicates the positive peak is however 7,08V. Since the measured value for any further calculations benötig is considered merely one decimal place for determining the level.

1. **Summary****& Outlook**

The resulting outcome of the present project is a demonstrator of the ISO / IEC 15118 in hardware and software. For a corresponding hardware has been defined and put into operation and split a communication stack on vehicles and charging stations page. Furthermore, an important part of the construction of the PWM signal and the implementation of IP-based communications between EV and EVSE.

In the course of building a "wallbox" is realized within the scope of a study work. An electric car charger of this type is widespread in private use, as it offers no payment option[(Wiki\_Wallbox, 2016)](https://translate.googleusercontent.com/translate_f#Wiki_wallbox) . Furthermore, an app with the protocol to be connected, so that a start of the communication protocol and the definition of the parameters contained in it can be transmitted via smartphone.

1. **attachment**
   1. **AC charging procedure**

**Figure 7.1: News Summary for AC charging cycle (1 of 2)**

**Figure 7.2: News Summary for AC charging cycle (2 of 2)**

* 1. **DC charging procedure**

**Figure 7.3: News Summary for DC charging cycle (1 of 2)**

**Figure 7.4: News Summary for DC charging cycle (2 of 2)**

* 1. **Flowcharts the charging station**

**Figure 7.5: Program flow chart of charging stations Page**

**Figure 7.8: Program flow chart of the function call State B2 Communications**

* 1. **Flowcharts of the electric vehicle**

1. **bibliography**

|  |  |
| --- | --- |
| (Barth, 2015) | **Alexander Barth, Michael R öhrich**  Studienarbeit ISO 15118  (E-mail dated 13.05.2015) |
| (Bustamante, 2014) | **Michele Leroux Bustamante**  Finite State Machines, Wizards, and the Web  [https://msdn.microsoft.com/en-us/library/aa478972.aspx](https://translate.google.com/translate?hl=en&prev=_t&sl=auto&tl=en&u=https://msdn.microsoft.com/en-us/library/aa478972.aspx)  (Published February 2004) |
| (Debian, 2016) | **What is Debian GNU?**  Debian A nwenderhandbuch  [http://debiananwenderhandbuch.de/wasistdebiangnu.html](https://translate.google.com/translate?hl=en&prev=_t&sl=auto&tl=en&u=http://debiananwenderhandbuch.de/wasistdebiangnu.html)  (Accessed on 22.04.2016) |
| (El-Ko, 2016) | **Electronics Compendium**  Powerline Communication  (Accessed on 13.04.2016) |
| (FileZilla, 2016) | **FileZilla**  [https://filezilla-project.org/](https://translate.google.com/translate?hl=en&prev=_t&sl=auto&tl=en&u=https://filezilla-project.org/)  (Accessed on 25.04.2016) |
| (Gro ß, 2015) | **Prof. Dr. Siegmar Gro ß**  Script parallel processing  Cape. 4 The socket interface  Fulda University, Department of Applied computer science  (Accessed on 21.06.2015) |
| (Hekman, 1998) | **Jessica Perry Hekman**  Linux in a Nutshell  O'Reilly  ISBN 3-930673-57-6  1. reprint January 1998 |
| (I2SE, 2016) | **I2SE GmbH**  EVAchargeSE SE board  [https://www.i2se.com/product/evacharge-se/](https://translate.google.com/translate?hl=en&prev=_t&sl=auto&tl=en&u=https://www.i2se.com/product/evacharge-se/)  (Accessed on 17.04.2016) |
| (ITWissen, 2016) | **ITWissen**  socket  [http://www.itwissen.info/definition/lexikon/Socket-socket.html](https://translate.google.com/translate?hl=en&prev=_t&sl=auto&tl=en&u=http://www.itwissen.info/definition/lexikon/Socket-socket.html)  ( Accessed on 01.03.2016) |
| (Lewandowski, 2014) | **Christian Lewandowski.**  . Design and performance evaluation of a communications architecture for the provision of balancing power by electric vehicles  Technical university Dortmund  (Published on 10/17/2014) |
| (M ültin, 2014) | **Marc M ültin**  Dissertation: The electric vehicle as flexible consumers and energy storage in the smart home  KIT Faculty of Economics at the Karlsruhe Institute of Technology (KIT)  (Accessed on 08.04.2015) |
| (Opatz, 2006) | **Felix Opatz**  Network programming with BSD sockets  revision 1:15  http://www.zotteljedi.de/socket-buch/socket-buch.pdf  (Published on 08/07/2006) |
| (OpenV2G, 2016) | **Www.** **Sourceforge.net**  Open source project to implement a complete ISO 15118 communication protocol  http://openv2g.sourceforge.net/index.php/home.html  (Accessed on 07.04.2016) |
| (Shell commands, 2016) | **Shell commands**  Overview of the most common shell commands  http://www.shellbefehle.de/befehle/  (Accessed on 27.04.2016) |
| (Shrestha, 2015) | **Nerad Shrestha**  10 sFTP Command Examples to transfer files on the remote server in Linux  http://www.tecmint.com/sftp-command-examples/  (Updated 01/03/2015) |
| (Sweet, 2016) | **Michael R. Sweet**  Serial Programming Guide  5th Edition, Revision 3re  [https://support.dce.felk.cvut.cz/pos/cv5/doc/serial.html#config](https://translate.google.com/translate?hl=en&prev=_t&sl=auto&tl=en&u=https://support.dce.felk.cvut.cz/pos/cv5/doc/serial.html%23config#config)  (Accessed on 05.01.2016) |
| (Ubuntu\_SSH, 2016) | **Wiki.Ubuntuusers**  SSH  [https://wiki.ubuntuusers.de/SSH/](https://translate.google.com/translate?hl=en&prev=_t&sl=auto&tl=en&u=https://wiki.ubuntuusers.de/SSH/)  (Accessed on 22.04.2016)) |
| (Wiki\_SSH, 2016) | **Wikipedia**  Secure Shell  https://de.wikipedia.org/wiki/Secure\_Shell  (Access on 25.04.2016) |
| (Wiki\_Stecker, 2016) | **Wikipedia**  Connector types and charging modes for electric vehicles  [https://de.wikipedia.org/wiki/IEC\_62196](https://translate.google.com/translate?hl=en&prev=_t&sl=auto&tl=en&u=https://de.wikipedia.org/wiki/IEC_62196)  https://de.wikipedia.org/wiki/IEC\_62196\_Typ\_2  (Accessed on 08.03.2016) |
| (Wiki\_Wallbox, 2016) | Wikipedia  Wall charger  [https://de.wikipedia.org/wiki/Wandladestation](https://translate.google.com/translate?hl=en&prev=_t&sl=auto&tl=en&u=https://de.wikipedia.org/wiki/Wandladestation)  (Access on 06.05.2016) |

|  |  |
| --- | --- |
| Finkbeiner, Melanie |  |
|  |  |

|  |  |
| --- | --- |
| Finkbeiner, Melanie |  |