



Charging Station for ISO / IEC 15118 Protocol

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

*IEC 61851*

Bachelor's Project

**Presented by**

Jiztom Kavalakkatt Francis

Nivas Gokul Manimurugesan

Raphael Scholz

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

Hochschule Heilbronn / Heilbronn University

Faculty of Mechanics und Electronics

Automotive Systems Engineering

**Table of Contents**

**Contents III**

**Abstract IV**

**List of tables V**

**List of Figures VI**

**List of Symbols and Abbreviations VII**

1. **Introduction**
   1. Short Version
   2. Task
   3. Chapter Overview
2. **Literature Survey**
   1. Scholastic survey
   2. Reference
3. **Problem Definition / Objective**
4. **Methodology**
   1. IEC 62196: Charging Plug
   2. IEC 61851
   3. ISO/IEC 15118
      1. Open v2g project
   4. TCP/IP communication
   5. UML
      1. UML State Machine Diagram
      2. UML Sequence Diagram
   6. Automatic State Machine
5. **Software Description**
   1. Filezilla
   2. Geany
   3. Putty Terminal
   4. EB GUIDE 6.3
   5. www.draw.io
   6. Visual Paradigm 14.0
   7. gcc compiler on Linux terminal
6. **Hardware Description**
   1. Raspberry Pi 3 with 7” Capacitive Touch screen
   2. EVAChargSE Board
   3. Communication between EVSE and EV
   4. Communication between EVSE and Raspberry Pi
   5. Integrating the system
   6. Testing the system using dummy parameters.
7. **Results**
8. **Conclusion**
9. **Summary & Outlook**
10. **Attachments**

**Bibliography**

**Abstract**

**List of Tables**

**List of Figures**

**List of Symbols and Abbreviations**

1. Introduction

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

* 1. Short Version

The present project work serves to set up a working model for the current standardization of the vehicle-column communication according to ISO / IEC 15118 along with the HMI display.

The built-up working model consists of two interconnected boards and a Raspberry Pi for teh display. 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. **Task**

The main task is to implement a basic working model of the car charging station with the HMI interfaced into the system. It involves the study of the basic working of the ISO 15118 protocol and defining the possible working structure of the Charging station.

The task involves the design of the communication mechanisms and processes between the main processor of an electric vehicle supply equipment (EVSE) – or called charging station - and a computer that runs the HMI on that EVSE (HMI), and the electric vehicle to be charged (EV)

The targets achieved:

- Analyzed the requirements of ISO 15118 and IEC 61851 based on the work of the references.

- Designed, discussed and finalized the state machine and the communication process between EVSE and HMI together with a German student (Raphael Scholz)

- Learned about UML as a description language for state machines and communication sequences

- Experienced TCP/IP communication with practical training on a Linux system

- Started coding for the implementation of the above

* 1. Chapter Overview

1. Literature Survey
   1. Scholastic Survey

The following were considered for the initial study of the Car Charging Process

* + 1. **Automotive Ethernet: in-vehicle networking and smart mobility**

**Authors:**

* Peter Hank NXP Semiconductors, Hamburg, Germany
* Steffen Müller NXP Semiconductors, Hamburg, Germany
* Ovidiu Vermesan SINTEF, Oslo, Norway
* Jeroen Van Den Keybus Triphase NV, Leuven, Belgium

**Proceedings**

DATE '13 Proceedings of the Conference on Design, Automation and Test in Europe

Pages 1735-1739

Grenoble, France — March 18 - 22, 2011

**Abstract:**

This paper discusses novel communication network topologies and components and describes an evolutionary path of bringing Ethernet into automotive applications with a focus on electric mobility. For next generation in-vehicle networking, the automotive industry identified Ethernet as a promising candidate besides CAN and FlexRay. Ethernet is an IEEE standard and is broadly used in consumer and industry domains. It will bring a number of changes for the design and management of in-vehicle networks and provides significant re-use of components, software, and tools. Ethernet is intended to connect inside the vehicle high-speed communication requiring sub-systems like Advanced Driver Assistant Systems (ADAS), navigation and positioning, multimedia, and connectivity systems. For hybrid (HEVs) or electric vehicles (EVs), Ethernet will be a powerful part of the communication architecture layer that enables the link between the vehicle electronics and the Internet where the vehicle is a part of a typical Internet of Things (IoT) application. Using Ethernet for vehicle connectivity will effectively manage the huge amount of data to be transferred between the outside world and the vehicle through vehicle-to-x (V2V and V2I or V2I+I) communication systems and cloud-based services for advanced energy management solutions. Ethernet is an enabling technology for introducing advanced features into the automotive domain and needs further optimizations in terms of scalability, cost, power, and electrical robustness in order to be adopted and widely used by the industry.

* + 1. **Towards standardized Vehicle Grid Integration: Current status, challenges, and next steps**

**Authors:**

* BO chen Argonne National Laboratory, USA
* Keith S. Hardy Argonne National Laboratory, USA
* Jason D. Harper Argonne National Laboratory, USA
* Daniel S. Dobrzynski Argonne National Laboratory, USA

**Published in:** Transportation Electrification Conference and Expo (ITEC), 2015 IEEE

**Abstract:**

This paper studies what are needed to enable the standardization of Vehicle Grid Integration (VGI). The requirements of interoperable VGI are examined at multiple interoperability layers defined by reference architecture models, including European Commission's Mandate 490 (EU-M490), National Institute of Standards and Technology (NIST) Smart Grid Architectural Methodology (SGAM), and the Institute of Electrical and Electronic Engineers (IEEE) 2030 Smart Grid Interoperability Reference Model (SGIRM). The current status of standards and technology development is reviewed and VGI demonstrations are discussed. The paper identifies barriers for the implementation of an interoperable VGI and provides recommendations to address these challenges.

* + 1. **ISO 15118 – charging communication between plug-in electric vehicles and charging infrastructure**

**Authors:**

* Dr. Andreas Heinrich Daimler AG, Holzgerlinen, Germany
* Michael Schwaiger BMW Group, Munich, Germany

**Book Title:** Grid Integration of Electric Mobility

**Book Subtitle:** 1st international ATZ Conference 2016

**Pages:** pp 213-227

**Abstract:**

* + 1. **Assuring Interoperability between Conductive EV and EVSE Charging SystemsAuthors:**
* M. Sc. Michael Tybel Scienlab electronic systems, Bochum
* Dr.-Ing Andrey Popov Scienlab electronic systems, Bochum
* Dr.-Ing Michael Schugt Scienlab electronic systems, Bochum

**Link to document:**

<http://www.p0p0v.com/science/downloads/TybelPopovSchugt15.pdf>

**Abstract:**

The development and deployment rate of electric vehicles (EV) and plug-in electric vehicles (PEV) substantially depends on the corresponding EV supply equipment (EVSE). The facts that vehicles are intrinsically mobile and hence require interoperability between manufacturers, countries and charging points, implies that the components of the charging systems should extensively be tested, in order to allow access of the companies to the global market.

The two most common automotive charging communication standards are the pulse width modulation (PWM) based IEC 61851-1 [1] and the CAN based CHAdeMO [2]. The third and latest approach is a power line communication (PLC) using V2Gprotocol specified by the ISO 15118 [3]. The norm has been published in 2014 and will become the standard in Europe and North America for DC- and AC-Charging within the next years since all major OEMs have decided to apply it. In order to implement the new standard in a way, that supports all specified use cases (e.g. private or public charging, plug & charge or external identification/payment) and simultaneously assure operation between all EV/EVSE communication controllers of different origins, dedicated verification techniques, and routines are required.

This paper introduces the ISO 15118 norm and suggests independent and reproducible test methods that allow developers and quality managers to achieve a high degree of interoperability.

* + 1. **Vehicle-to-Grid AC Charging Station: AN approach for Smart Charging Development**

**Authors:**

* D. Wellisch Deggendorf Institute of Technology, Freyung
* J. Lenz Deggendorf Institute of Technology, Freyung
* A. Faschingbauer Deggendorf Institute of Technology, Freyung
* R. Pöschl Deggendorf Institute of Technology, Freyung
* S. Kunze Deggendorf Institute of Technology, Freyung

**Link to the document:** [researchgate.com](https://www.researchgate.net/profile/Rainer_Poeschl/publication/282846691_Vehicle-to-grid_AC_Charging_Station_An_Approach_for_Smart_Charging_Development/links/561e15f808aec7945a253e1c.pdf)

**Abstract:**

The use of electric powered vehicles is increasing steadily. This also leads to new challenges for the power grid. An electric powered vehicle provides heavy stress for the grid, especially when many vehicles are loading their accumulators simultaneously. To counteract these negative effects, smart charging is developed. With intelligent vehicle-to-grid communication, the stress for the grid, during the charging process, can be reduced. This is especially important when renewable energy sources are utilized. Using new software protocols and suiting hardware applications, smart charging can harmonize the needs of renewable energy sources and electromobility. In this paper, a smart charging capable AC charging station for hardware and software evaluation is proposed. This system is based on OCPP 2.0 and the ISO 15118 standard.

* 1. Reference

1. Problem statement /Objective
2. Methodology

This chapter provides an overview of work and information to which the project is worked up. These include, inter alia, the former way of loading a vehicle as well as the previous exchange of information, the different vehicle used the connector for loading of electronic vehicles. Further information, for this work, are a study work, which is to describe the 15118 ISO accurate and a dissertation of Dr. Marc Mültin which is engaged in the electric vehicle as a "flexible consumers and energy storage in the smart home".

* 1. IEC 62196: vehicle plug

Connector types and charging modes of electric vehicles are defined by the International Electrotechnical Commission in IEC 62196 (Wiki\_plug, 2016),

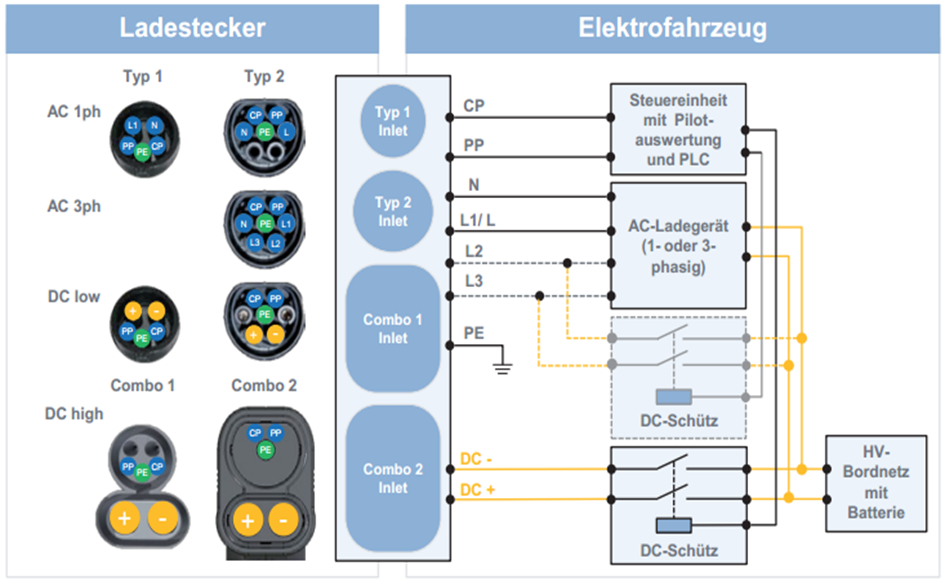
The second part of the standard was published in 2011 and includes different types of connectors. This includes three of the most popular at this time charging plug.

The Type 1 charging plug, which in Figure 2.1 is shown, takes its specification of the SAE J1772. This was first published in 1996 by the Society of Automotive Engineers and has since been expanded and maintained by this. The disadvantage of this connector type 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 found in figure 2.1. The plug finds its origins through a collaboration of the connector manufacturer Mennekes with the power company RWE and the carmaker Daimler. The naming of the Mennekes plug thus receives this 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.

For all defined types of connectors as defined in Type 1 Signal contacts CP (Control Pilot) and PP are (Proximity pilot) included which allow charging to IEC 61851



In the Figure 2.1 Signal contacts shown are defined as follows:

|  |  |  |
| --- | --- | --- |
| abbreviation | Contact | function |
| CP | Control pilot | Control signals charging station🡪electric vehicle |
| PP | Proximity pilot | Check the presence of a charging cable |
| N | Neutral | For AC charging |
| L1, L2, L3 | Current-carrying phases | For AC charging with a (L1 / L) or three (L1, L2, L3) Phases |
| PE | Protective Earth | protective conductor |
| DC +/- | Current-carrying phases | For DC charging |

* 1. IEC 61851

The IEC 62196 is an international standard for a number of types of plugs and charging modes for electric vehicles and of the International Electrotechnical Commission maintained (IEC). The standard is valid in Germany as a DIN standard DIN EN 62196. It consists of several parts which have been passed in succession. The third part was published in June 2014. In June 2015, the standardization process for part 4 (light- weight electrical connections) began.

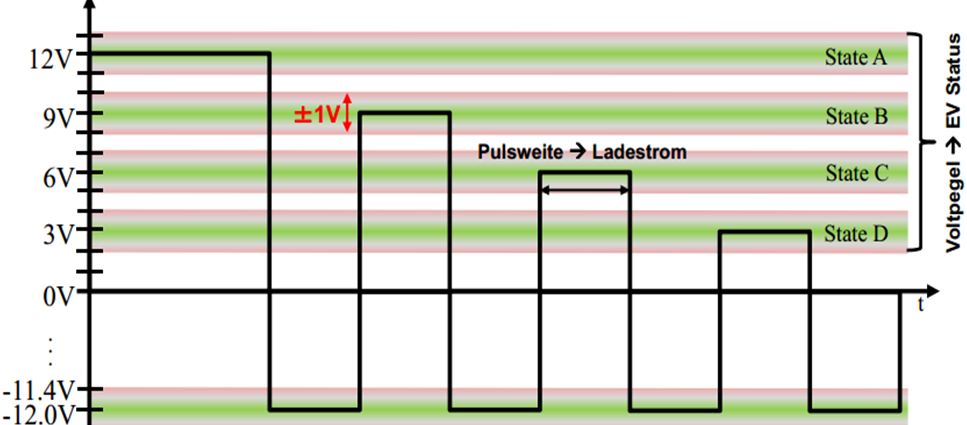
The standard adopts the IEC 61851 definition for a signal pin that switches the charging current - the charging station remains de-energized until an electric vehicle is connected. During the charging process, the vehicle cannot be put into operation.

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. The signals of the Control Pilot (CP), Protective Earth (PE) and Proximity Pin (PP) contacts described in section 2.1 are required to determine the parameters required for loading.

For charging the vehicle, both communication subscribers are first connected to one another. A 1 kHz 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).

On the vehicle side, resistors are connected between CP and PE or PP and PE. Different charging states are indicated by different switchable levels of the voltage between the CP and PP contacts, as shown in Figure 2.2. Please note that the negative voltage value is permanently -12V, and only the positive values ​​change. A definition of the individual states is Table 2.2.

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.



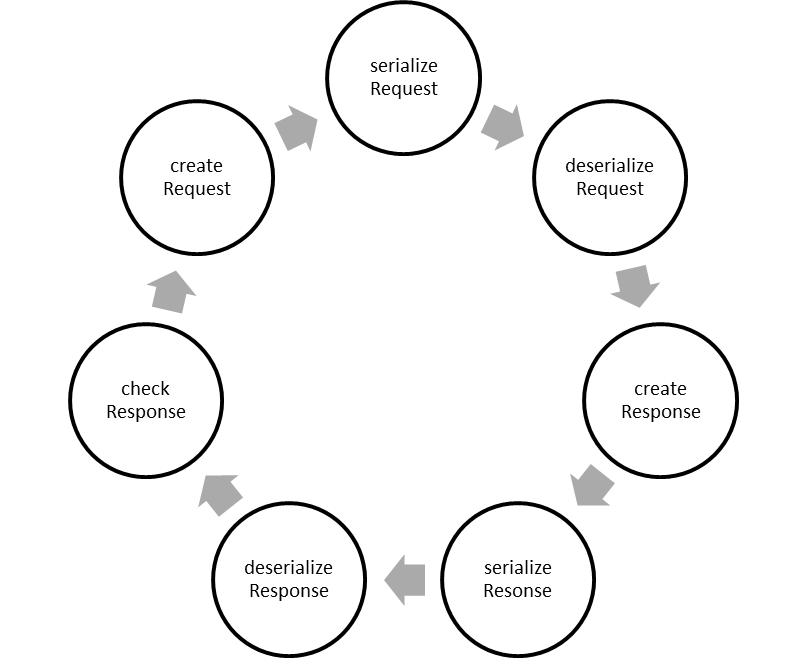
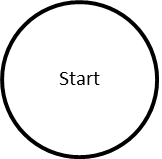
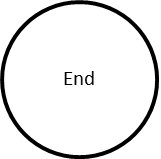
|  |  |  |
| --- | --- | --- |
| Level | State | Condition Description |
| 12 ± 1 V | State A | Electric vehicle is not connected |
| 9 ± 1 V | State B | connected electric vehicle, not charging Ready |
| 6 ± 1 V | State C | connected electric vehicle, ready to charge |
| 3 ± 1 V | State D | connected electric vehicle, ready for loading, ventilation needed |
| 0 ± 1 V | State E | Network problem, PP Short to earth |
| -12V | State F | Vehicle unavailable Error |

* 1. ISO 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), This should be a "plug-and-charge" mechanism for authentication, authorization, accounting, and for load control, so that needed to load enable parameters are stored in the vehicle and the user both communication parties must connect only. The individual communication Content will be the level of tension control pin signal from Chapter2.2 correspondingly Figure 2.3 assigned.

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 to Figure 7.4, 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)

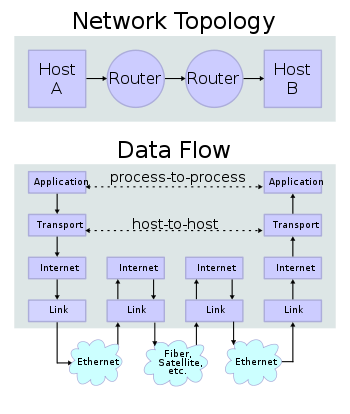
* + 1. Open V2G Project

An already far-reaching example for the implementation of a communication pack according to ISO / IEC 15118 has already been initiated by the support of Siemens Corporate Technology as Open Source project (OpenV2G, 2016). Both the loading column and the vehicle side are displayed in a program code and the messages are generated, 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.

* 1. TCP/IP Communication:

The Internet protocol suite is the conceptual model and set of communications protocols used on the Internet and similar computer networks. It is commonly known as TCP/IP because the original protocols in the suite are the Transmission Control Protocol (TCP) and the Internet Protocol (IP).

The Internet protocol suite provides end-to-end data communication specifying how data should be packetized, addressed, transmitted, routed and received. This functionality is organized into four abstraction layers which are used to sort all related protocols according to the scope of networking involved. From lowest to highest, the layers are the link layer, containing communication methods for data that remains within a single network segment (link); the internet layer, connecting independent networks, thus providing internetworking; the transport layer handling host-to-host communication; and the application layer, which provides process-to-process data exchange for applications.

Technical standards specifying the Internet protocol suite and many of its constituent protocols are maintained by the Internet Engineering Task Force (IETF). The Internet protocol suite model is a simpler model developed prior to the OSI model.

TCP/IP is a two-layer program. The higher layer, Transmission Control Protocol, manages the assembling of a message or file into smaller packets that are transmitted over the Internet and received by a TCP layer that reassembles the packets into the original message. The lower layer, Internet Protocol, handles the address part of each packet so that it gets to the right destination. Each gateway computer on the network checks this address to see where to forward the message. Even though some packets from the same message are routed differently than others, they'll be reassembled at the destination.



TCP/IP (Transmission Control Protocol/Internet Protocol) is the basic communication language or protocol of the Internet. It can also be used as a communications protocol in a private network (either an intranet or an extranet). TCP/IP uses the client/server model of communication in which a computer user (a client) requests and is provided a service (such as sending a Web page) by another computer (a server) in the network. TCP/IP communication is primarily point-to-point, meaning each communication is from one point (or host computer) in the network to another point or host computer. TCP/IP and the higher-level applications that use it are collectively said to be "stateless" because each client request is considered a new request unrelated to any previous one (unlike ordinary phone conversations that require a dedicated connection for the call duration). Being stateless frees network paths so that everyone can use them continuously. (Note that the TCP layer itself is not stateless as far as any one message is concerned. Its connection remains in place until all packets in a message have been received.)

* 1. UML
     1. Introduction

UML stands for Unified Modelling Language. UML is a way of visualizing a software program using a collection of diagrams. The notation has evolved from the work of Grady Booch, James Rumbaugh, Ivar Jacobson, and the Rational Software Corporation to be used for object-oriented design, but it has since been extended to cover a wider variety of software engineering projects. Today, UML is accepted by the Object Management Group (OMG) as the standard for modelling software development.

* + - 1. UML 2.0

UML 2.0 helped extend the original UML specification to cover a wider portion of software development efforts including agile practices.

Here are some of the changes made to UML diagrams in UML 2.0:

• Improved integration between structural models like class diagrams and behaviour models like activity diagrams.

• Added the ability to define a hierarchy and decompose a software system into components and sub-components.

• The original UML specified nine diagrams; UML 2.x brings that number up to 13. The four new diagrams are called: communication diagram, composite structure diagram, interaction overview diagram, and timing diagram. It also renamed state chart diagrams to state machine diagrams, also known as state diagrams.

* + - 1. Types of UML Diagrams

The current UML standards call for 13 different types of diagrams: class, activity, object, use case, sequence, package, state, component, communication, composite structure, interaction overview, timing, and deployment.

These diagrams are organized into two distinct groups: structural diagrams and behavioral or interaction diagrams.

1. Structural UML diagrams

• Class diagram

• Package diagram

• Object diagram

• Component diagram

• Composite structure diagram

• Deployment diagram

1. Behavioural UML diagrams

• Activity diagram

• Sequence diagram

• Use case diagram

• State diagram

• Communication diagram

• Interaction overview diagram

• Timing diagram

In our project we deal with behavioural UML diagram to explain how the Charging station components behave at different states and kinds. In Behavioural UML the state machine diagram and the sequence diagram is well used.

* + 1. State Machine Diagram

A state diagram shows the behaviour of classes in response to external stimuli. Specifically a state diagram describes the behaviour of a single object in response to a series of events in a system. Sometimes it's also known as a Harel state chart or a state machine diagram. This UML diagram models the dynamic flow of control from state to state of a particular object within a system.

* + - 1. Difference between a State Diagram and a Flowchart

A flowchart illustrates processes that are executed in the system that change the state of objects. A state diagram shows the actual changes in state, not the processes or commands that created those changes.

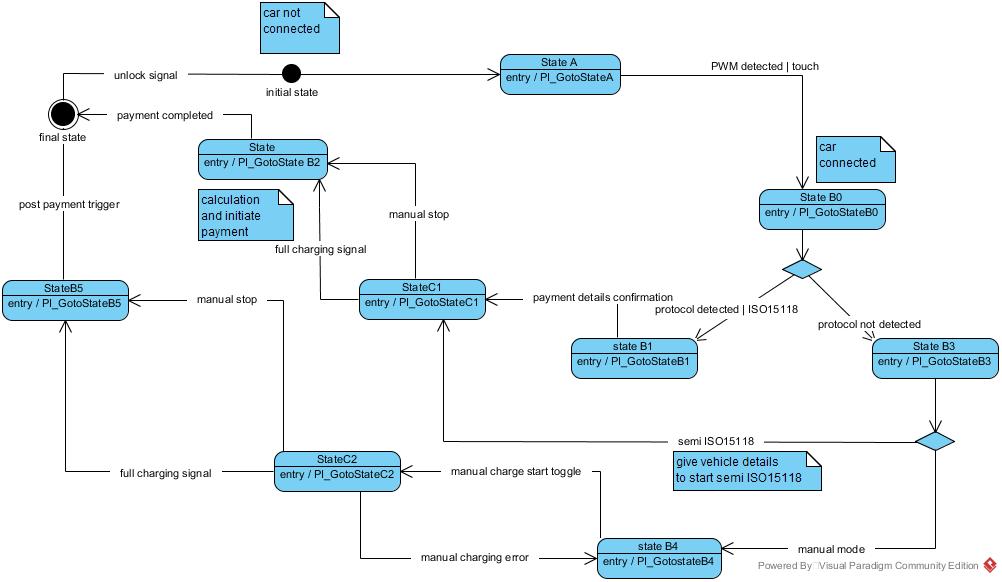
* + - 1. Steps to Drawing a State Diagram

Before you begin your drawing find the initial and final state of the object in question.

Next, think of the states the object might undergo. For example, in e-commerce a product will have a release or available date, a sold out state, a restocked state, placed in cart state, a saved on wish list state, a purchased state, and so on.

Certain transitions will not be applicable when an object is in a particular state, for example a product can be in a purchased state or a saved in cart state if its previous state is sold out.

* + - 1. The Charging Station – State Machine Diagram



* + 1. Sequence Diagram

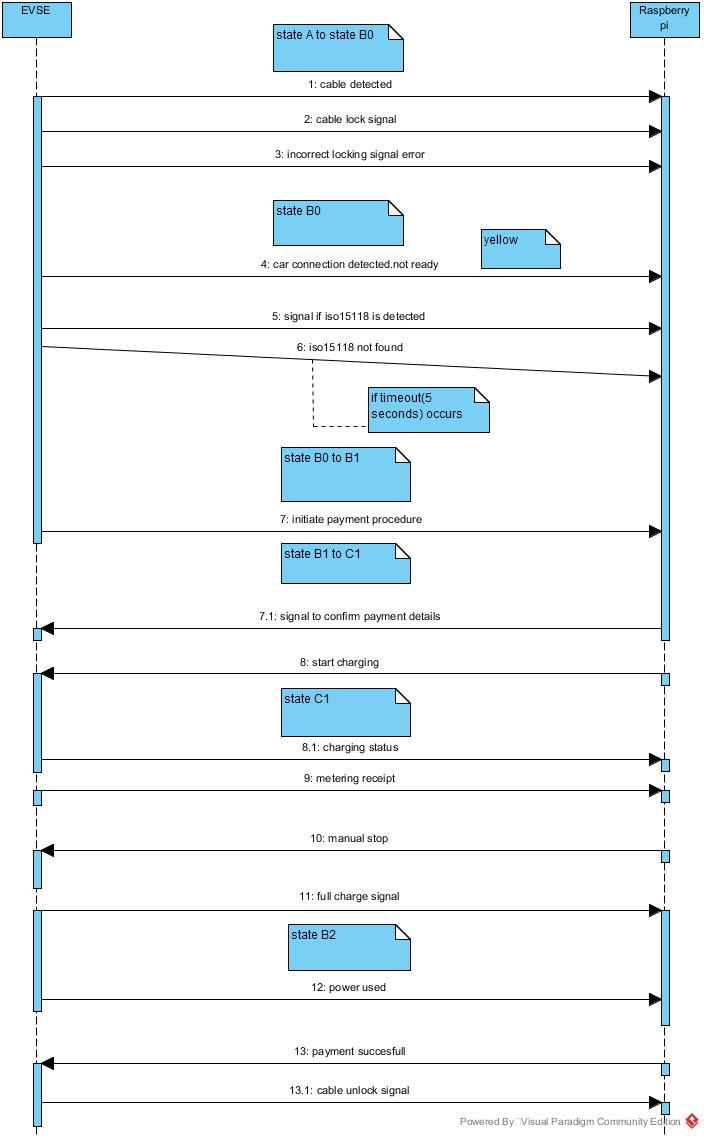
Sequence diagrams describe interactions among classes in terms of an exchange of messages over time. They're also called event diagrams. A sequence diagram is a good way to visualize and validate various runtime scenarios. These can help to predict how a system will behave and to discover responsibilities a class may need to have in the process of modelling a new system.

* + - 1. How to Use Sequence Diagrams

• Model and document how your system will behave in various scenarios

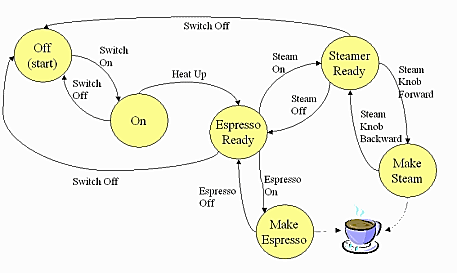
• Validate the logic of complex operations and functions

* + - 1. The Charging Station – The State Machine Diagram



* 1. Automatic State Diagram:
     1. Introduction

An automatic state machine consists of states, status transitions, and actions. The purpose of these tools is to implement the control of a system which 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.



An illustration of such an automatic state machine is provided by a coffee machine as shown in Figure 2.6. 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 cannot 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 switch case 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 for the coffee machine Figure 2.6 is shown in Figure 2.7.

switch (current\_state)

case OFF: if (switch\_OFF) current\_state = switch\_ON; break;

case ON: if(Heat\_up) current\_state = Espresso\_ready; break;

case Espresso\_ready: if(Steam\_on) current\_state = Steamer\_ready;

else if(Espresso\_on) current\_state = Make Espresso;

break;

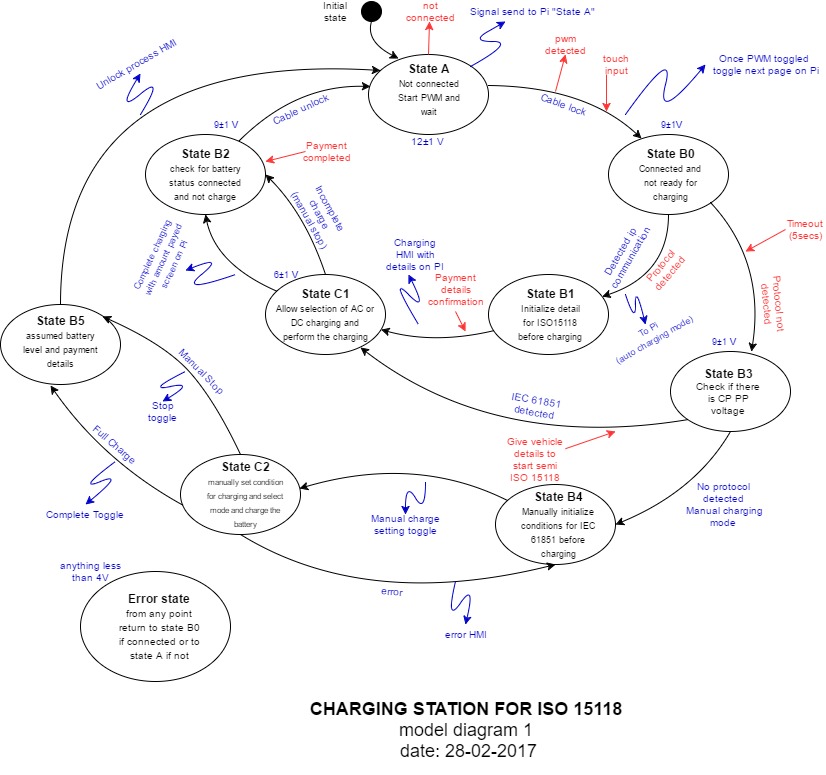
…

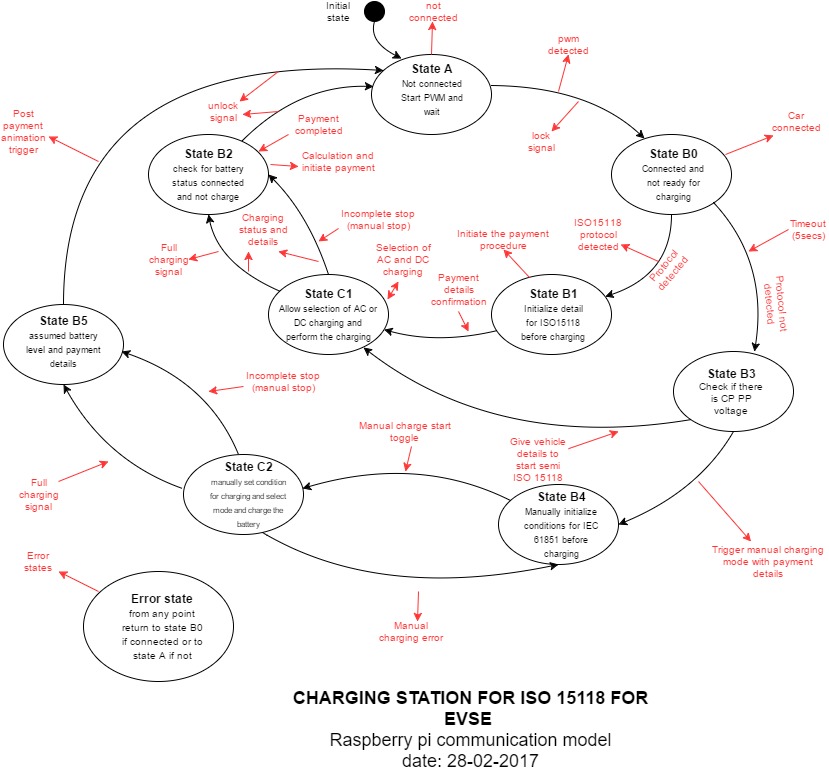
…

* + 1. State diagram of the charging station

The charging station requires the need of an automatic state diagram to explain its working. It helps the developers visualize the process and set up the required process faster. In the state diagram designed we have used the online drawing tool **www.draw.io**. It is an easy to use tool with several options.

The finalized state diagram is shown in figure … and also in figure … There is a separate state diagram for the EVSE process and the signal the Raspberry pi will work with.





1. Software Description
   1. FileZilla

FileZilla is a free software, cross-platform FTP application, consisting of FileZilla Client and FileZilla Server. Client binaries are available for Windows, Linux, and macOS, server binaries are available for Windows only. The client supports FTP, SFTP and FTPS (FTP over SSL/TLS).

FileZilla's source code is hosted on Source Forge and the project was featured as Project of the Month in November 2003.However, there have been criticisms that Source Forge bundles malicious software with the application; and that FileZilla stores users' FTP passwords insecurely.

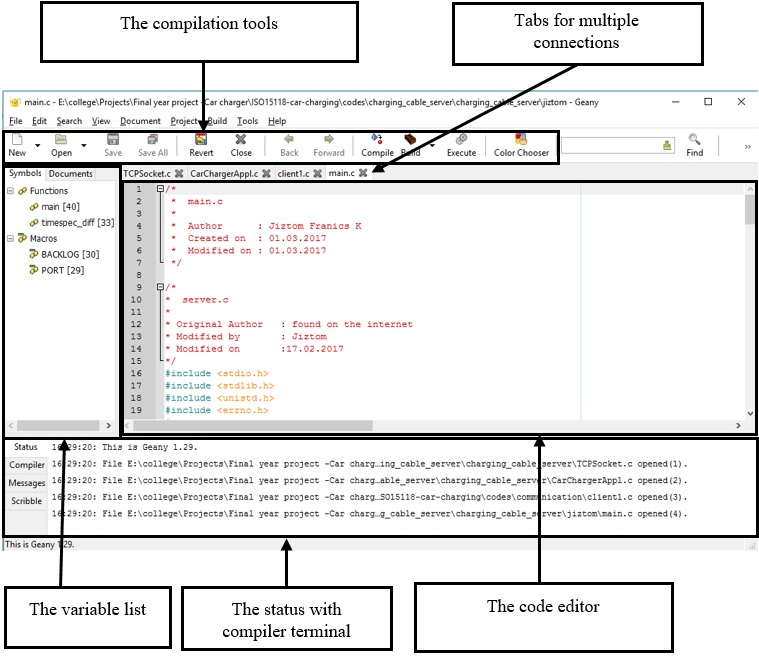
These are some features of FileZilla.

* Transfer files in FTP, SFTP, encrypted FTP such as FTPS and SFTP.
* Support IPv6 which is the latest version of internet protocol.
* Available in 47 languages worldwide.
* Supports resume which means the file transfer process can be paused and continued.
* Tabbed user interface for multitasking, to allow browsing more than one server or even transfer files simultaneously between multiple servers.
* Site Manager to manage server lists and transfer queue for ordering file transfer tasks.
* Bookmarks for easy access to most frequent use.
* Drag and drop to download and upload.
* Directory comparison for comparing local files and server files in the same directory. When the file doesn't have the same information (name not match, or size not match) it will highlight that file in colour.
* Configurable transfer speed limits to limit the speed transferring the files, which helps reducing error of transferring
* Filename filters, users can filter only specific files that have the conditions they want.
* Network configuration wizard, help configuring confusing network settings in form of step-by-step wizard
* Remote file editing, for quickly edit file on server side on-the-fly. No need to download, edit on the computer and re-upload back to the server.
* Keep-alive, if the connection has been idle for the long time it will check by sending keep-alive command.
* HTTP/1.1, SOCKS5 and FTP-Proxy support
* Logging to file
* Synchronised directory browsing
* Remote file search to search file on the server remotely.



* 1. Geany

Geany is a lightweight GUI text editor using Scintilla and GTK+, including basic IDE features. It is designed to have short load times, with limited dependency on separate packages or external libraries on Linux. It has been ported to a wide range of operating systems, such as BSD, Linux, mac OS X, Solaris and Windows. Because Windows lacks a virtual terminal equivalent, the Windows port lacks an embedded terminal window. Also missing from the Windows version are the external development tools present under UNIX, unless installed separately by the user. Among the supported programming languages and mark-up languages are C, C++, C#, Java, JavaScript, PHP, HTML, LaTeX, CSS, Python, Perl, Ruby, Pascal, Haskell, Erlang, Vala and many others.

In contrast to traditional Unix-based editors like Emacs or Vim, Geany more closely resembles a small and fast IDE.

.

* 1. Putty Terminal

1. Hardware Description
2. Results
3. Conclusion
4. Summary & outlook
5. Attachments

Bibliography