UNIVERSITETET I AGDER

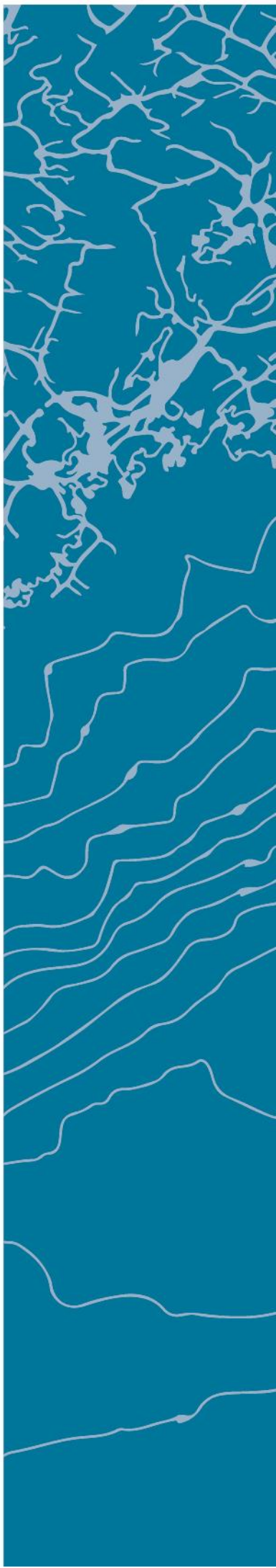
Smart Electric Vehicle Charging System

Controlling Multiple Electrical Vehicle Chargers using OCPP to Limit Electricity Demand.

GAUTE NESS

SUPERVISOR
Mohan Lal Kohle & Nils Ulltveit-Moe

University of Agder, 2017
Faculty of Engineering and Science
Department of Engineering and Science



Abstract

Peak demand is a problem when Electrical Vehicle charging is introduced in the electricity grid. Local limitations like fuses and transformer capacity can rapidly be overloaded if multiple Electrical Vehicles are charging at the same time. This can be solved by shifting these loads in time. This master's Thesis presents a solution by using the communication protocol OCPP to restrict one or more chargers below a set demand limit. The solution is developed in cooperation with Sharebox [1] and Grønn Kontakt [2], and is in prototype phase at the end of this Master's Thesis. Testing demonstrates that the prototype can reduce the electricity demand of Charge Points in response to other loads.

The solution is adaptable and can with small modifications be improved to limit charging based on overall grid capacity. This improvement will allow demand side management to counter grid overload instead of the common solution to have supply side resolve any grid demands.

Key Words:

Energy Management, OCPP, EVSE, Demand Limit

Preface

I would like to express gratitude to my supervisors, Nils Ulltveit-Moe and Mohan Lal Kolhe who have helped me through this Master's Thesis. This Master Thesis is part of a bigger project in cooperation with Sharebox and Grønn Kontakt, and I am thankful for the good cooperation with the employees of these two companies.

Both the Norwegian and French departments of Schneider have been weary helpful in realizing this project.

This Master Thesis is part of a project receiving funding from VRI (Virkemidler for regional FoU og innovasjon).

Table of content

List of Figures

Abbreviations

EVSE	Electrical Vehicle Charging Station. Charge Point will be used in this document based on OCPP documentation
OCPP	Open Charge Point Protocol
EV	Electric Vehicle
CT sensor	Current Transformer sensor
RMS	Root Mean Square
API	Application Programming Interface

1 Introduction

The number of Electrical Vehicles (EVs) are rapidly increasing in Norway. At the start of 2017 there were 97 500 EVs, a 40% increase in one year [3]. The electrical load caused by EVs require smart solutions to avoid grid peak overloads. On a large scale, smart charging solutions can be based on variable power costs depending on the time of the day. This will move large electricity loads outside of the peak periods of the day and will be beneficial to the nationwide grid by reducing electricity peaks.

The company Grønn Kontakt operates 488 [2] Charge Points in Norway where a user can pay to charge an EV. They want greater control of these Charge Points where there are challenges with limited electricity supply. To find a solution, a VRI (Virkemidler for regional FoU og innovasjon) [4] project was made in cooperation with UiA, and this Master Thesis is part of that VRI project.

The desired solution is a Local Controller able to limit EV electricity consumption below a demand limit while also supporting the commercial payment options needed for the Grønn Kontakt business model. There are existing solutions for load balancing Charge Points, but they are either proprietary and will not support other vendors, or they do not support the OCPP payment method. There are also working solutions using the older OCPP version 1.5, but this can only turn Charge Points on or off, while the OCPP version 1.6 gradually reduces load.

This Local Controller solution will help limit peak load on existing Charge points, and will allow multiple Charge Points to be installed that, on full load, will overload the local system. It is also possible to implement a premium solution where users can pay more to be prioritized, when the available electricity is shared between Charge Points.

A Local Controller will also open several new business options for Grønn Kontakt. Home Charge Points can be regulated based on other loads in homes, and can also be rented out to others at times when the owner is not using the Charge Point.

Another example where a Local Controller can be beneficial in the home market, is apartments with joined garage facilities. The electricity supply is often common to all the garages or parking lots and electricity is billed to the entire complex instead of individual users. With this new Grønn Kontakt solution, the users with EVs can pay for their electricity consumption even if they use electricity from the line common to the entire apartment complex.

1.1 Problem statement

Grønn Kontakt [2] has a practical problem with peak load on Charge Points. This problem is relevant for individual homes and for locations with several Charge Points that can overload the electricity grid. Local transformers are a typical example of a current limitation. Upgrading the electricity grid can be expensive, and a smart charging solution can prevent this. This will also open new business opportunities for the company.

1.2 Literature review

There is much literature regarding theoretical EV balancing against the grid and peak loads. This Master Thesis aims for a practical prototype implementation, and the algorithms used will therefore be less complex than some proposed in literature. A practical implementation of an earlier version of OCPP is presented in a Master Thesis from 2015, and some of the challenges are similar to the once faced in this project.

The framework for this Master Thesis was given by Grønn Kontakt requirements, and the work was based on datasheet and protocol specifications, rather than theoretical literature or control theory. The literature review will therefore be used to plan for further work and how the Local Controller and similar smart systems can affect the electricity grid on the large scale.

1.2.1 Agent-based Charging Scheduling of Electric Vehicles

This document is written by Armin Ghasem Azar and Rune Hylsberg Jacobsen at Aarhus University, Denmark. It is a method for controlling EV load by running localized algorithms for transformers and individual homes. The purpose of these algorithms is to prevent transformer overload while charging for the lowest cost possible. The algorithms require the input of estimated end of charge and from there on calculating how much load can be moved to different times of the day to either prevent peaks or to lower costs. The algorithm for the Transformer only handles the connected households and the individual households manage themselves, this prevents overly complicated and time demanding calculations [5].

The Local Controller for this project can be considered a part of agent based charging as described. The main differences between the approach described in this paper and the Grønn Kontakt solution, is how loads are shifted in time. The information gathered from Charge Points does not include the state of Charge, nor is it planned to implement a time function for when the EV's should be fully charged. In the future, prioritized load could be a premium function where priority is based on how much a user is paying to charge a vehicle.

1.2.2 Challenges of the future Danish energy scenario with 50 % wind integration and how electric vehicles adaptive charging can help to mitigate.

This document is written by Adrian Sanchez Garcia at NTNU, Norway. It is about the potential of using the energy stored in EV-batteries both ways, to enable more intermittent wind energy in Denmark. Denmark aims for 50% wind energy by 2020 and this can be helped by the large battery bank that the total number of EV's represent [6].

While reversing energy flow from EV's to the grid is not part of this Master Thesis, it shows the potential of integrating logic in the EV domain.

1.2.3 OCPP Compatibility between a Central System and Electric Vehicle Charging Stations

This Master Thesis is written by Alexandre Court at KTH, Sweden. At this time OCPP version 1.5 was under development and the standard was unclear. The goal of this master was to test the protocol against 3 different EV-charger vendors and see if it is implemented in the same way [7].

This report is interesting because the vendor related differences is a potential problem in this project when implementing OCPP version 1.6.

1.3 Problem solution

The goal of this Master thesis is to implement a Local Controller with OCPP 1.6, and using this to control one or more Charge Points according to a simple algorithm. The total electricity consumption shall also be measured in the fuse box to account for other loads than the Charge Points. The algorithm will reduce load on the Charge Points with the highest electricity consumption until the total load is below a demand limit. The Controller should also fulfil the Grønn Kontakt requirements described at the beginning of section 3.

1.4 Methodology

This project will follow a SCRUM methodology. SCRUM is a framework for developing and sustaining complex products [8], and it fits a small team with a short timeframe. This is based on empirical process control where transparency, inspection and then adaption is important. Trello [9] is a web service that can be used for tracking progress and will be used to obtain the necessary transparency for inspection and adaption.

2 Theoretical Background

In this section I will introduce the Electrical Vehicle Supply Equipment (EVSE), referred to as a Charge Point and some basic understanding of the grid will be discussed.

2.1 The Grid

Some basic understanding of the grid is important to be able to calculate the currents and voltages in the different grid scenarios. In Norway, the grid is either IT, TT or TN, and in this section the relevant differences will be discussed.

2.1.1 IT, TT and TN networks and home transformers

IT and TT type networks both have 230 V between the phases and 400 V is not available. TN network uses 400 V between the phases and a “neutral” cable in addition to the 3 phase cables. It is possible to take 230 V between the phases and the “neutral” cable.

Most Charge Points are built for either 230 V single phase or 400 V three phase, this means that 3 phase charging is not available in IT or TT. Single phase charging is slower and causes large uneven loads on the grid. To achieve 3 phase charging in a IT or TT network, it is common to install a D-Y 230 V to 400 V transformer. See Figure 2-1 and Figure 2-2 for example of this [10].

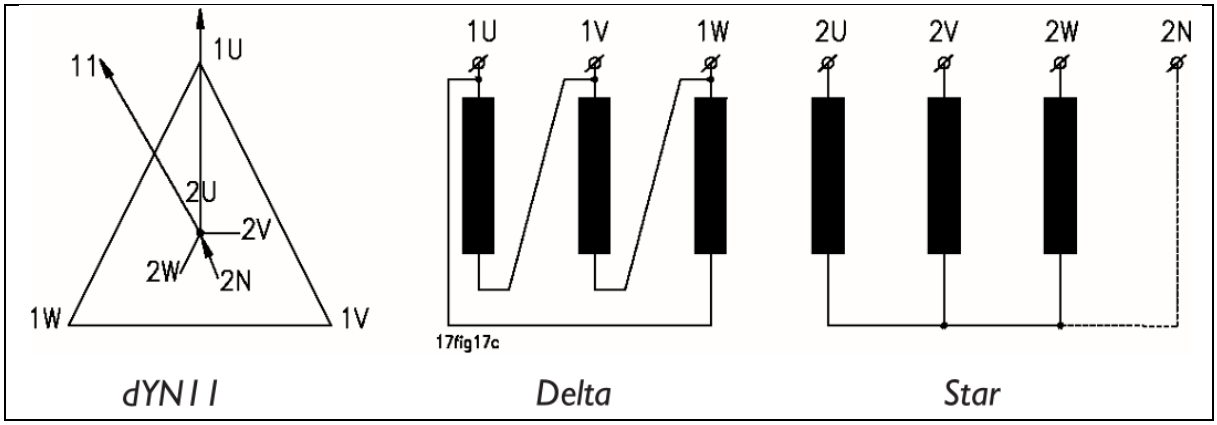


Figure 2-1 D-Y Transformer configuration

Figure 2-2 Picture of a home transformer

It is important to know if a transformer is installed because the current measured on the 3 main cables is larger than the equivalent current measured by the Charge Point.

2.2 Electrical Vehicle Supply Equipment (EVSE / Charge Point)

Charge Points are used to protect the user and ensure safe EV charging with a higher current. The Charge Point can also allow commercial use by measuring power used by the EV and logging the time used for charging. The Charge Point does not change the current in any way, but communicates with the EV internal charger who then changes the level of charging. The Figure 2-3 shows some examples of common Charge Point sizes. There are multiple Charge Point connectors available, but the most common are described in 2.2.1.

Power	Ampere	Connection
3680 w	16	Single Phase 230 V
7360 w	32	Single Phase 230 V
11085 w	16	Three Phase 400 V
22170 w	32	Three Phase 400 V

Figure 2-3 Table of common Charge Point sizes

2.2.1 IEC-62196 Charging Plug Type 1 and 2

Most European countries have chosen IEC 62196-2 standard [11] [12] [13] [14]. The Type 2 plug is a 7-pin connector with the configuration shown in Figure 2-4. The pins are Control Pilot, Protected Earth, Proximity Pilot, Neutral and 3 AC phase pins.

The type 1 plug is Interchangeable with the Type 2 plug and uses 5 out of 7 pins. These are Control Pilot, Protected Earth, Proximity Pilot and 2 pins for 230 V AC current. It is common for EV's with this kind of plug to use a cable with a type 2 plug on the Charge Point side.

None of the standards introduced here give the Charge Point any information regarding the state of charge for EV.

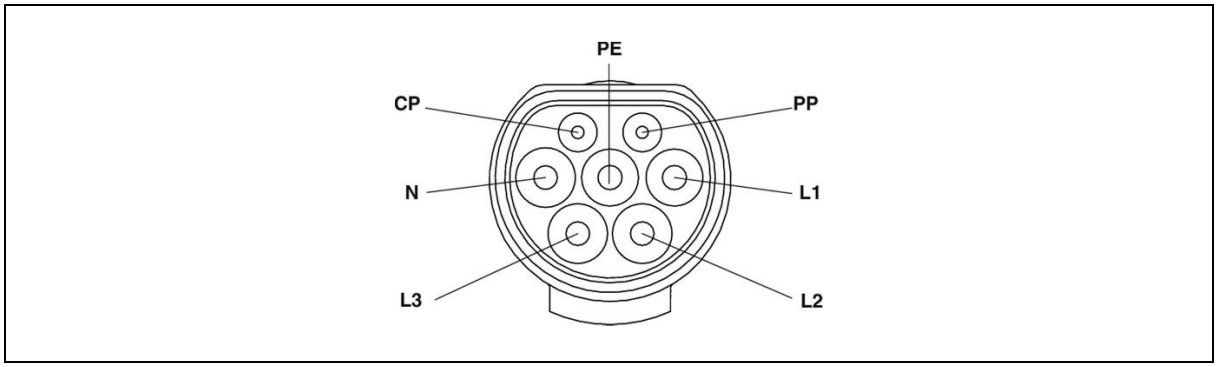


Figure 2-4 Type 2 Plug

2.2.1.1 Proximity Pilot (PP)

This pin has two different functions for the Charge Point and the EV. The EV measures resistance between PP and PE to indicate if a plug is present, this information is used to prevent movement while connected and arching caused by disconnecting while charging. The Charge Point measures resistance between PP and PE to receive information about the charging cable current rating. Resistance from Open or 1500 Ω to 50 Ω indicates a current rating between 6 A to 80 A.

2.2.1.2 Control Pilot (CP)

The control pilot is connected to the PE pin through a resistance in the EV. The size of the resistance informs the Charge Point about the states for the EV. Open loop means not connected, 2740 Ω means connected, 882 Ω means the EV can charge without ventilation and 246 Ω means the EV needs ventilation to charge. The Charge Point supplies a 12 V +/- peak-to-peak voltage signal between the pins, and the modulation of this signal informs the EV about the maximum charging capacity of the Charge Point.

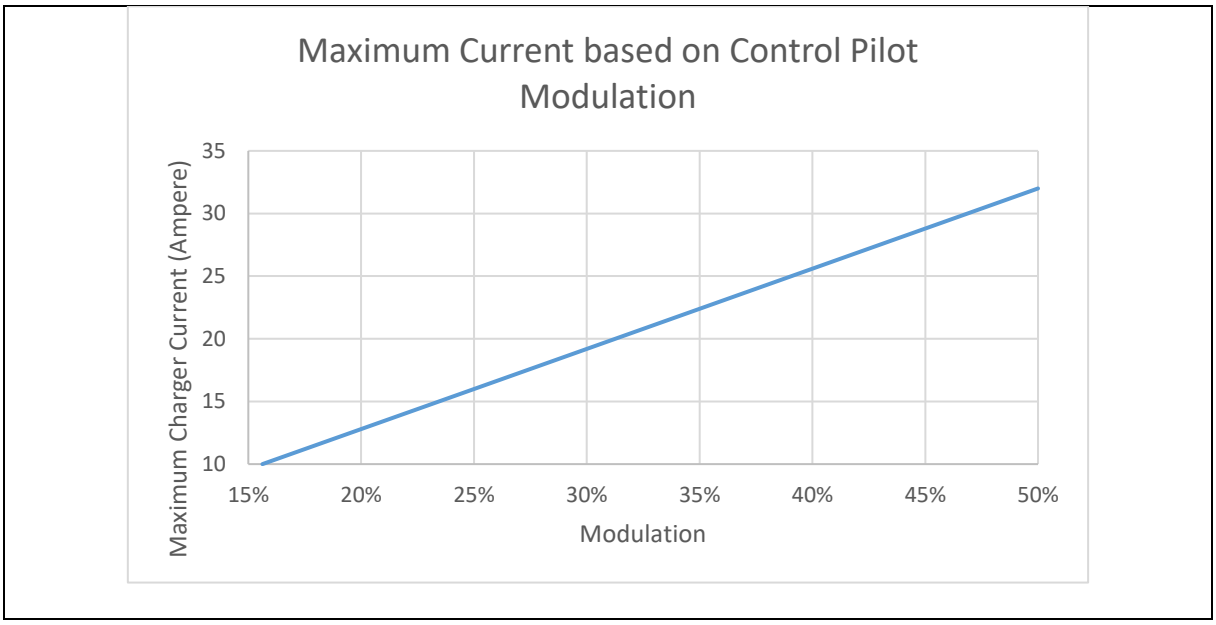


Figure 2-5 Control Pilot Modulation vs Ampere

2.2.1.3 Neutral, L1, L2, L3

These 4 pins are normally used for 230 V AC single phase charging between Neutral and L1, or 3 phase 400 V AC charging between L1, L2 and L3. There are also different configurations.

2.2.2 Schneider EVLINK Wall Box

The Schneider EVLINK Wallbox is an example of a Charge Point using a type 2 plug. This is a Charge Point supporting OCPP and has an identification reader on the bottom left hand side, see Figure 2-6 for example. The internal logic for this Charge Point runs in a 15 second loop, and can only return meter values and enforce charging limits after each 15 second period. This is the Charge Point used in all tests in this Master Thesis [15].

Figure 2-6 Schneider EVLINK WallBox

2.2.3 Zaptec Charge Point

The Zaptec Charge Point is a Norwegian product specially designed for the IT and TT networks common in Norway. This Charge Point transforms 230 V to 400 V and can be used for 3 phase charging without upgrading the home grid with a transformer. The Charge Point does not support OCPP, but uses a different method to balance multiple Charge Point in larger facilities. This makes the Charge Point irrelevant for this Thesis [16].

Figure 2-7 Zaptec Charge Point

2.3 OCPP

OCPP is an open communication protocol designed by the Open Charge Alliance [17]. This protocol is designed for communication between one or more Charge Points and a central server. The

advantage of this solution compared to proprietary solutions, is that you can control multiple Charge Points from different vendors.

2.3.1 OCPP 1.5

OCPP version 1.5 is currently in commercial use. This protocol can authorize users by identification to allow charging, and then measure how long and how much they have charged an EV. A central service can then use this protocol for billing purposes. This version of OCPP is based on SOAP web service.

2.3.2 OCPP 1.6

OCPP version 1.6 has the same base functionality as 1.5, but it also supports load control. The OCPP protocol contains 10 messages originating from the Charge Point to the Local Controller and 19 messages originating from the Local Controller going to the Charge Point. OCPP 1.6 messages have 2 different structures depending if it is a call or the result response to a call message.

Call message:

```
[<MessageTypeId>, "<UniqueId>", "<Action>", {<Payload>}]
```

Result message:

```
[<MessageTypeId>, "<UniqueId>", {<Payload>}]
```

The ”MessageTypeId” is an integer from 2 to 4 where 2 identifies the message call, 3 identifies the message as a response and 4 is an error. ”UniqueId” identifies which call and result are matching, as there is always a result to a call message. ”Action” identifies the type of message, for instance an “BootNotification”. “Payload” contains further message specific information and is described in section 3.3.2 in this thesis [18].

3 Solution

At the start of this project Grønn Kontakt issued a list of requirements that the Local Controller should fulfil.

3.1.1 OCPP 1.6

The local controller shall use OCPP 1.6 to communicate with Charge Points. This requirement is there to enable control of Charge Points by multiple vendors.

3.1.2 Current measurement

The local Controller shall be able to measure current on all phases entering a charger area, the rating of this measurement shall be at least 64 Ampere. The measurement shall handle both 230 V IT or TT networks and 400 V TN networks. This requirement is there to enable control of the current to stay below a predefined limit set by local fuse size. The sample interval shall be maximum 1 second to ensure a rapid response in case of increasing loads.

3.1.3 Current Control

The Local Controller shall be able to control the current used by Charge Points, this is a functionality of OCPP 1.6.

3.1.4 Communication with Grønn Kontakt

The Local Controller shall be able to communicate with a Grønn Kontakt central server. This requirement is there to obtain billing information and enable remote start of charging by telephone

application. The communication to the central server shall not require configuration of the customer's router.

3.1.5 Non-proprietary solution

The Local Controller shall not be based on a proprietary solution. This requirement is there to hinder a lock effect that can occur if you base a product on a specific vendor.

3.1.6 Fast and easy installation

The product shall be easy to install to ensure that the electrician who installs the Charge Point can install the Local Controller as well.

3.1.7 Remote Update

Grønn Kontakt shall be able to remotely update the Charging Controller with new software.

3.1.8 Low Cost

The cost of a Local Controller unit shall be low.

3.1.9 CE rating

The Local Controller shall have a CE rating. A CE rating is a mandatory conformity marking within the European Economic Area.

3.2 Design Specification

In this section, 3 different hardware solutions are considered to fulfil the requirements described in the beginning of section 3. A GCU1000, Develco and a Raspberry Pi based solution was considered. The raspberry Pi was found to be the only platform to fulfil all the requirements, and was therefore chosen.

3.2.1 Grid Controller GCU100 Solution

The GCU100 Grid Controller is a premade device that measure electricity and control Charge Points. The controller uses OCPP 1.5 and can therefore not limit the Charge Points, only turn them on or off. This solution communicates with a OCPP 1.5 web server called Charge Storm that is currently used by Grøn Kontakt, therefore, little modification is needed [19].

Figure 3-1 GCU Grid Controller

3.2.2 Develco Solution

The Develco solution consists of a Develco controller and a Develco External meter interface or a Develco Prosumer Meter [20].

3.2.2.1 Develco Squid.link Gateway

The platform for this project is the Squid.Link Gateway [1]. This is a modular platform that supports several connection protocols. Supported protocols include WiFI 3G, ZigBee, Z-Wave and Wireless M-bus. The gateway comes with Linux and supports Java programs. The flexibility of handling many different protocols and Java programming, enables the Gatway to be the core of a home automation system regardless of brand and technology [20].

Figure 3-2 Develco Gateway

3.2.2.2 Develco External Meter Interface

The Develco External Meter Interface measures electricity consumption based on the blink frequency of the electricity meter. This sensor reports the current summation at 5 second intervals. The sensor is battery driven using 3 AA Alkaline batteries with an expected battery lifetime of two years with 5 second update intervals. It can optionally be driven by a 5V power supply. This sensor communicates with the Squid.link gateway from Develco Systems using the Zigbee protocol. This solution is limited to a 5 second resolution which is too slow to respond to sudden load changes. This is also an overall measurement based on the electricity meter instead of measuring individual phases [20].

Figure 3-3 Develco External Meter Interface

3.2.2.3 Develco Prosumer Meter

The Develco prosumer(REF) meter is 3x3 400V phase meter. It is intended for use in homes with solar power production to measure both the home consumption, home production and grid power. It communicates with the Squid.Link Gateway by ZigBee radio. This product is limited by its 60 A per phase rating because 63 A fuse size is common in Norway. This solution is invasive and the power intake of the household needs to be modified by an electrician [20].

Figure 3-4 Develco Prosumer Meter

3.2.2.4 Develco solution summary

The Develco solution only excels if the ZigBee sensors that the platform supports are utilized. The sensors do not fulfil the project requirements and cannot be used for this purpose. This is also a proprietary solution and Grønn Kontakt expressed that this is not desirable.

3.2.3 Raspberry Pi Solution

The Raspberry Pi solution uses a Raspberry pi 3B with an added shield through the GPIO pins. The shield is connected to 3 non-invasive current sensors. The raspberry Pi is a credit card-sized PC with 4 USB ports, HDMI, Ethernet and 40 GPIO pins. The GPIO pins can be configured for different types of input and output. Raspberry Pi is a commonly used platform and there are much documentation and many libraries simplifying the implementation [21].

3.2.3.1 Non-invasive CT sensor and shield

A non-invasive CT sensor is a clamp that can be fitted around a live wire without opening the circuit as shown in Figure 3-6, this eases installation in the fuse box. The sensors function as a current transformer and returns a current linear to the current in the main wire. The current passes a resistor, and the voltage across this is measured in an attiny85 microchip. The Attiny85 reads 1000 samples and calculates RMS current from these samples. The Attiny85 then transmits the RMS current to the Raspberry PI by RS232 communication, Figure 3-5 describes the setup. The full schematic for the shield is added in Appendix 1.

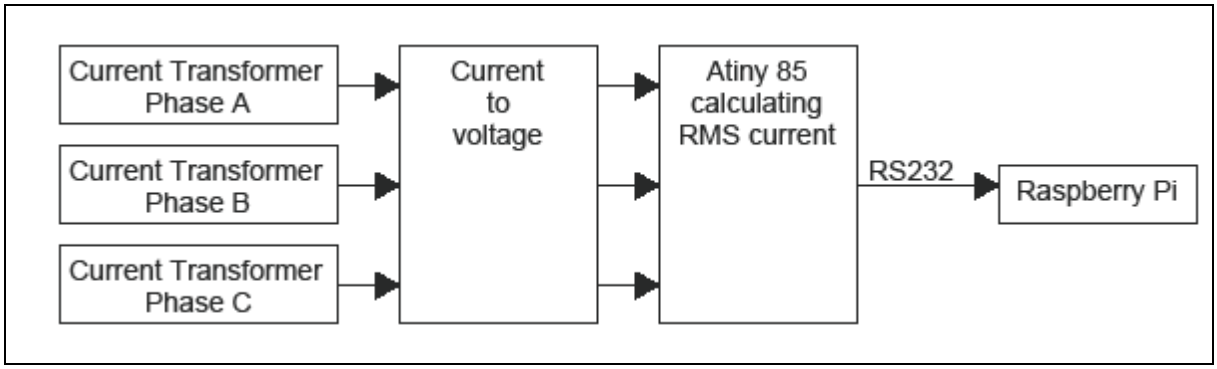


Figure 3-5 Current Measurement Shield

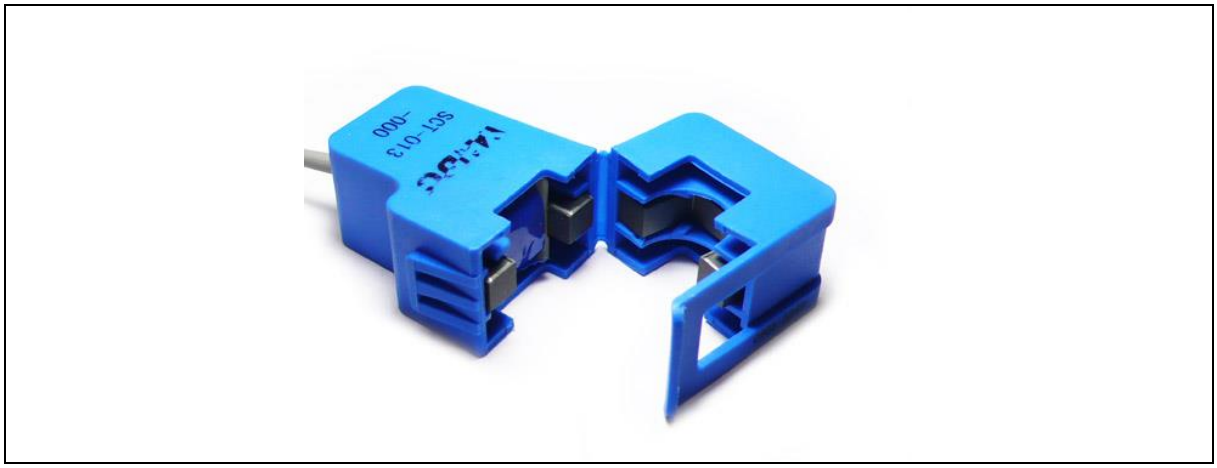


Figure 3-6 CT sensor

3.2.3.2 Raspberry solution summary

The raspberry solution was chosen because it is a low-cost solution that fulfils the project requirements. As this project has a short timeframe the additional support for Raspberry is also helpful. A potential problem with this solution is obtaining a CE certification. The Raspberry and the current measurements are CE rated, but the shield, and the whole solution is not. Figure 3-7 is a raspberry mounted with the shield for current measurement and 2 connected sensors.

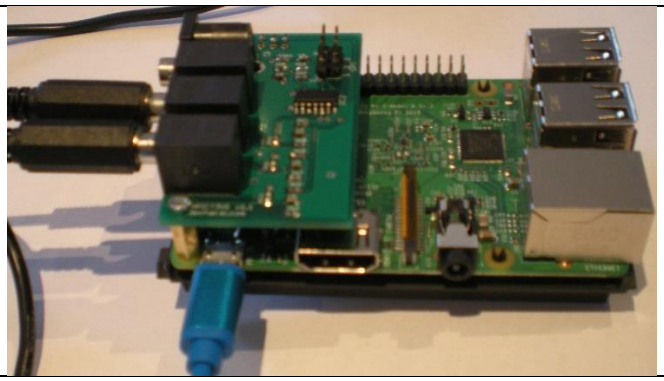


Figure 3-7 Raspberry with shield

3.2.4 Software Solution

After looking at a few software alternatives we found a premade JAVA library for OCPP 1.6 [22]. This library had large parts of OCPP implemented and tested, and was therefore chosen as programming language.

3.3 Implementation

The Raspberry Pi Local Controller measures current on the three main phases CT sensors. The Charge Points will be connected by Ethernet cable and controlled by OCPP 1.6. Based on data from the Charge Points and the CT sensors the Local Controller will limit the electricity delivered by the Charge Points. The Grønn Kontakt server will be reached by WiFi trough customer internet. Figure 3-8 describes the setup with 230V as example, this can also be 400V three phase.

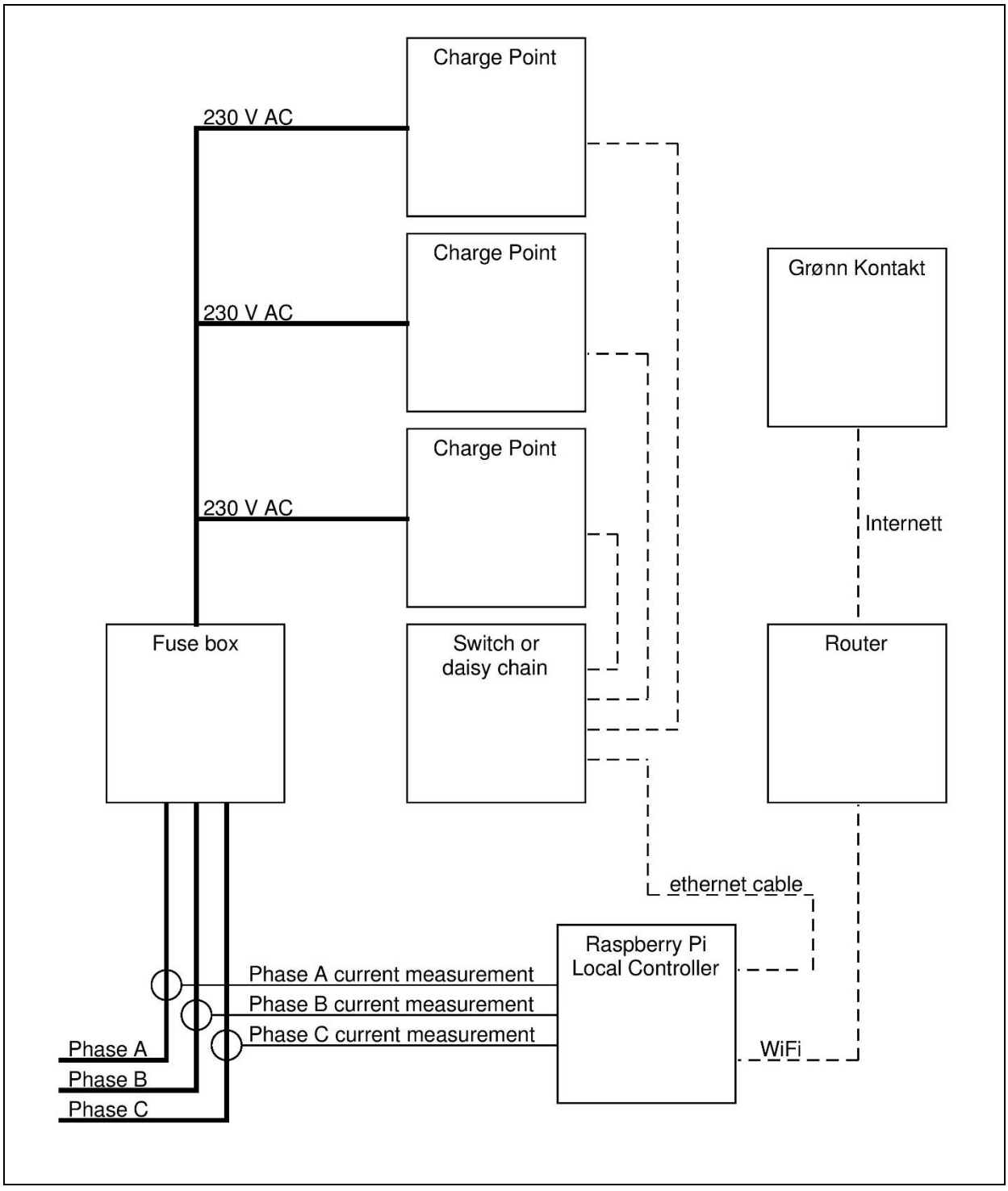


Figure 3-8 Overall Setup

3.3.1 OCPP Charge Flow

The OCPP protocol expects charging to happen in a set order and has defined the steps of charging in Figure 3-9, this figure is from the OCPP protocol manual [18]. Session is a period from the first contact between a user and the Charge Point, this will be either a telephone Application or a chip used locally on the Charge Point. Both the local chip and the remote identification is referred to as authorization. A transaction starts when all conditions for charging are fulfilled, this typically means the user is authorized and has connected the EV to the Charge Point.

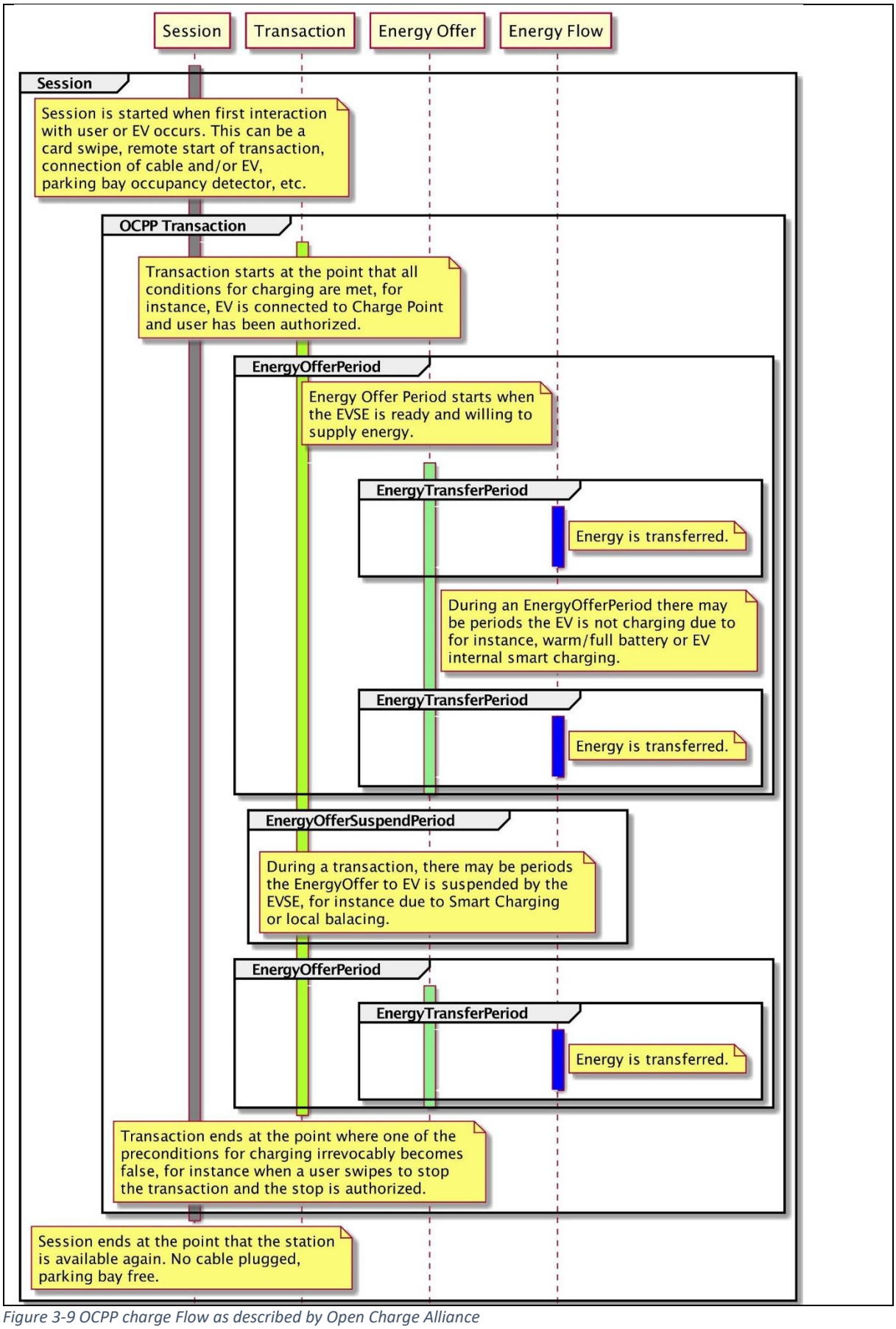


Figure 3-9 OCPP charge Flow as described by Open Charge Alliance

3.3.2 Detailed message description

The local controller is not only about OCPP, it also communicates with a Grønn Kontakt central server. This central communication is a unique API but will reflect the OCPP messages. A normal charging will be initiated by user authorization, this can either be remote by telephone application or locally with a chip. Once a user is authorized and has connected a EV to the Charge Point, a Start Transaction message is sent. When the charging is done, and an EV is disconnected, a Stop Transaction message is sent. These two Transaction message contains meter information and will be used for billing purposes. If the Local Controller is disconnected from the Grønn Kontakt server, then transaction related messages will be buffered and sent when the connection is restored. Figure 3-10 describes a normal charging cycle including both the OCPP messages and the API between the central server. The sections 3.3.2.1 to 3.3.2.9 will explain in detail the content of the messages.

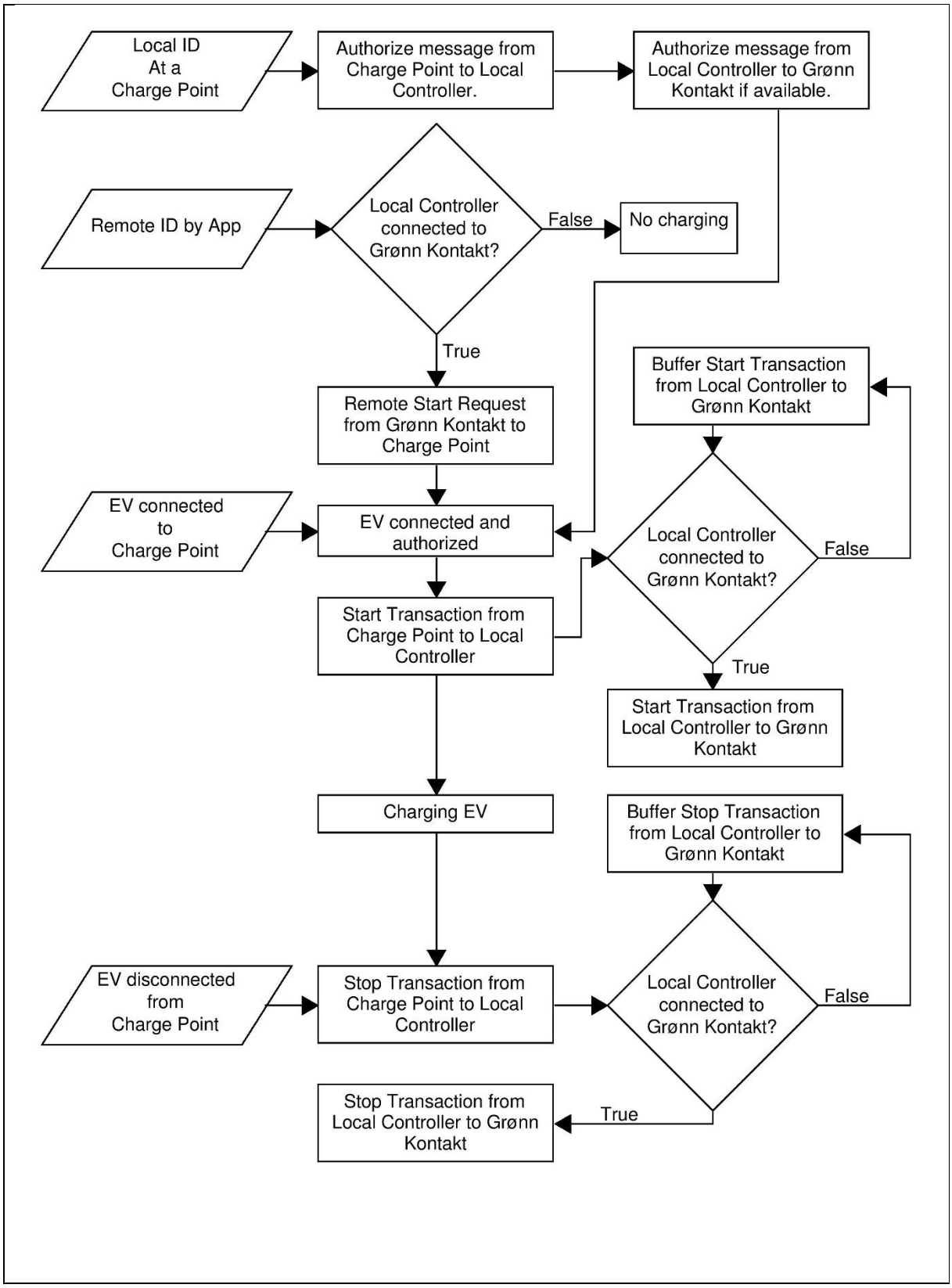


Figure 3-10 Normal Charging Cycle

3.3.2.1 Boot Notification

Boot Notification is sent on Charge Point boot or on reconnect, if the local controller has lost communication with the Charge Point. When the Charge Point connects, the name of the Charge Point is also given as part of the connection address, for example 192.168.1.242:8080/CP001.

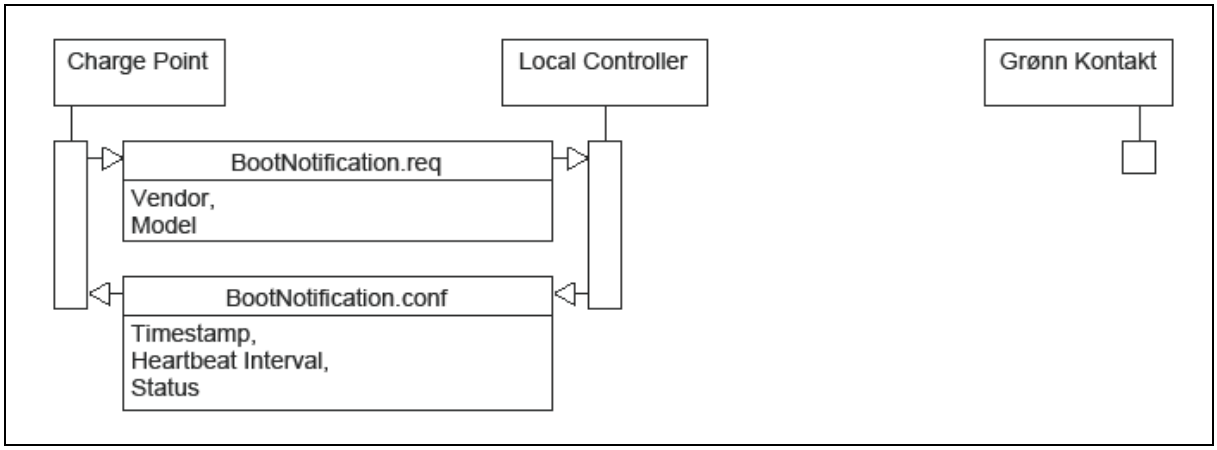


Figure 3-11 Boot Notification

The boot notification carries the following parameters:

Vendor: The producer of the Charge Point

Model: The model of the Charge Point

Heartbeat Interval: The interval between heartbeats.

Status: The status of the Boot Notification, can be Accepted, Pending or Rejected.

3.3.2.2 Heartbeat

Heartbeat is sent at intervals set in Boot Notification message between Charge Points and the Local Controller. The Local Controller also has a separate heartbeat with the Grønn Kontakt server.

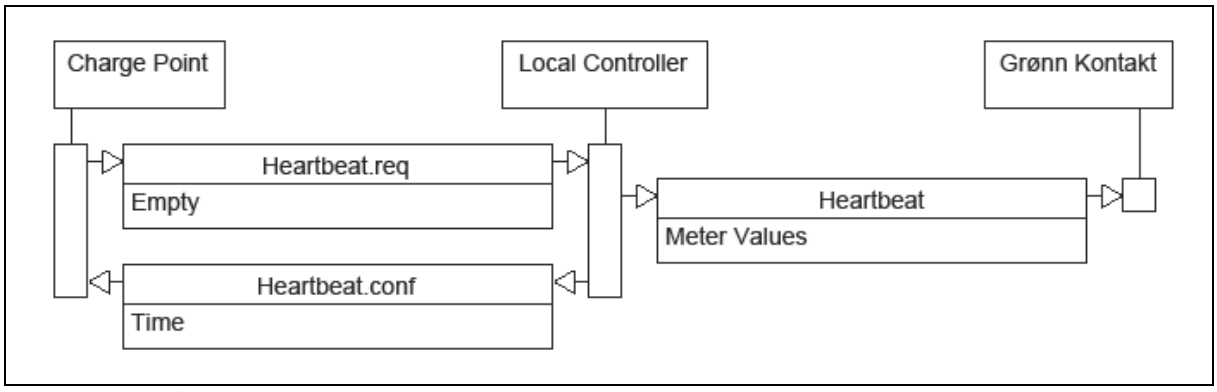


Figure 3-12 Heartbeat

The heartbeat carries the following parameters:

Time: The time of the local controller, can be used for time synchronization.

Meter Values: Contains current measurements on the 3 main phases.

3.3.2.3 Authorize

This message is sent if a user uses chip at a Charge Point. If the Local Controller has lost contact with Grønn Kontakt then the Authorize is automatically accepted.

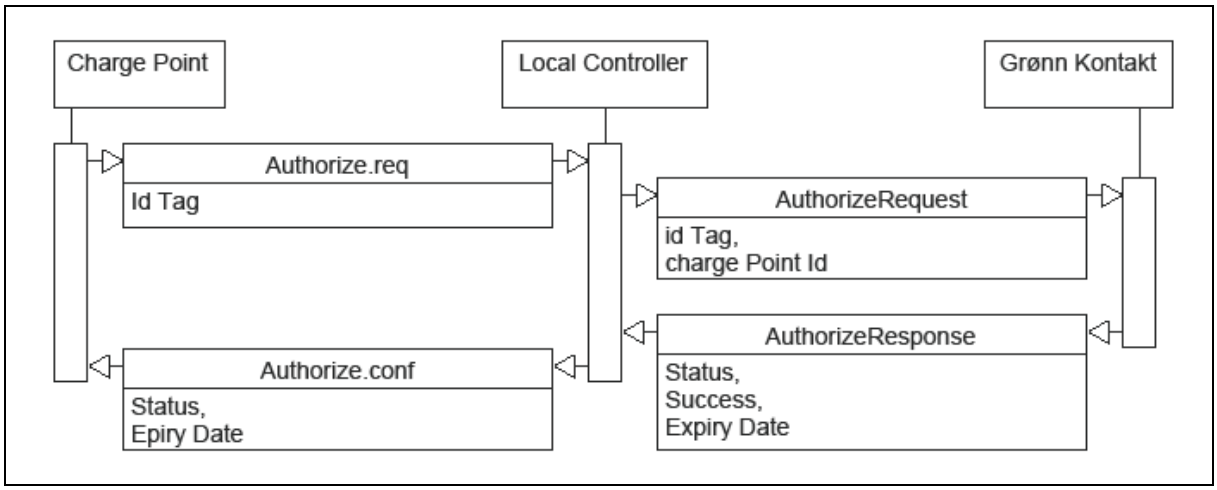


Figure 3-13 Authorize

The authorize carries the following parameters:

IdTag – Identification string.

Charge Point Id: Identification string of the Charge Point.

Success: can be True or False

Status: The status of the authorize request, can be accepted, blocked, expired or invalid.

Expiry Date: The date when the Authorization expires.

3.3.2.4 Start Transaction

This message is sent when a car is connected and authorized. It is used to inform the Local Controller that Charging has started, and log billing relevant information. If the Local Controller has contact with Grønn Kontakt, this information is passed on. If the Local Controller is offline from Grønn Kontakt, the transaction information is buffered and will be sent when connection is restored.

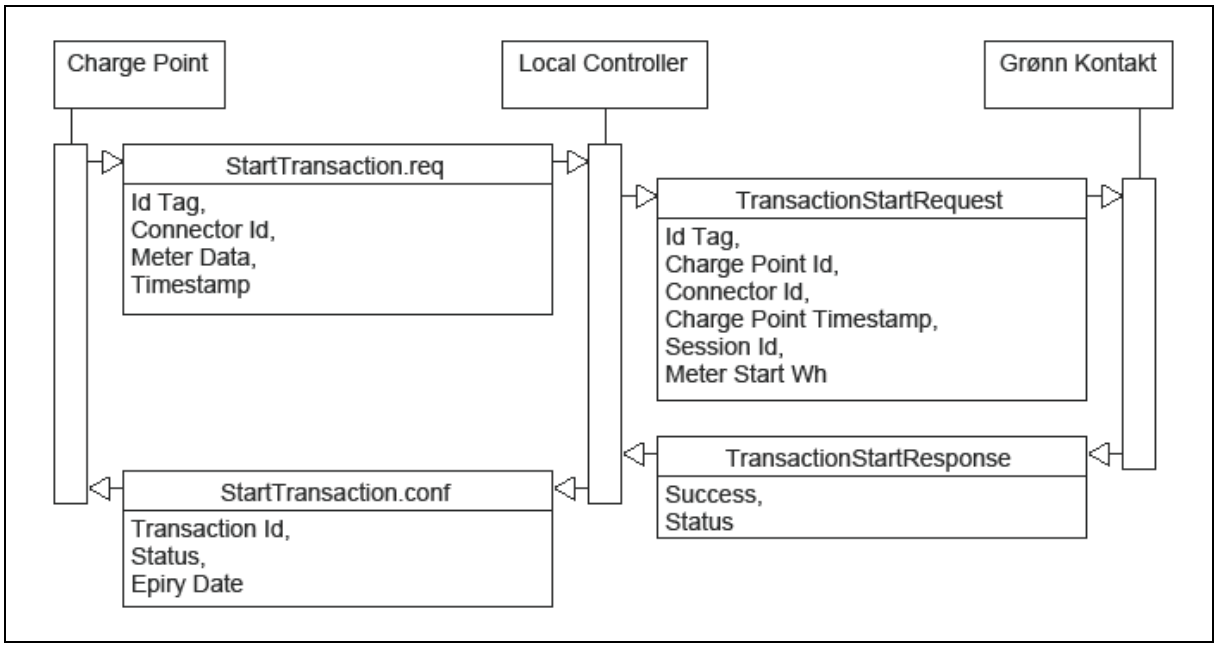


Figure 3-14 Start Transaction

The start transaction carries the following parameters:

IdTag: Identification string.

Meter data: Contains the meter data in Wh at the start of a transaction.

Timestamp: Timestamp at transaction start.

Status: The status of the transaction request, can be accepted, blocked, expired or invalid.

Transaction Id: A identification number for the transaction.

Expiry Date: The date when the Authorization expires.

Charge Point Id: Identification string of the Charge Point.

Charge Point Timestamp: the timestamp from the Charge Point at transaction start.

Connector Id: Identification number of the connector.

Session Id: The identification of the charging session.

Meter Start Wh: The meter value at start of a transaction in Wh.

Success: Can be True or False.

3.3.2.5 Stop Transaction

This message is sent when a car stops charging. It informs the Local Controller that Charging has stopped and log billing relevant information. If the Local Controller has contact with Grønn Kontakt, then this information is passed on, if the Local Controller is offline from Grønn Kontakt, then the transaction information is stored and will be sent when connection is restored.

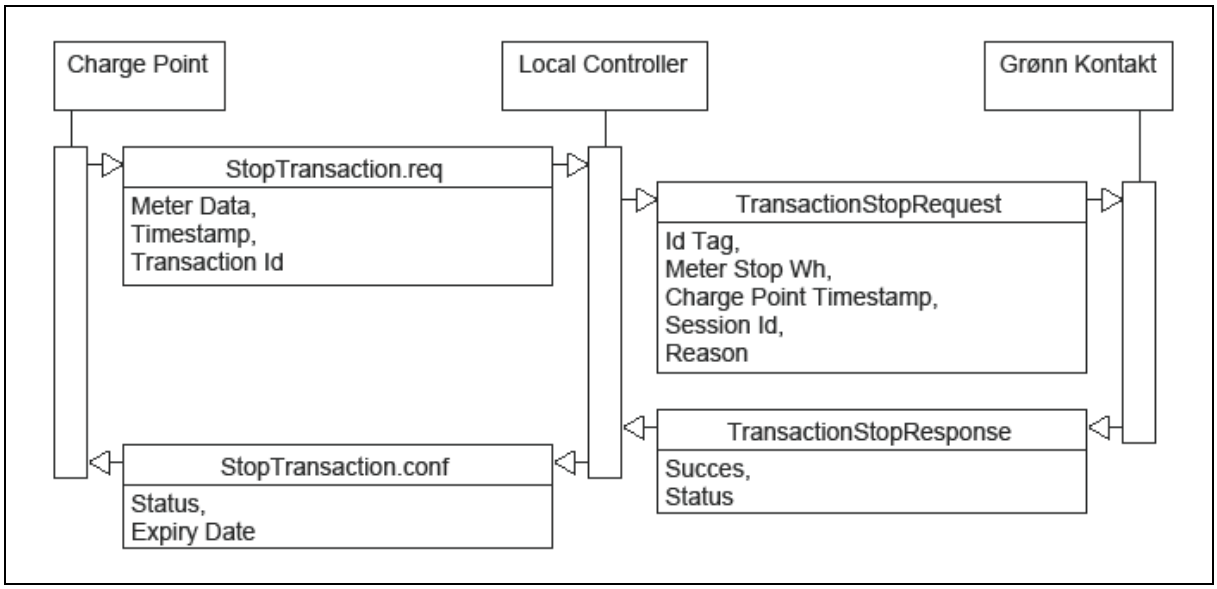


Figure 3-15 Stop Transaction

The stop transaction carries the following parameters:

Meter data: Contains the meter data in Wh at the stop of a transaction.

Timestamp: Timestamp at transaction start.

Transaction Id: A identification number for the transaction.

Status: The status of the transaction request, can be accepted, blocked, expired or invalid.

Expiry Date: The time when the Authorization expires.

IdTag: Identification string.

Meter Stop Wh: The meter value at the end of a transaction in Wh.

Charge Point Timestamp: the timestamp from the Charge Point at transaction start.

Session Id: The identification of the charging session.

Reason: Can be any of the following, EmergencyStop, EVDisconnected, HardReset, Local, Other, PowerLoss, Reboot, Remote, SoftReset, UnlockCommand, DeAuthorized

Success: Can be True or False.

3.3.2.6 Meter Values

Meter values are sent at predefined intervals or when a trigger message is received. This message informs the Local Controller how much power is being used by the charger. The information is used in the controlling algorithm to limit current.

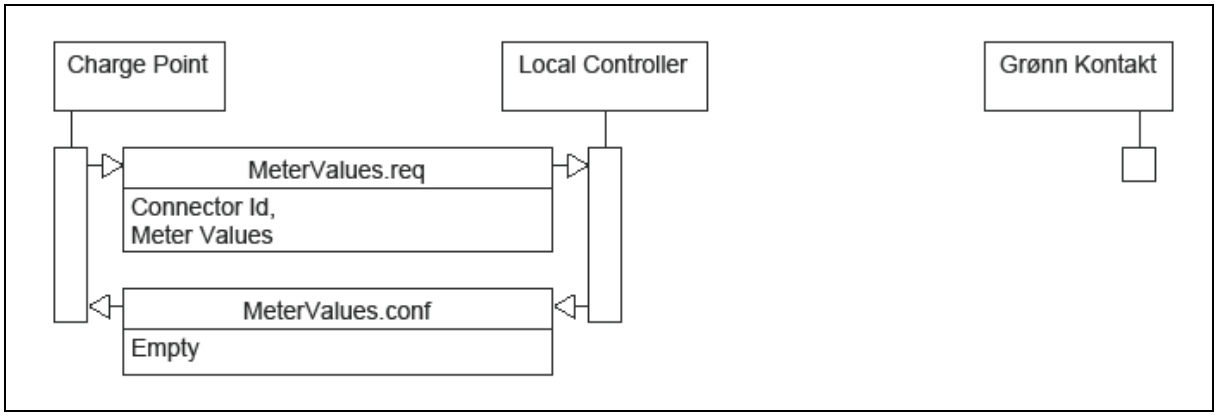


Figure 3-16 Meter Values

The meter values carry the following parameters:

Connector Id: Identification number of the connector.

Meter Values: Contains 2-dimensional array. The first array is based on time, the second internal array is different measurements taken at a specific point in time. This can include measurement at different phases, current to and from the EV and a few additional options.

3.3.2.7 Remote Start Transaction

This message is initiated when a user authorizes charging from a App. A message is sent from the Grønn Kontakt server to the Local Controller, and from there to the Charge Point. This message triggers the Start Transaction message. This message will not work if the Local Controller is offline from Grønn Kontakt.

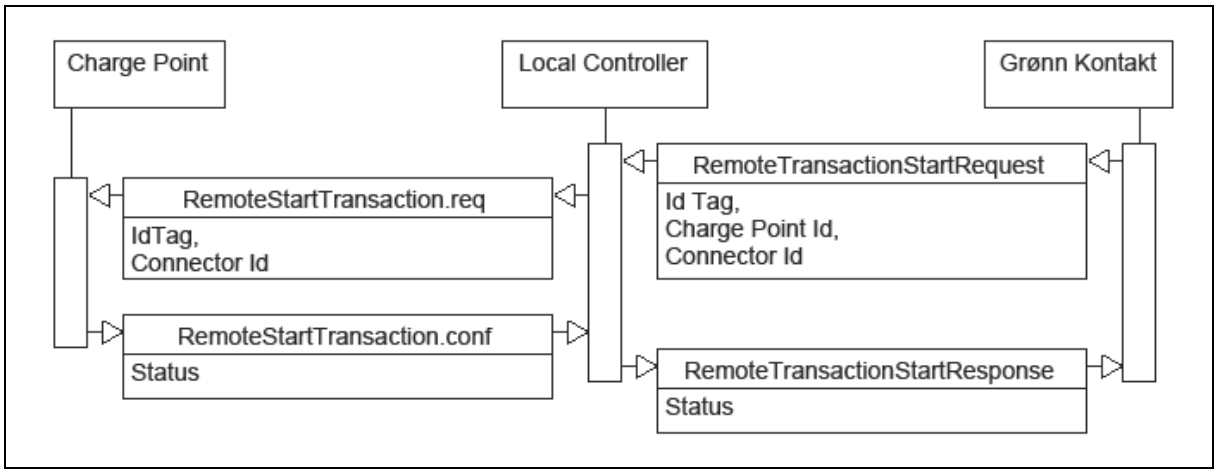


Figure 3-17 Remote Start Transaction

The remote start transaction carries the following parameters:

IdTag: Identification string.

Connector Id: Identifies the connector if a Charge Point has more than one connection.

Status: The status of the remote start transaction request, can be accepted or rejected.

Charge Point Id: Identification string of the Charge Point.

Connector Id: Identification number of the connector.

3.3.2.8 Remote Stop Transaction

This message is initiated when a user stopps charging from a App. A message is sent from the Grønn Kontakt server to the Local Controller, and from there to the Charge Point. This message triggers the Stop Transaction message. This message will not work if the Local Controller is offline from Grønn Kontakt.

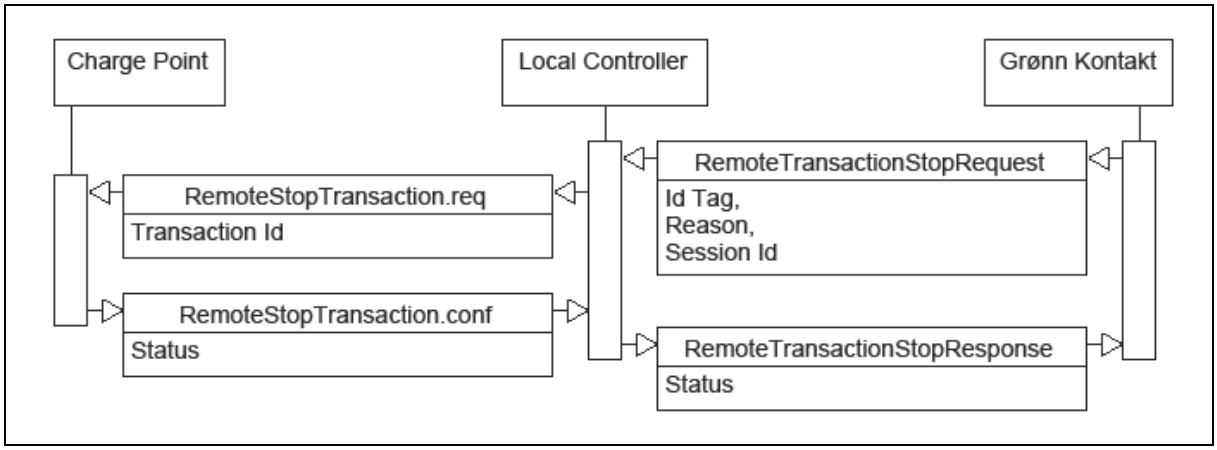


Figure 3-18 Remote Stop Transaction

The remote stop transaction carries the following parameters:

Status: The status of the remote start transaction request, can be accepted or rejected.

Transaction Id: A identification number for the transaction.

IdTag: Identification string.

Reason: EmergencyStop, EVDisconnected, HardReset, Local, Other.

Session Id: The identification of the charging session.

3.3.2.9 Set Charging Profile

Sets a profile with an output limit on the Charge Point. This message is sent every second while there is an ongoing transaction.

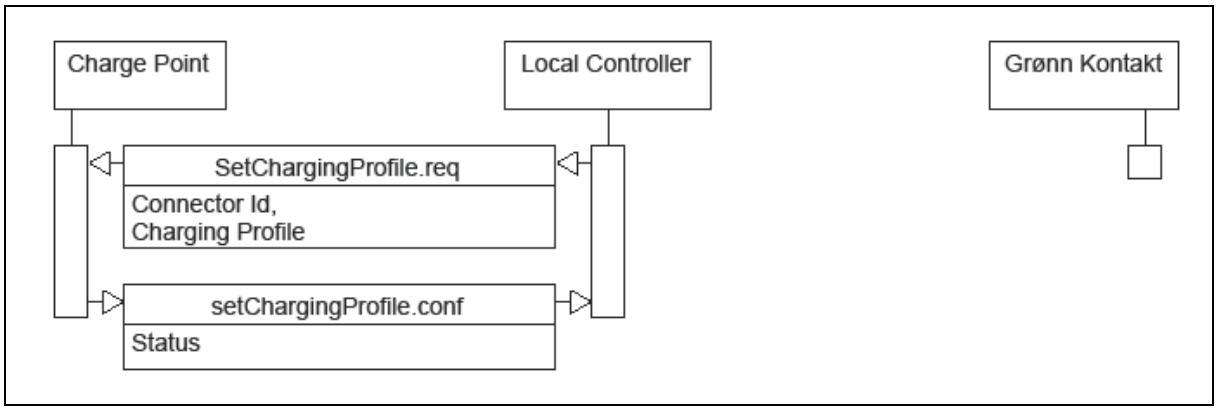


Figure 3-19 Set Charging Profile

The charging profile carries the following parameters:

Connector Id: Identifies the connector if a Charge Point has more than one connection.

Status: The status of the remote start transaction request, can be accepted or rejected.

Charging Profile: The charging profile is described in below.

OCPP uses charging profiles to limit the electricity consumption of Charge Points. A charging profile is set from the Local Controller and has several parameters in addition to a charging limit. These parameters allow a Charge Point to have multiple profiles that are active depending on time and priority. It is also possible to set low priority “default” profiles that will be used if no other profile is used.

The Charging Profiles can be used to limit current at pre-set times of the day, for instance based on a variable electricity cost. The full functionality of the charging profiles will not be utilized in this project. A charging profile with an electricity limit will be set with a short duration for each cycle of the limitation algorithm.

3.3.3 Electricity Regulation

The electricity regulation is an algorithm running once per second. The Inputs used, are the current data from the CT-clamps and meter data from the Charge Points. The outputs are individual electricity limits set on each Charge Point connector.

The algorithm uses the highest of the 3 CT current measurements in the logic, and does not consider which phase the load is actually placed between.

Section 2.2 describes the information flow between an EV and a Charge Point. The state of charge for the EV is not transferred, and if a charging limit is enforced, then the current the EV would have used without a limit is not transferred. Any algorithm must function without knowing this information. It must also be considered that some EVs require a minimum current to charge at all.

To set limits on the Charge Point connectors, we first calculate how much current is available. To do this, we use the following formula:

$$I_{Available} = I_{Fuse} - I_{CT\ clamp} + I_{EV} - I_{Headroom}$$

$Where$

$I_{Available} = The\ available\ current\ for\ EV\ charging$

$I_{CT\ clamp} = The\ highest\ current\ measured\ by\ the\ three\ CT\ clamps$

$I_{Fuse} = The\ fuse\ size\ limiting\ higher\ current.$

$I_{EV} = The\ current\ used\ by\ all\ charging\ EVs$

$I_{Headroom} = The\ headroom\ given\ to\ avoid\ blowing\ a\ fuse.$

When the available current is known, we can start calculating the limit for each charge point. This calculation is a three-step process because there are several factors to consider.

Charge Points can use variable amount of electricity, and it is desirable to avoid sudden jumps in the EV electricity consumption. To avoid this, we start by setting the individual Charge Point limit to the electricity consumption from the last received meter data with an added buffer on top. During the test phase of this project this buffer was 4 A, but this might change with further testing of the algorithm. The controlling algorithm is run once per second, and this logic will cause the Charge Point to slowly step up the consumption if the initial limit is too low.

The second step is to ensure each charger is set above a minimum limit. This is because most EVs require a minimum amount of electricity to charge at all. During testing this minimum limit was set to 10 A, but some EVs might require a higher setting.

The third step is to reduce the limit for the EVs with the highest electricity consumption until we are below the calculated available current for charging. When the limits for all EVs are calculated, the Local Controller will use Set Charging Profile messages to change the limit for all Charge Points. During testing the Set Charging Profile message was not implemented in the Schneider Charge Point, as the Charge Point firmware was in beta version. A Change Configuration message was used instead, but the result should be the same.

Figure 3-20 represents the control flow of a single iteration.

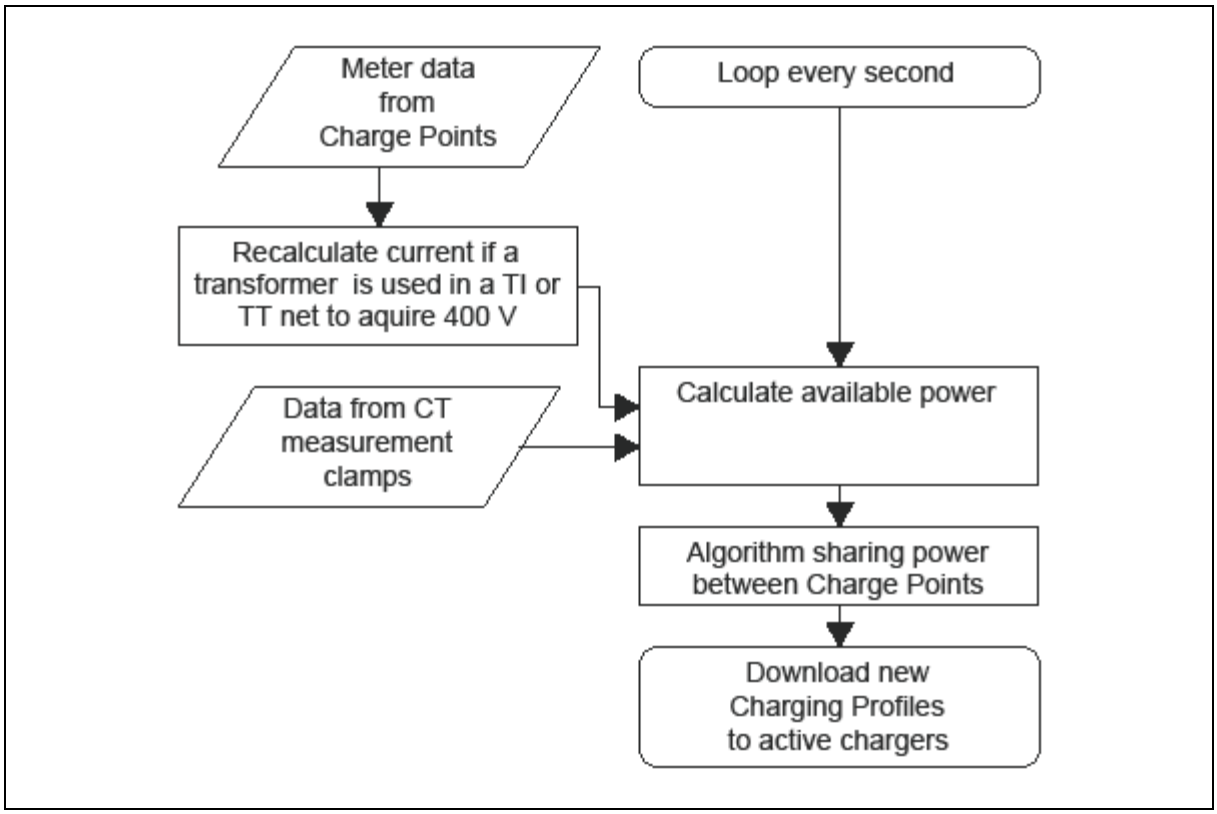


Figure 3-20 Current Regulation Flowchart

Figure 3-21 and Figure 3-22 shows a theoretical output of the algorithm applied to 5 chargers with variable charging level in a facility with a 63 A main fuse and 5 ampere headroom. Charge Point 1 is limited to the minimum of 10 A, but only uses 5. Charge Point 2 and 3 are both set to electricity consumption with a 4 A buffer. Charge Point 4 and 5 are limited to 16.5 A to keep the total electricity consumption at 58 A, which is the fuse limit minus the headroom.

Charger	Limit set by Local Controller	Current used by EV	Current used by EV without limitation
Charger 1	10	5	5
Charger 2	12	8	8
Charger 3	16	12	12
Charger 4	16,5	16,5	20
Charger 5	16,5	16,5	25
Sum	**63**	**58**	**75**

Figure 3-21 Example Regulation Table

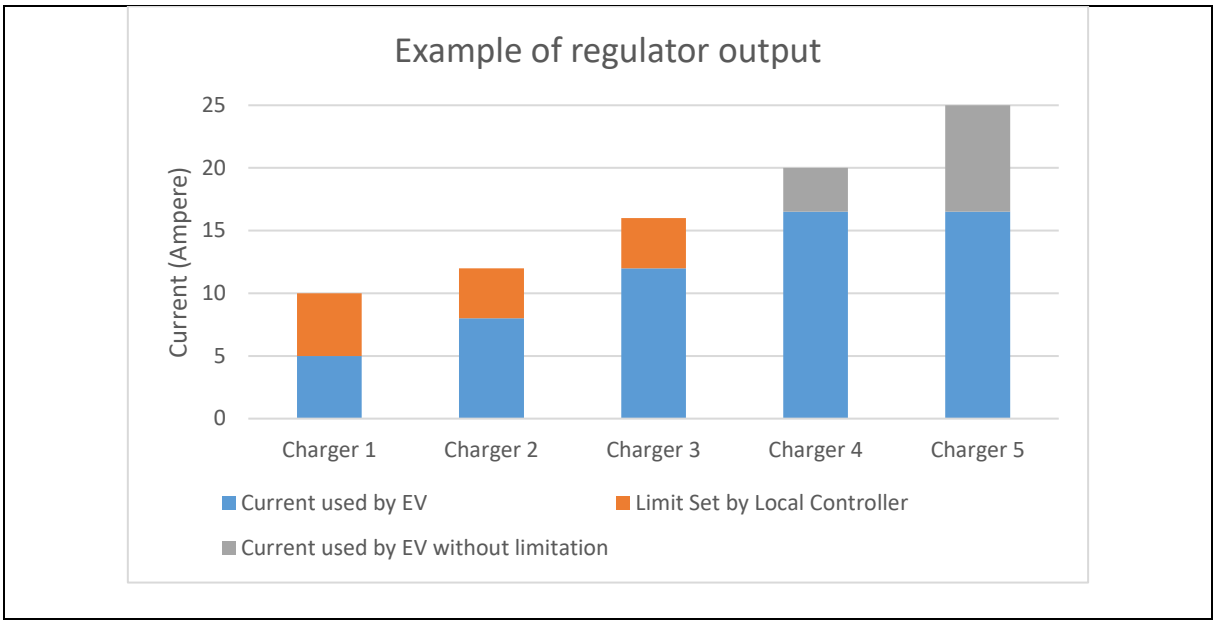


Figure 3-22Example Regulation Graph

3.3.4 Software architecture

The controlling software running on the Raspberry Pi local controller is programmed in Java using a build development called IntelliJ [23]. A project management add called Maven was used to handle all build dependant packages used in the development process [24]. The final program is converted to a runnable JAR file that runs on the Raspberry Pi. To handle version control, Bitbucket and GIT was used. Version control is particularly important when multiple people are working on the same software.

There are other functions installed on the Local Controller that is not directly related to the Charge Point Control. One of the functions planned, but not currently implemented, is a private update server. All Local Controller's will periodically check the server for updates using Linux apt-get. This function will allow remote update of all Local Controller. Another function is a web interface for easy installation. This is explained in section 3.3.4.1.

3.3.4.1 Installation of Local Controller

To ease installation of the Local Controller, a web interface will be available. This web interface can be connected to by WiFi, distributed by the Local Controller. When connected, you will be forwarded to a configuration page where statuses like the current measurements of the main wires and connected Charge Points will be shown. The web interface is also used to connect the local controller to the customer network. Figure 3-23 part 1 shows this.

A different web page will be displayed when the local controller is connected to a customer network. This is shown in Figure 3-23 part 2. Location based settings will be available in this second web page. This includes the network type, TT, IT or TN, and individual voltage settings for each charger in case they are connected in different ways. The combination of 3 phase 400 V Charge Point and IT or TT networks indicates a home transformer is present. It also has settings for fuse size and headroom. These will be used for the limitation algorithm. All these settings will be saved in a configuration file on the local controller.

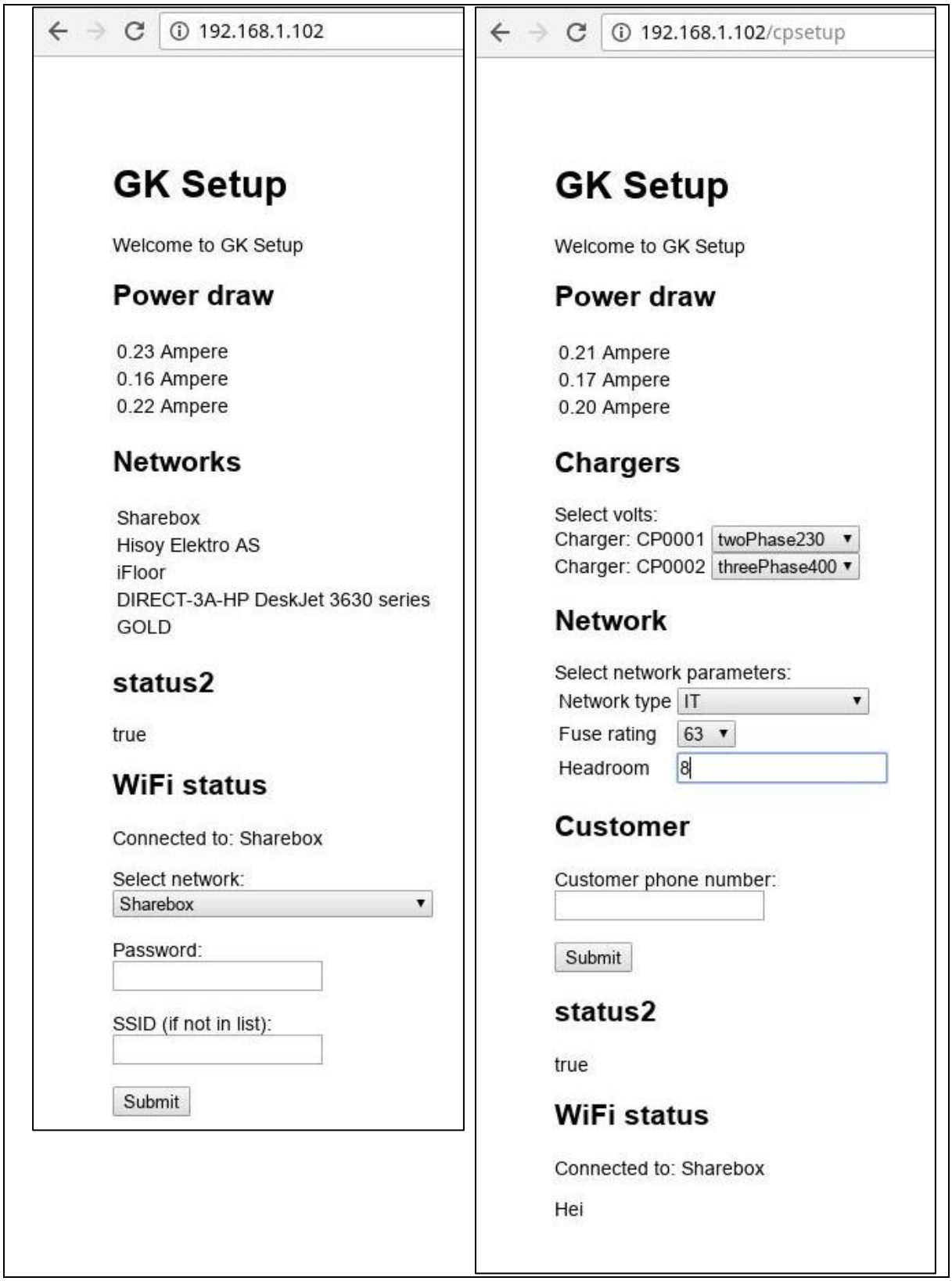


Figure 3-23 Web configuration of controller part 1 (left) and part 2 (right)

3.3.4.2 OCPP library

The foundation of the local controller java program is a OCPP 1.6 library [22]. This library handles all the OCPP messages and creates threads for each message. The library also has checks to see if a message has all the required parameters set. To interface this library, we use java override methods.

An example of the code is added in appendix B for the Meter Values message. When the Local Controller receives a message from a Charge Point, the OCPP library will call a thread with the message content added as a java object. In the override function, it is added logic to handle any actions by the message content. The thread will also return the message response to the Charge Point.

3.3.4.3 Server for OCPP messages

The java program runs a thread server that responds to incoming OCPP messages and can also send OCPP messages. When a Charge Point connects to the local controller OCPP server, the program will create a java object for that Charge Point. A Charge Point object can have several connector objects that represents the actual Connectors. Each connector has a set of parameters that identifies the current state of charging. The program is event driven in response to incoming messages.

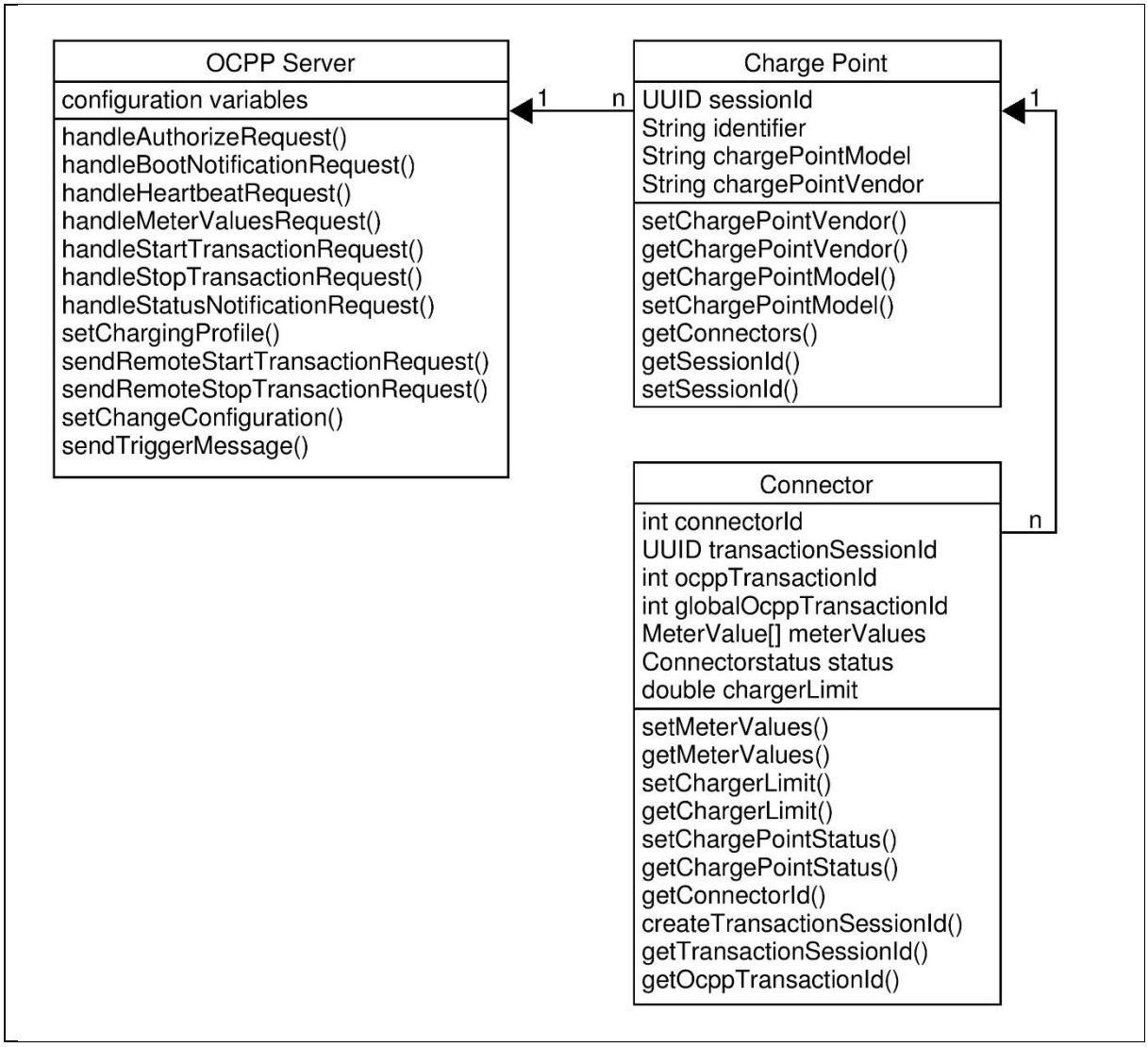


Figure 3-24 Java Hierarchy

3.3.4.4 Current reader

The java program runs a thread that responds to RS232 communication on the GPIO pins on the raspberry pi. This RS232 communication is from the shield described in 3.2.3.1. The shield returns 3 10 bit values that represents the current in the main wires. The current in the main wires is calculated based on coil turns of the CT-sensor, burden resistor and the supply voltage to the shield.

3.3.4.5 Control loop

The java program has a control loop running once every second, the loop performs the control algorithm described in 3.3.3. The control is based on the information stored in the Connector objects described in 3.3.4.3 and the reader data described in 3.3.4.4. When a loop is completed the OCPP server will send new charging profiles with limitations to the chargers where this is relevant, the charging profiles are described in section 3.3.2.9.

The beta version of the software used in the Schneider Charge Point used for testing the charge profile was not implemented. Instead a non-OCPP change configuration message was used to set limits on the Charge Point.

3.3.4.6 Charge Point Simulator

As a separate runnable part of the java software a Charge Point simulator was implemented. This will not run on the Local Controller, but used for testing only. This function can simulate several charge points and responds to load, changing commands from the Local Controller.

3.3.4.7 Web Security

WSS encrypted communication is used in communication between the Local Controller and the Grønn Kontakt sentral server.

3.3.5 Installation procedure

The Local Controller is planned to be easy to install. The following steps will be needed to install a Controller at each location.

1. Install the Local Controller in the customer fuse box.
2. Install CT Sensor clamps in fuse box and connect them to the Local Controller.
3. Install charge points and wire up connections.
4. Power on Charge Point(s).
5. Configure each Charge Point with:
   IP
   Subnet mask
   Enable OCPP 1.6
   Supervision URL: ws://x.x.x.x:8080/ocpp (Local Controller)
   Station ID (CPxxxx)
6. Power on Local Controller.
7. Wait for Grønn Kontakt setup wireless SSID to show up in Wireless Network on laptop, tablet or smartphone and connect to this.
8. Open browser and go to setup.gronnkontakt.no
9. Verify ampere input from CT Sensor clamps.
10. Verify connected chargers on web page.
11. Choose/enter SSID and password on customer's wireless network and click "submit."
12. Configure fuse size, network type, headroom and how each Charge Point is connected.
13. Verify that the Local Controller has communication with Grønn Kontakt.
14. Click "submit."

3.4 Validation and testing

The Local Controller was tested in three different ways: First the simulator described in section 3.3.4.6 was used. When the basic functionality was working, the Local Controller was tested against a Schneider Charger with new software under development in France. Lastly, The Local Controller was tested with a real Charge Point.

3.4.1.1 Simulated test for memory leach and CPU load

The java program creates many threads and objects, and memory leak could be a potential problem. To test if memory leak was happening, 1000 simulated Charge Points was started and each had two active connectors with an active transaction. The program memory was steady at 500-540 megabyte, and CPU load was low. Note that the memory usage is quite high, but 1000 chargers with two active connectors is an unrealistic high number.

The simulated Charge Point also has a function to randomly start and stop transactions. In this test 10 Charge Points were simulated, all with 2 connectors that could be utilized. In this test the memory was steady at 90-100 megabyte and the CPU load was inconsiderably low. The total simulated load was also monitored. The test verified that the controlling algorithm functions with multiple Charge Points, and can control the Charge Points below a predefined limit.

3.4.2 Test with Schneider 22.02.2017

Schneider was working on updating Charge Points to support OCPP 1.6, and we tested several commands against their Charge Point. After some initial problems with mismatching date formats and some other minor differentness, it was confirmed that boot notification, authorize, start transaction and stop transaction were working.

In this test the java software of the Local Controller was running on a PC instead of a Raspberry Pi. The Schneider Charge Point was located in France, and all OCPP communication was through the internet.

3.4.3 Test with Schneider 08.03.2017.

After further improvements, the Local Controller was tested again, against the Schneider Charge Point in France. This test was also performed via internet and all software was running on a PC. The normal sequences of charging were tested, one sequence where charging was locally authorized, and one where charging was started remote. During this test, a fault was discovered with the Stop Transaction message where the program was expecting an object, but received a string from Schneider. There was also a mismatch in the timestamp. Both errors were improved before testing continued.

3.4.3.1 Local authorization scenario:

This scenario simulates a user with local authorization like a card or another physical token of identification.

3.4.3.1.1 Boot Notification

Boot notification request from Schneider.

```
[
      2,
      "1",
      "BootNotification",
      {
            "chargeBoxSerialNumber":
"EV.2S22P22RFc16363pRbNubIdproduId6E",
            "chargePointModel": "MONOBLOCK",
            "chargePointSerialNumber": "3N152820234A1S1B7551700014",
            "chargePointVendor": "Schneider Electric",
            "firmwareVersion": "3.1.1.15",
            "iccid": "8945020184541387888",
            "imsi": "238028230388623"
      }
]
```

Boot notification reply from the local controller.

```
[
      3,
      "1",
      {
            "currentTime": "2017-03-08T13:27:44.130Z",
            "interval": 1,
            "status": "Accepted"
      }
]
```

3.4.3.1.2 Heartbeat

Heartbeat from Schneider.

```
[
      2,
      "568",
      "Heartbeat",
      {}
]
```

Heartbeat reply from the local controller.

```
[
      3,
      "568",
      {
            "currentTime": "2017-03-08T13:53:06Z"
      }
]
```

3.4.3.1.3 StatusNotification

Status notification from Schneider.

```
[
      2,
      "678",
      "StatusNotification",
      {
            "connectorId": 1,
            "errorCode": "NoError",
            "status": "Preparing",
            "timestamp": "2017-03-08T13:55:01Z"
      }
]
```

Status notification reply from the local controller.

```
[
      3,
      "678",
      {}
]
```

3.4.3.1.4 Authorize

Authorize request from Schneider.

```
[
      2,
      "4",
      "Authorize",
      {
            "idTag": "0700001B065920"
      }
]
```

Authorize reply from the local controller.

```
[
      3,
      "4",
      {
            "idTagInfo": {
                  "expiryDate": "2017-03-08T13:56:10Z",
                  "parentIdTag": "DummyParentIdTag",
                  "status": "Accepted"
            }
      }
]
```

3.4.3.1.5 Start Transaction

Start transaction from Schneider.

```
[
      2,
      "9",
      "StartTransaction",
      {
            "connectorId": 1,
            "idTag": "0700001B065920",
            "meterStart": 0,
            "timestamp": "2017-03-08T13:56:35Z"
      }
]
```

Start transaction reply from the local controller.

```
[
      3,
      "9",
      {
            "idTagInfo": {
                  "status": "Accepted"
            },
            "transactionId": 123
      }
]
```

3.4.3.1.6 Trigger Message (Meter Values)

Trigger message was not implemented by Schneider, as can be seen by the reply. This message is from the local controller to Schneider.

```
[
      2,
      "4591ba68-14ea-4963-8ff5-ded7c704bec6",
      "TriggerMessage",
      {
            "requestedMessage": "MeterValues"
      }
]
```

Reply from Schneider indicating the message is not implemented in the charger.

```
[
      4,
      "4591ba68-14ea-4963-8ff5-ded7c704bec6",
      "NotSupported",
      "Unknown action 'TriggerMessage'",
      {}
]
```

3.4.3.1.7 Meter Values

Meter values from Schneider.

```
[
      2,
      "24",
```

```
"MeterValues",
{
      "connectorId": 1,
      "meterValue": [
            {
                  "sampledValue": [
                        {
                              "value": "0"
                        }
                  ],
                  "timestamp": "2017-03-08T13:59:04Z"
            },
            {
                  "sampledValue": [
                        {
                              "value": "0"
                        }
                  ],
                  "timestamp": "2017-03-08T13:59:24Z"
            },
            {
                  "sampledValue": [
                        {
                              "value": "0"
                        }
                  ],
                  "timestamp": "2017-03-08T13:59:44Z"
            },
            {
                  "sampledValue": [
                        {
                              "value": "0"
                        }
                  ],
                  "timestamp": "2017-03-08T14:00:03Z"
            },
            {
                  "sampledValue": [
                        {
                              "value": "0"
                        }
                  ],
                  "timestamp": "2017-03-08T14:00:23Z"
            },
            {
                  "sampledValue": [
                        {
                              "value": "0"
                        }
                  ],
                  "timestamp": "2017-03-08T14:00:43Z"
            },
```

```
                {
                      "sampledValue": [
                            {
                                  "value": "0"
                            }
                      ],
                      "timestamp": "2017-03-08T14:01:03Z"
                },
                {
                      "sampledValue": [
                            {
                                  "value": "0"
                            }
                      ],
                      "timestamp": "2017-03-08T14:01:23Z"
                },
                {
                      "sampledValue": [
                            {
                                  "value": "0"
                            }
                      ],
                      "timestamp": "2017-03-08T14:01:43Z"
                },
                {
                      "sampledValue": [
                            {
                                  "value": "0"
                            }
                      ],
                      "timestamp": "2017-03-08T14:02:03Z"
                }
          ],
          "transactionId": 123
      }
]
```

Meter values reply from the local controller.

```
[

      3,
      "24",
      {}
]
```

3.4.3.1.8 Stop Transaction

Stop transaction from Schneider.

```
[
      2,
      "26",
      "StopTransaction",
      {
            "idTag": "0700001B065920",
            "meterStop": 0,
            "reason": "Other",
            "timestamp": "2017-03-08T14:03:20Z",
            "transactionId": 123
      }
]
```

Stop transaction reply from the local controller.

```
[
      3,
      "26",
      {}
]
```

3.4.3.1.9 Change Configuration

Change configuration from the local controller.

```
[
      2,
      "6718c9fd-dc22-424a-8e77-7ffed9c006cd",
      "ChangeConfiguration",
      {
            "key": "EMsetting",
            "value": "3"
      }
]
```

Change configuration reply from Schneider.

```
[
      3,
      "6718c9fd-dc22-424a-8e77-7ffed9c006cd",
      {
            "status": "Accepted"
      }
]
```

3.4.3.2 Remote authorization scenario:

This scenario simulates a user remote authorization, this is normally a telephone application.

3.4.3.2.1 Remote Start Transaction

Remote start transaction from the local controller. This message contains a charging profile, but Schneider is not using this profile.


```
[
      2,
      "138ffd31-9245-4c63-9185-4036067dffb9",
      "RemoteStartTransaction",
      {
            "connectorId": 1,
            "idTag": "dummyToken",
            "chargingProfile": {
                  "chargingProfileId": 2,
                  "transactionId": 123,
                  "stackLevel": 1,
                  "chargingProfilePurpose": "TxProfile",
                  "chargingProfileKind": "Relative",
                  "chargingSchedule": {
                        "duration": 5,
                        "chargingRateUnit": "W",
                        "chargingSchedulePeriod": [
                              {
                                    "startPeriod": 0,
                                    "limit": 1000,
                                    "numberPhases": 3
                              }
                        ]
                  }
            }
      }
]
```


Remote start transaction reply from Schneider.


```
[
      3,
      "138ffd31-9245-4c63-9185-4036067dffb9",
      {
            "status": "Accepted"
      }
]
```

3.4.3.2.2 Start Transaction

Start transaction from Schneider.

```
[
      2,
      "26",
      "StartTransaction",
      {
            "connectorId": 1,
            "idTag": "dummyToken",
            "meterStart": 0,
            "timestamp": "2017-03-08T14:22:27Z"
      }
]
```

Start transaction reply from the local controller.

```
[
      3,
      "26",
      {
            "idTagInfo": {
                  "status": "Accepted"
            },
            "transactionId": 123
      }
]
```

3.4.3.2.3 Remote Stop Transaction

Remote stop transaction from the local controller.

```
[
      2,
      "0a8a0fab-7775-4f72-9f4b-ed5bd5a4bed2",
      "RemoteStopTransaction",
      {
            "transactionId": 123
      }
]
```

Remote stop transaction reply from Schneider.

```
[
      3,
      "0a8a0fab-7775-4f72-9f4b-ed5bd5a4bed2",
      {
            "status": "Accepted"
      }
]
```

3.4.3.2.4 Stop Transaction

Stop transaction from Schneider.

```
[
      2,
      "32",
      "StopTransaction",
      {
            "meterStop": 0,
            "reason": "Remote",
            "timestamp": "2017-03-08T14:23:48Z",
            "transactionId": 123
      }
]
```

Stop transaction reply from the local controller.

```
[
      3,
      "32",
      {}
]
```

3.4.3.3 Test Conclusion

The test uncovered two faults in the programming. When these faults where removed, the above messages were proven to be implemented correctly. The above test was lacking background logic and most messages where manually triggered.

3.4.4 Test at with a real Charge Point 05.04.2017 - 11.04.2017

05.04.2017 Schneider installed a beta version of the new Charge Point firmware that supports OCPP 1.6. There were missing features in this version, but this was handled with some modification to the Local Controller.

3.4.4.1 Charge Point Configuration

To start using a Charge Point with OCPP it must be configured. The Charge Point has static IP 192.168.0.102 as factory settings. To communicate with it over Ethernet, the pc used for configuration has to have a IP between 192.0.168.241 and 192.0.168.249 [25]. When the IP configurations are done correctly, it is possible to log onto the Charge Point using the IP in a browser, see Figure 3-25. The Charge Point was configured for OCPP 1.6 and the IP of the Local Controller 192.168.0.243 was added as a supervision URL.

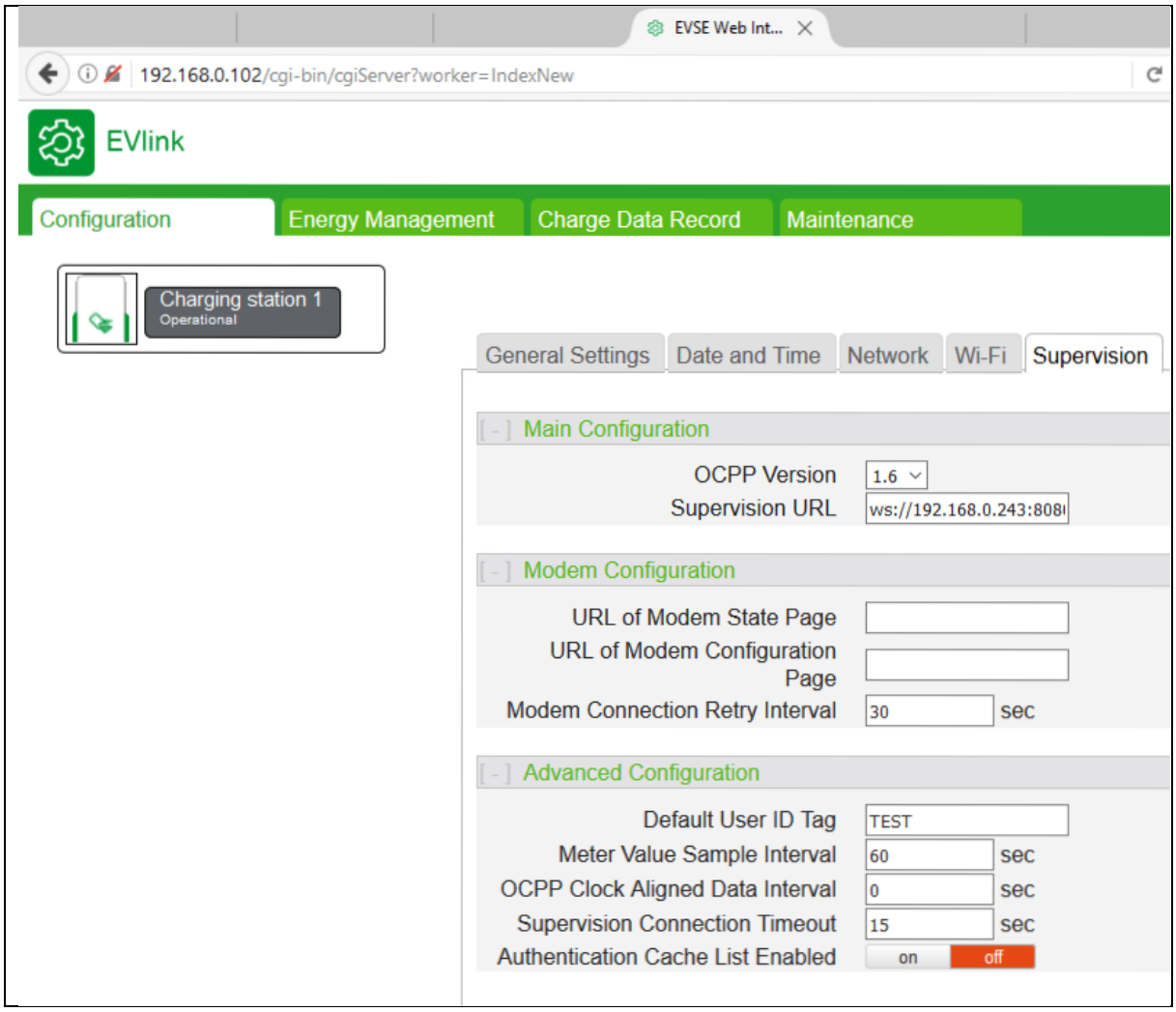


Figure 3-25 Charge Point Configuration

3.4.4.2 Test setup

This test was performed at the Sharebox office on Hisøy. There was only one connected charger as seen in Figure 3-26. The Grønn Kontakt web server was in a test mode, so no economic impact was made from the testing. The facility had TT net and the charger was connected in single phase 230 V, the relevant fuse was 80 A. The facility was in use and it was not possible to obtain controlled test conditions on the main current. This also caused the test to be more realistic, although it is hard to identify the source of loads affecting the measurements.

When analysing data, it was discovered an error with the logging. This error causes the log to report the smallest of the actual current used by the EV and the limit set on the Charge Point.

Figure 3-26 Charge Point with OCPP 1.6 "beta" version

The Local Controller was installed in the fuse box on a DIN Rail and connected the current measurements clamps. Figure 3-27 shows the Raspberry installed and Ethernet cable connected to the charger in Figure 3-26, the blue CT clamps can also be seen in Figure 3-28 connected to the 3 main cables.

Figure 3-27 Raspberry mounted on DIN rail in a fuse box

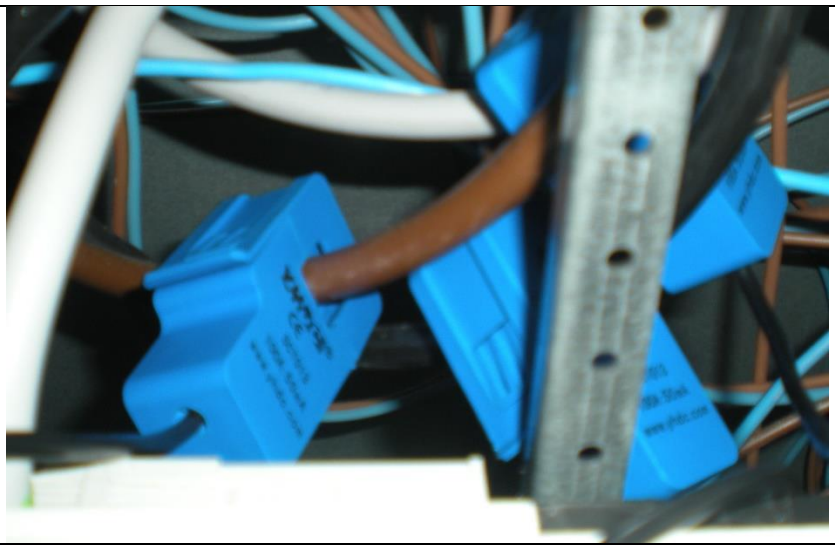


Figure 3-28 Current measurement clamps

3.4.4.3 Test results

Several tests were performed, but only the final will be presented here as the software was under continuous modification. Charging was started with a local chip on the Charge Point, or by a manually triggered Remote Start message from the Grønn Kontakt server. During testing the Grønn Kontakt server was monitored to see if transaction relevant statuses were received correctly.

The presented test was performed on 11.04.2017 from 11:07 to 12:47. As the charging lasted for a long time, interesting sections will be presented as individual graphs. Figure 3-29 explains the lines used in the presented graphs below.

Line	Explanation
Charger Limit (Orange)	The limit of how much current a selected Charge Point can use.
Control Limit (Red)	The total current limitation for the facility, this is based on fuse size or another limitation parameter.
Main Current (Blue)	The largest of the 3 currents measured on the main currents to the house.
EV consumption (Green)	Feedback from the Charge Point on how much current the Electric Vehicle is using.

Figure 3-29 Charging Graphs Explanations

Figure 3-30 presents the whole charging.

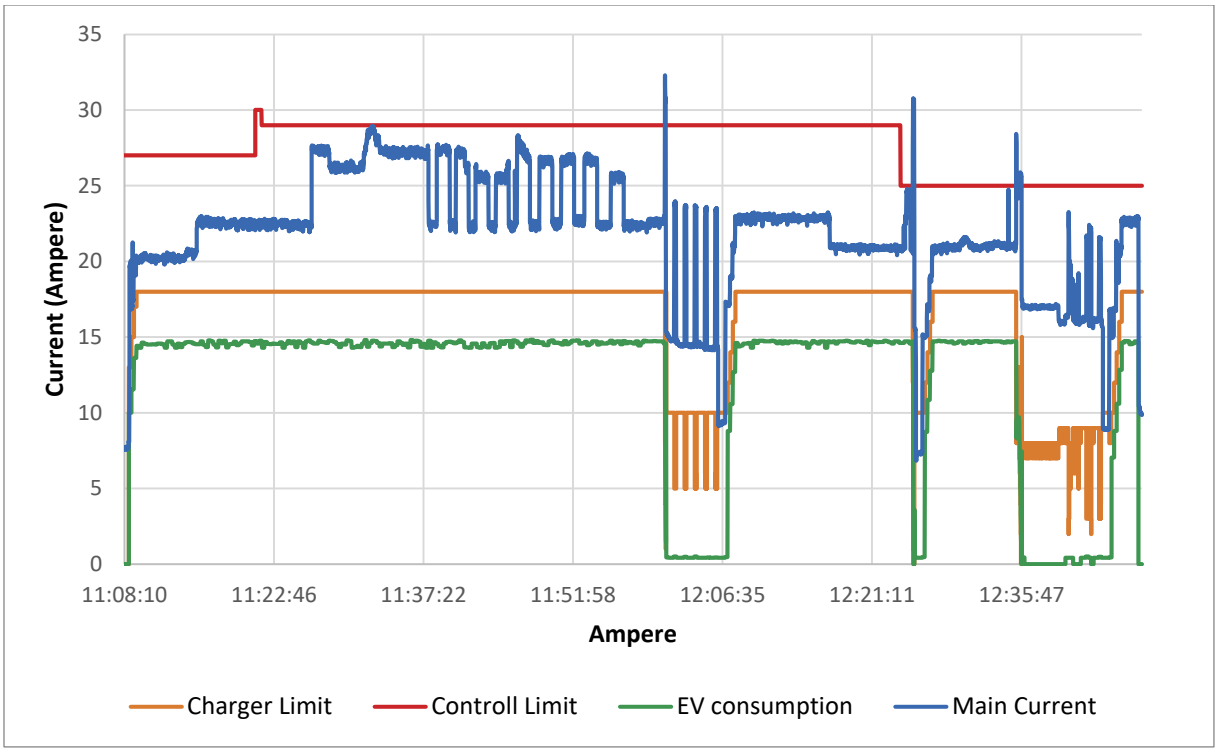


Figure 3-30 Charging Graph Example 1

Figure 3-31 is a close-up of a period where the charger was unaffected by any limits.

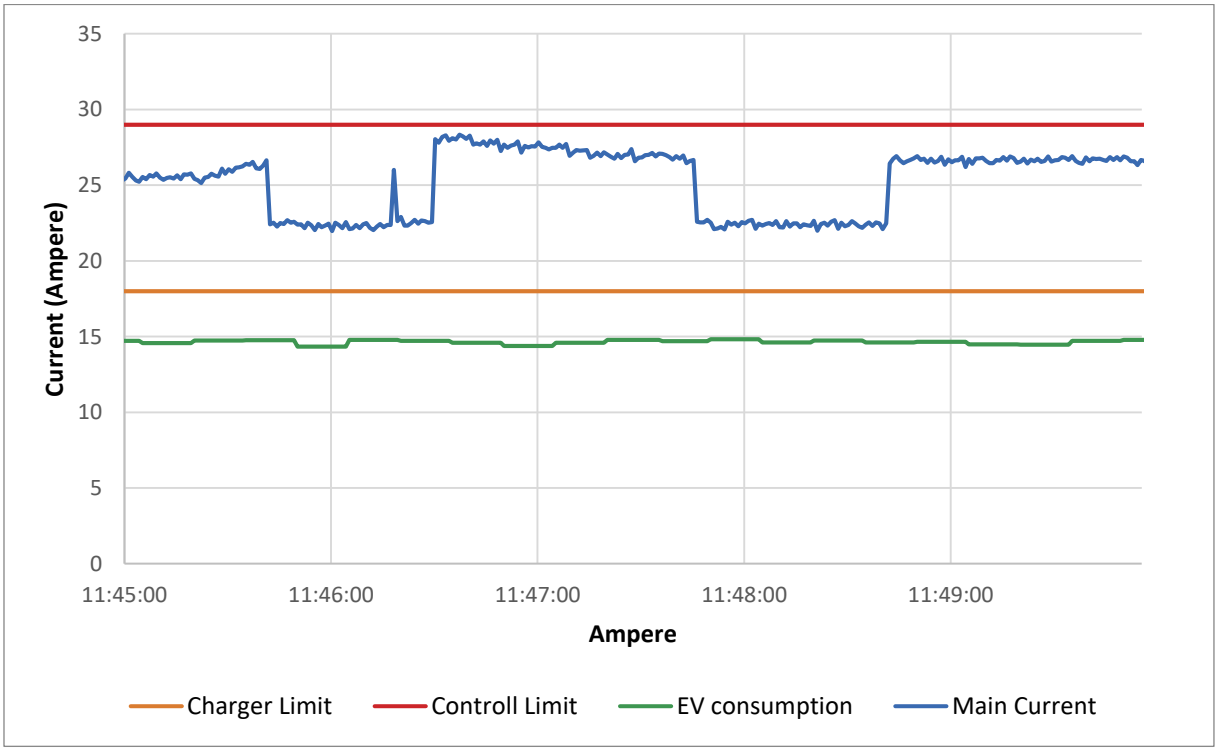


Figure 3-31 Charging Graph Example 2

Figure 3-32 is a close-up of a period where it is believed a water heater with a keep warm function was periodically turning on. This large load stopped EV charging until the water heater was shut off.

When the EV starts charging again it can be seen that the Charge Point internal program runs on a 15 second loop.

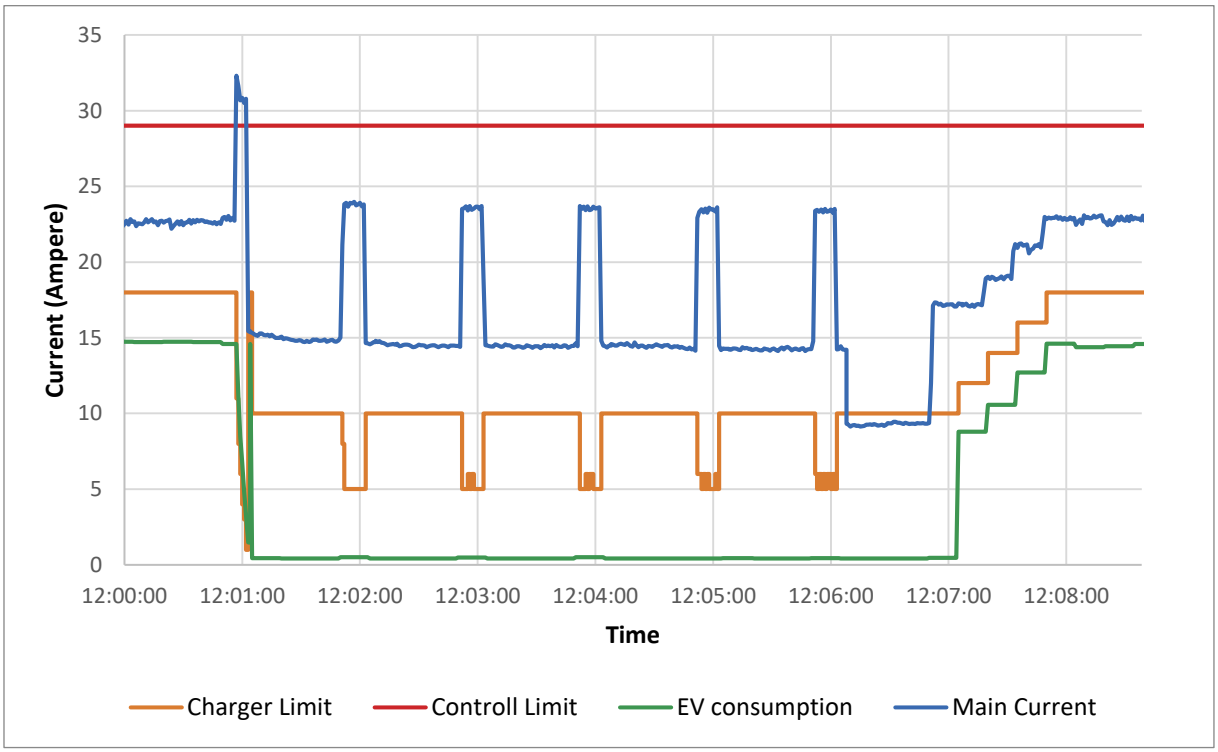


Figure 3-32 Charging Graph Example 3

Figure 3-33 is a close-up of a period where the EV is limited, this graph is affected by a log error where the smallest of the limit and the actual current was reported, the Schneider Charge Point used runs on a 15 second loop, and any changes to the limits will not be considered faster than this.

From 12:35:20 to 12:35:33 it can also be seen that the charger limit is set too high and does not follow the calculation presented in section 3.3.3. The log data is not synchronous and relevant data is missing, so the case needs to be recreated and studied further to conclude on the cause.

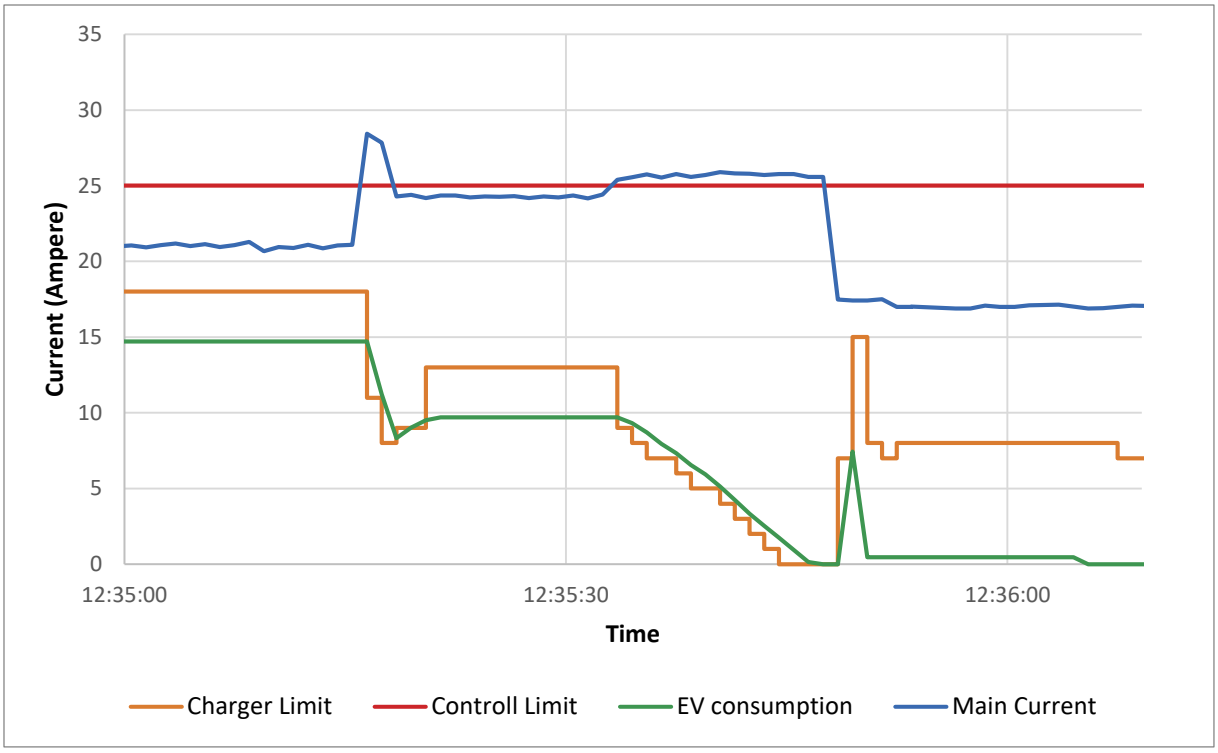


Figure 3-33 Charging Graph Example 4

3.4.4.4 Test Conclusion

The tests prove that all OCPP messages are functioning between the Local Controller and the Charge Point. Relevant messages are also transferred to the Grønn Kontakt server and all transaction statuses were received correctly. The Local Controller can control a single Charge Point based on other loads and keep total current below a predefined demand limit.

Incoming data is asynchronous, and is received every 15 second from the Charge Points and every second from the CT clamps. Further improvements in the algorithms is needed to handle this, and the 15 second response time when setting limits on the Charge Points.

An error in data acquisition prevents good data analysis. Only the lower of the limit set on a Charge Point and the Charge Point meter value is returned. The test is performed over a long period, but the regulation is only active for a fraction of this time, and there are few interesting data points to analyse. Not all the relevant data was logged because of the described error in logging. Better data acquisition and a longer test in different conditions is needed to analyse key parameters such as standard deviation and response time.

4 Discussion

The local controller is functioning, but does not fulfil all the requirements given by Grønn Kontakt. The system should have a 1 second response time to handle rapidly increasing loads. The Local Controller runs a regulation loop once per second and will in most cases respond within one second, but the Charge Point can use up to 15 seconds to respond, and the complete system is therefore far outside the 1 second response time. At the time of writing the Local Controller did not support remote update, but it was planned. CE certification is also missing.

The Local Controller is designed to handle any combination of grid network, TT, TI, and TN, with or without a home transformer. The Controller should also handle multiple Charge Points of different vendors. The performed tests only included a TT network without home transformer, a single Charger and one specific vendor. If this product is to be used commercially, extensive testing is needed with different setups and over extended periods of time to expose flaws. As described in by Alexandre Court [7], OCPP 1.5 was not implemented exactly the same way by different vendors. It is expected that OCPP 1.6 also will have some vendor specific differences, and this needs testing.

The current setup only considers the main fuse as a limiting factor. The software could be expanded to also consider the Charge Point circuit fuse as another limitation.

The Local Controller also expects all single-phase Charge Points to be between the same phases. Perform the controlling algorithm for loads between multiple phases would require a more complex calculation, the regulator logic would also need phase information from both the current measurements and the Charge Points. A single charger was present during testing, so this was not relevant at the time.

5 Conclusion

Grønn kontakt had a practical problem with peak load on Charge Points. The desired solution is a Local Controller that can limit the electricity consumption of Charge Points. The Local Controller should also communicate with Grønn Kontakt to transmit business relevant information so that users can be billed for charging an EV.

The Local Controller is based on a Raspberry Pi that can be installed in a fuse box. The Local Controller has an added shield that measures current in the 3 main phases of the fuse box. The Charge Points are controlled using OCPP 1.6 protocol.

The Local Controller is functional on a prototype level at the end of this Master Thesis and can limit Charge Point consumption based on current measurement in the fuse box. There are some requirements that are not fulfilled, and the total regulation time is too slow to counter rapidly increasing loads.

Further testing and modification of the control algorithms are needed before the Local Controller is ready as a commercial product.

6 References

[1] 'Sharebox - a better way to share your keys'. [Online]. Available: http://www.sharebox.no/. [Accessed: 24-Apr-2017].

[2] 'Grønn Kontakt'. [Online]. Available: https://gronnkontakt.no/. [Accessed: 24-Apr-2017].

[3] 'Registrerte kjøretøy - SSB'. [Online]. Available: https://www.ssb.no/transport-og-reiseliv/statistikker/bilreg/aar/2017-03-28#content. [Accessed: 26-Apr-2017].

[4] 'Forside - VRI'. [Online]. Available: http://www.forskningsradet.no/prognett-vri/Forside/1224529235249. [Accessed: 01-May-2017].

[5] A. G. Azar and R. H. Jacobsen, 'Agent-based charging scheduling of electric vehicles', in *Green Communications (OnlineGreenComm), 2016 IEEE Online Conference on*, 2016, pp. 64–69.

[6] A. S. Garcia, 'The role of EV adaptive charging in facing higher integration levels of wind energy in Denmark', 2015.

[7] A. Court, 'OCPP Compatibility between a Central System and Electric Vehicle Charging Station'. .

[8] '* Tavler | Trello'. [Online]. Available: https://trello.com/. [Accessed: 14-Mar-2017].

[9] 'Scrum Guide | Scrum Guides'. [Online]. Available: http://www.scrumguides.org/scrum-guide.html. [Accessed: 14-Mar-2017].

[10] 'Home ::: NORATEL - When PERFORMANCE matters'. [Online]. Available: http://www.noratel.com/home/. [Accessed: 25-Apr-2017].

[11] 'The solution for Europe: Type 2 charging socket with or without shutter'. Mennekes Solutions.

[12] 'Technical Specifications AC Connector Type 2'. Poenix Contact.

[13] 'IEC 62196'. IEC.

[14] 'IEC 62196 - Wikipedia'. [Online]. Available: https://en.wikipedia.org/wiki/IEC_62196. [Accessed: 25-Apr-2017].

[15] 'EVlink Wallbox User Manual'. Schneider Electric.

[16] 'ZAPTEC - Transform everything'. [Online]. Available: http://www.zaptec.com/. [Accessed: 10-May-2017].

[17] 'Home - Open Charge Alliance'. [Online]. Available: http://www.openchargealliance.org/. [Accessed: 25-Apr-2017].

[18] 'Open Charge Point Protocol 1.6'. Open Charge Alliance.

[19] 'Grid Controller - GCU100 load balancing - Chargestorm'. [Online]. Available: https://chargestorm.se/products/load-balancing-gcu100/?lang=en. [Accessed: 07-Mar-2017].

[20] 'Develco Products - White label products for smart home, smart energy, home security, and assisted living.' [Online]. Available: https://www.develcoproducts.com/. [Accessed: 26-Apr-2017].

[21] 'Raspberry Pi 3 Model B - Raspberry Pi'. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/. [Accessed: 26-Apr-2017].

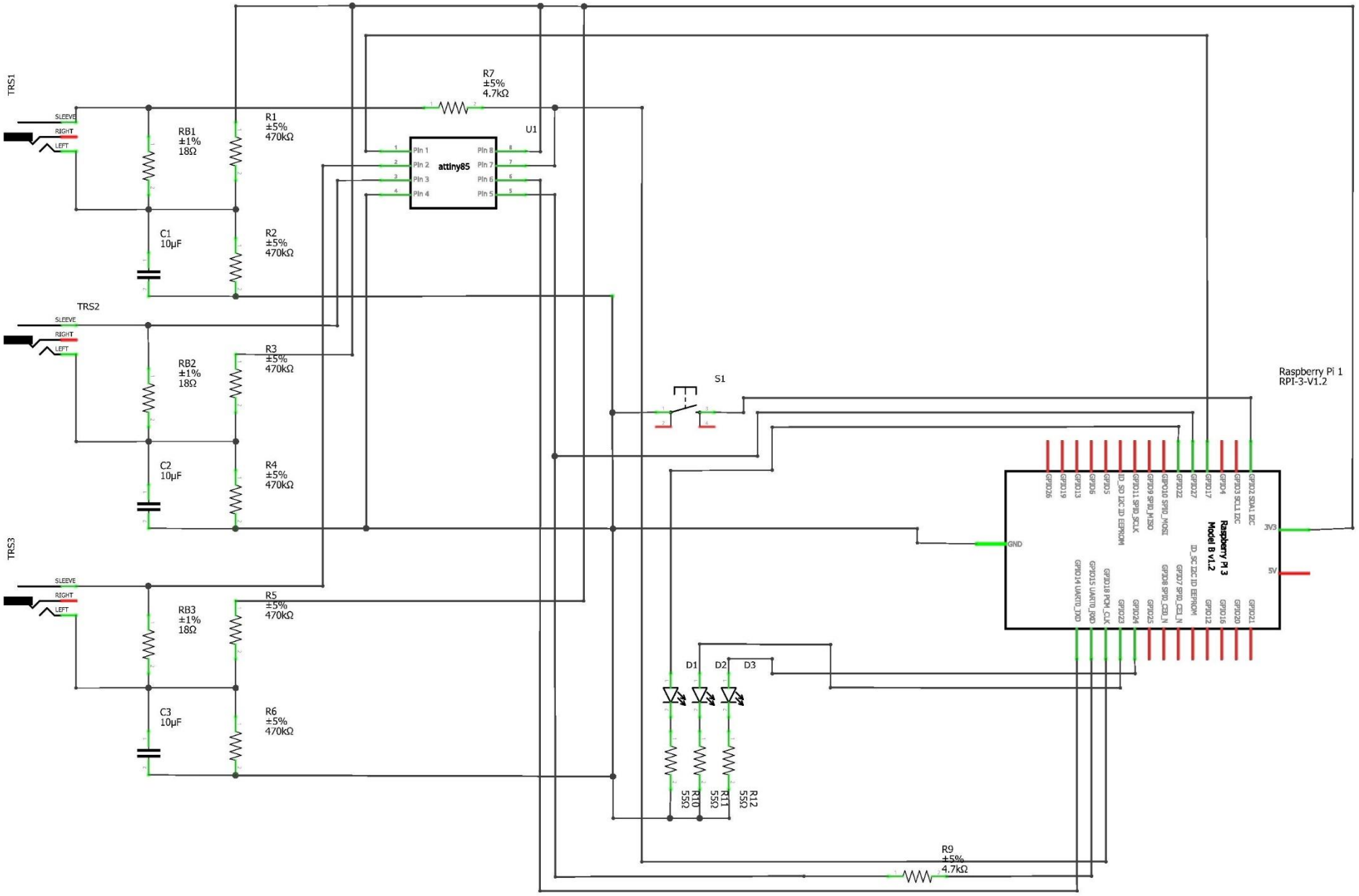
[22] 'GitHub - ChargeTimeEU/Java-OCA-OCPP: Client and server library of Open Charge-Point Protocol from openchargealliance.org'. [Online]. Available: https://github.com/ChargeTimeEU/Java-OCA-OCPP. [Accessed: 10-May-2017].

[23] 'IntelliJ IDEA the Java IDE'. [Online]. Available: https://www.jetbrains.com/idea/. [Accessed: 20-Apr-2017].

[24] 'Maven – Welcome to Apache Maven'. [Online]. Available: http://maven.apache.org/index.html. [Accessed: 26-Jan-2017].

[25] 'DOCA0117NO-00'. Schneider Electric.

Appendix 1 Current Measurement Schematic



Appendix B Code Example

The following code is run as a thread when a Meter Values message is received.

```
@Override
public MeterValuesConfirmation handleMeterValuesRequest(UUID sessionIndex, MeterValuesRequest request) {

    ChargePoint charger = OcppServer.this.chargers.getBySessionId(sessionIndex);

    logger.debug("Handling Meter Values request from charger '{}', Meter Values '{}'.", charger != null ?
charger.getIdentifier() : "???", request.getMeterValue()[0].getSampledValue()[0].getValue());

    MeterValue[] values = request.getMeterValue();

    charger.connectors.getOrCreateByConnectorId(request.getConnectorId()).setMeterValues(values);

    return new MeterValuesConfirmation(); // returning null means unsupported feature
}
```