|  |  |
| --- | --- |
| **HAN EV Charger Technical Reference** | |
|  |  |
| *Supervisor :* |  |
| *Participants :* |  |
| *Author :* |  |
| *Reviewed by :* |  |
| *Date :* |  |
| *Projectnummer :* |  |
| *Documentversion :* |  |

#### Introduction

## Background

From 2013-2018 project CSGrip has been executed by project partners Alfen, TU Delft, Han, Alliander, WES and Bredenoord. In this project an energy container from Alfen is used to create an island grid. This energy container contains inverters and batteries to supply the electrical power for this electricity grid. One specific tasks of the energy container is to try to make a match between the generated electricity and the consumed electricity at any moment. One option for this would be to regulate down renewable energy sources. This energy is then lost (or not generated), which is not preferable. Another option is to use controllable generation, such as a diesel generator set. This is often also a non-preferable option, when the energy container is used in places with low availability (high prices / remote location) of diesel fuel or for sustainability reasons. The last option is to control the consumption. Controllable consumption is only used rarely. In order to test the energy container with controllable consumption a controllable EV charger was created in the CS Grip project. With this charger the charging of an electric car becomes controllable. It becomes possible to charge more in times when a lot of electricity is generated, or less when electricity is scarce.

## About this document

In chapter 2 the basics of electric vehicle charging process is explained. In chapter 3 the used hardware is introduced. Chapter 4 and 5 discuss the software on the Olimex microcontroller and Particle microcontroller respectively. Then in chapter 6 the user interface website is explained. Finally Chapter 7 adds a conclusion.

#### Basics of electric vehicle charging

## Explanation of used standards

EV charging is described in several standards the most important one would be NEN-IEC 61851-1. The currently most recent version is of 2011. In this standard several charging modes are described, which are listed as:

* Mode 1: AC with standardized socket-outlets
* Mode 2: AC with control pilot and RCD as part of the in-cable control box
* Mode 3: AC with control pilot and a dedicated EVSE permanently connected to mains
* Mode 4: Using an off-board charger, with control pilot, permanent mains connection.

For this charger mode 3 is used, in combination with a type 2 (Mennekes) plug. This is often referred to as a type 2, mode 3 charger. The type 2 plug is a 7 pin plug using the 3-phases, neutral, earth as used in every 3-phase connector with two added control pins. The control pins are the control pilot (CP) and proximity pilot (PP).

The CP is a control signal generated by the charger. With this signal the charger communicates a maximum charge current a car is allowed to take. This can be either single phase or 3-phase. The signal on the CP line has a voltage between +12 and -12 Volt. It is a pulse width modulated signal of which the width of the signal represents the allowed maximum current. The car sends information to the charger about its state. The voltage on the line represents the state of the car. The states are defined in the standard and differentiate between no car connected (+12V), car connected (+9V), ready to be charged (without/with ventilation, +6/+3V) and a few error states. The stated voltages are the top voltages of the PWM signal. An example of the signal on the CP line during a charge session is shown in

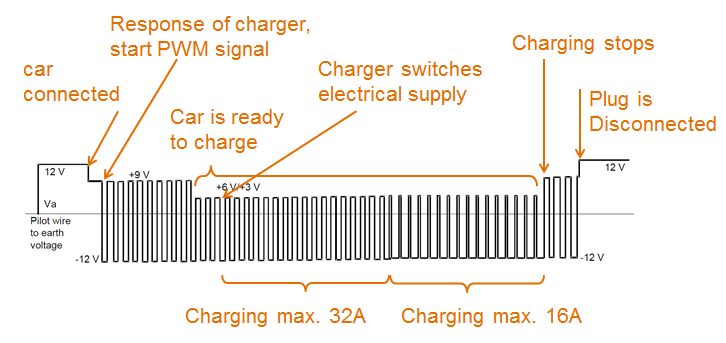


Figure 1: Example of CP signal during charging session.

The PP signal is not connected between the connector of the charge cord and the connector of the charger. It only extends towards to the car and communicates the current capacity of the type of wire that is used. Also when the release/stop button on the plug is pressed, this signal is communicated to the car via the proximity pilot.

#### Hardware of the EV charger

The hardware of the EV charger can be split into a power section and a control section.

## Power Circuit

The power section is very straight forward and contains a circuit breaker (fuse) to protect for overcurrent and short circuit. It contains a residual current device (RCD) type A to protect for AC and pulsed leakage currents. In addition to this, in the future a detection of DC leakage currents is also needed, which is implemented by a separate current measurement device. A relay is used the switch on power towards the car. Also an energy meter is used, which can be read via a Modbus over RS-485 interface. Finally the car connector is the last component in the power circuit, which also has a locking actuator to lock the plug of the car. Separately a small circuit breaker and socket with 24Volt power supply is used to supply the control circuit.

## Control circuit

The heart of the central controller is an Olimex microcontroller board. In addition to this an EV Arduino shield is used. This is a modified version of the Open EVSE shield which support two charger instead of one. The EV shield has two connector for the RGB leds as a basic feedback to the user. Two transistor circuits to switch on the relay’s. And finally two amplifier circuits to create the CP signals from the PWM signal of a digital pin of the Olimex Board. This signal also has a feedback to an analog pin of the Olimex. As an option a DIY residual current device circuit can be added to the board.

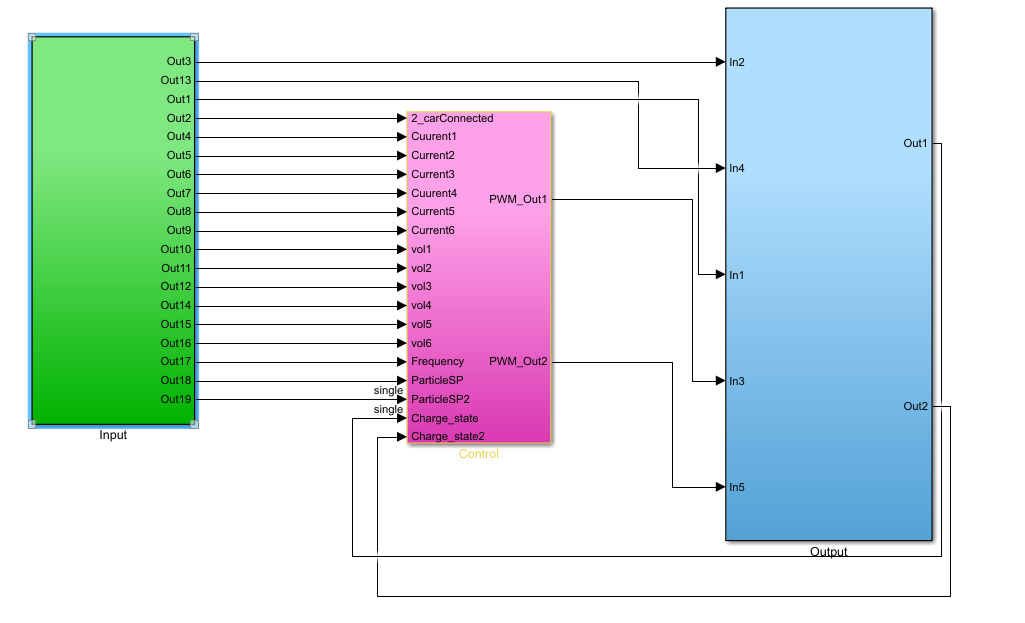
A second shield is the communication shield. This has a second microcontroller, which is a WiFi enabled Particle Photon. The Particle is connected to the Olimex via serial (UART) and has a connection to two RFID readers. It receives measurements on currents, power, etc. from the Olimex and sends these to our website. Also it receives a whitelist from the website of EV users who have permission to charge and it updates charging start/stop events for every user to the website. From there the other chargers are signaled to refresh the whitelist on every change.

The last components are a UART (Serial) to RS-485 converter that is in between the energy meter and the Olimex board. And secondly, a relay board is used that switches the locking actuator.

#### Software of the Olimex EVSE controller

To do

There are 3 main blocks: Input, Control and Output.



Input block has the measures Current, Voltage, Frequency from energy meter (Modbus).

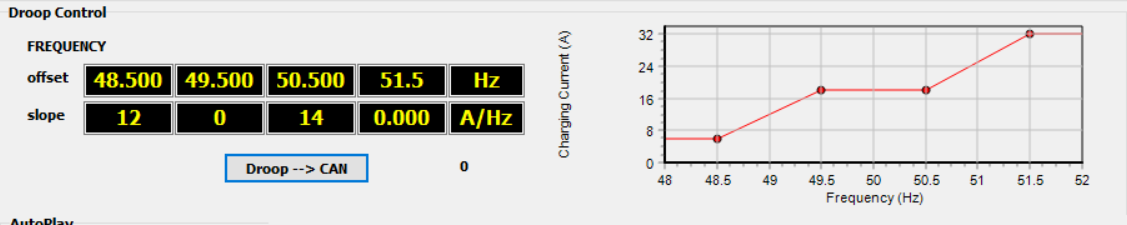
It also waiting for the card Authorization signal from particle cloud.

The control block (magenta) contents: EV control methods, safety protection function and auto detecting function.

There are 5 different control modes: Free Chare, Frequency and Voltage base control, Hantune monitoring and control, Particle or Cloud control. The complete code will be deliver together with the documents.

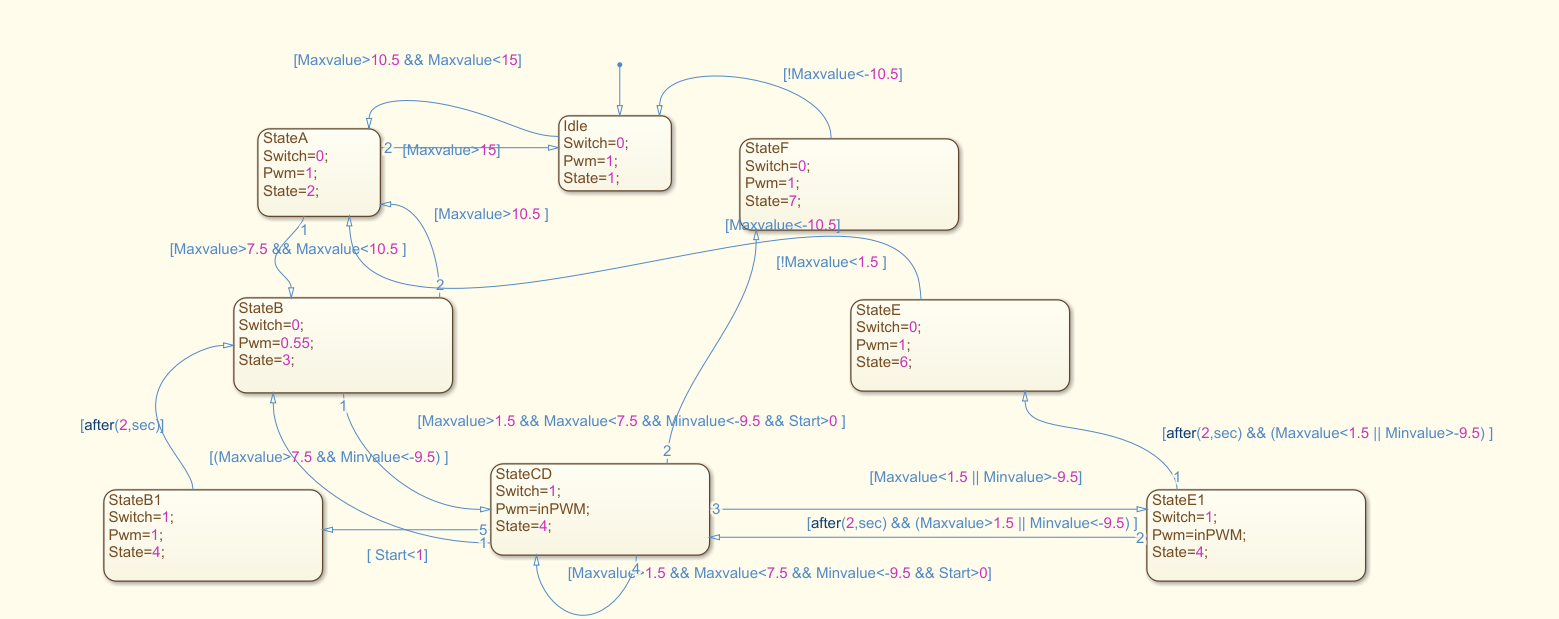
* Free Chare: Let the car charge with maximum power.
* Frequency base control : The power output will increase or decrease based on grid frequency.
* Voltage base control : Similar to frequency control but the output will change based on voltage.
* HanTune monitor and control: Control the power output with HanTune monitoring systems.
* Particle or Cloud control: The control set point will be calculated via Cloud (ex: Solar charging, optimal control…etc) and send to the main controller.

In picture below an example of frequency base control is shows. The plot on the right show how current output react with the frequency change.



The output block has a state machine which following NEN-IEC 61851-1 standard. The max and min value of the pilot signal will define the status of the car.

State A and Idle means no car connected. When the car plug has been connect the systems will move to state B. In state B the car and the chare pole need to communicate, the charge pole need to send the message to the car that it does have enough power to charge (pwm) then up to the car to make a decision that it does want to charge or not. If the car want to charge -> systems go to state CD.



#### Software of the Particle communication shield

As stated in chapter 3 the Particle communication shield has two functions. During the charging process, measurements are send to the website, which appear on the user interface. Also the Particle is connected to the RFID readers which are used for access control for the chargers. In the future, controlling the charger via the user interface, could also be an option. Where the Particle is a men in the middle, between the Olimex and the user interface website.

The Particle Photon runs some software for these tasks. It retrieves measurements over serial port from the Olimex board and it can send setpoints back to the Olimex. Also some administrator functions are programmed; e.g. to put the Olimex into programming mode. A detailed structure of the software is explained in Figure 2.

Figure 2: Overview of software structure on Particle Photon

getUsers () (Particle cloud function)

*//Retreive the authorized EVuserlist from server with GetHttp(), copy line by line to local array: EVUserlist[k]*

maxCurrentC1 () (Particle cloud function)

*//Send a current maximum to the Olimex in a message:*

*// 0xFE,1 (for charger1),setPoint,0xFF*

maxCurrentC2() (Particle cloud function)

*//Send a current maximum to the Olimex in a message:*

*// 0xFE,2 (for charger2),setPoint,0xFF*

resetOlimex () (Particle cloud function)

*//Toggle reset pin of Olimex*

progModeOlmx () (Particle cloud function)

*//Toggle WakeUp and reset pin of Olimex to put it in program mode*

**Particle Cloud functions**

loop() (runs continuously)

* readSerialOlimex();
* if (either socket is used and 30 seconds passed) 🡪Upload latest measurement to server with GetHttp()
* readRFIDCard(1);readRFIDCard(2);
* if(charging did not start after 1 min on charger 1) 🡪 stop charging , update local+online userlist
* idem on charger 2

setup() (runs only once)

* Initialize Serial ports, SPI, RFID library
* Set digital pins to output + HIGH or LOW state
* Fill the EVUserlist with zero’s
* resetOlimex()
* getUsers()
* register all particle functions: getUsers(), maxCurrentC1(), maxCurrentC2(), resetOlimex(), progModeOlmx().

readSerialOlimex ()

Everything is in a loop to make sure all serial charcters are read.

* If(Serial1.available())🡪Serial1.read(); and put in buffer buff.
* stringParse(buff, bufpos);

stringParse ()

* Parse the string send from Olimex, of the form:
* startbyte,function,M1,M2,M3,M4,M5-M12,stopbyte
* startbyte = 0x0F for socket1, 0x0E for socket2
* function: 1=voltage, 2=current, 3=frequency,4=power,5=energy
* M1-M12 are RAW Modbus registers, where M5-12 are optional (used for 3 phase measurements). There are converted from RAW to float using bytesToFloat() and bytesArrToFloatArr()
* Stopbyte is not checked

bytesToFloat ()

bytesArrToFloatArr ()

GetHttp ()

*//‘retrieves’ a website. It is only used to pass variable towards the server during the request*

readRFIDCard ()

* Reads unique ID of RFID card @RFID reader1
* testUser (readID,charger1)
* Reads unique ID of RFID card @RFID reader2
* testUser (readID,charger2)

testUser ()

* Loop over the EVuserlist, Find matching uniqueID
* Check if user is currently not charging
  + Check if someone else is charging here and last check in/out >20 s ago
    - Update EVuserlist local+server (and force other Particles to get new userlist via getUsers()), start charging
* User swiped card at charger where he’s charging and last check in/out >20 s ago
  + - Update EVuserlist local+server (+force other Particles) , stop charge

#### Website

To be added after the new student group has finished the final website.

#### Conclusion

To do

Recommendation : pay attention on dc earth leaked new standard.

BOM list:

Tester box shematics (box with resistor)

User list auth: change from Liandon server to Han server.

ISO standard docs: 61851 part 1 AC charge

DC part 23.

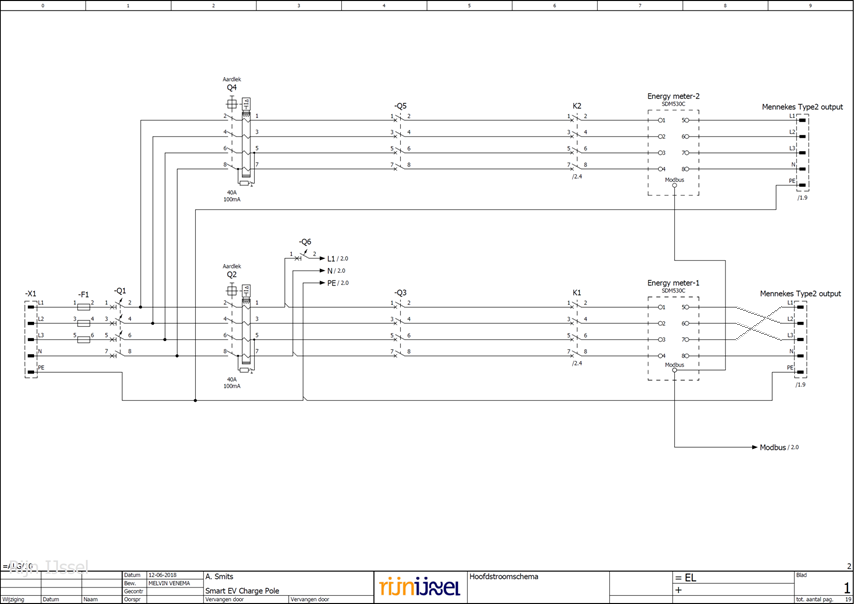
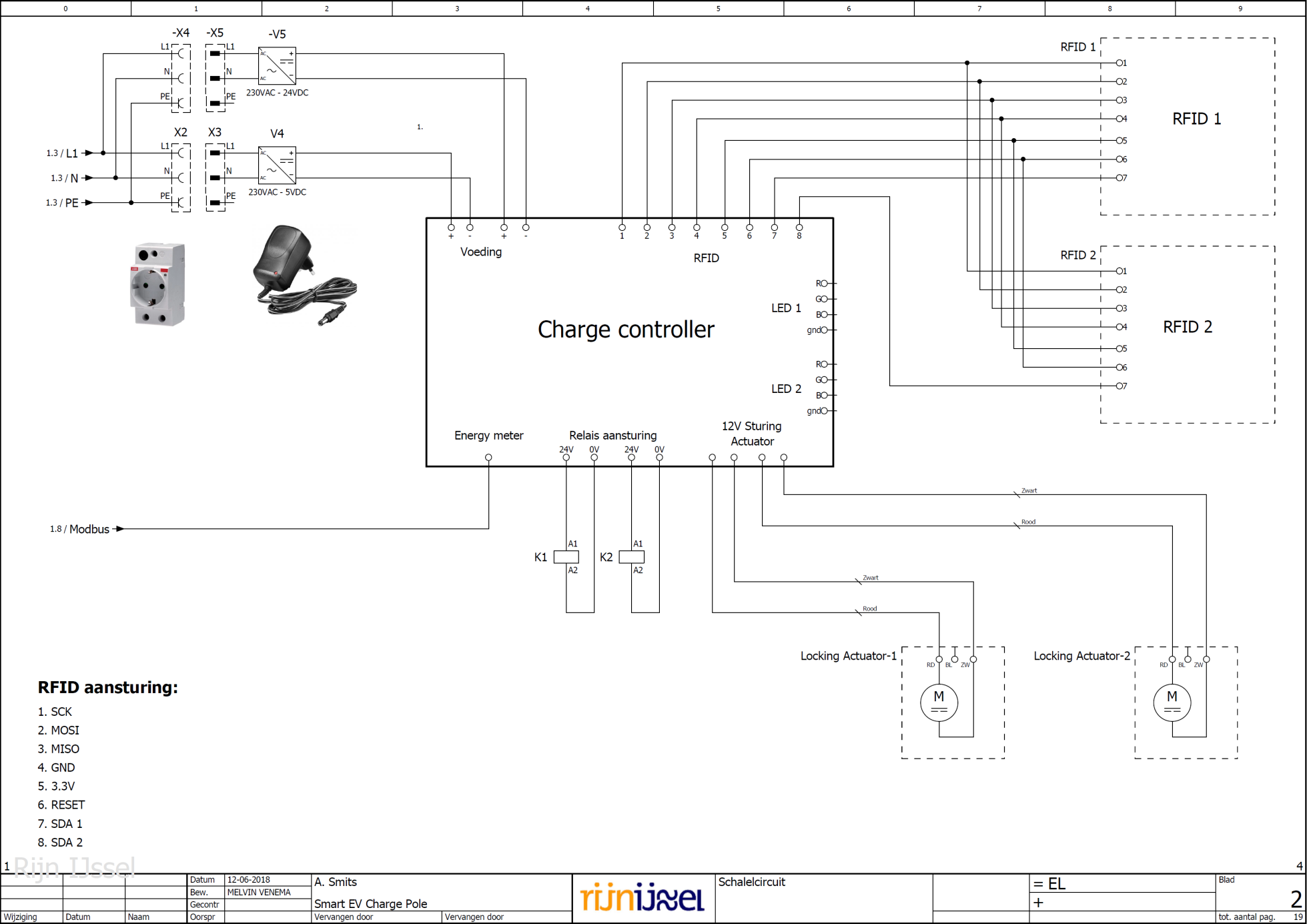
#### Note

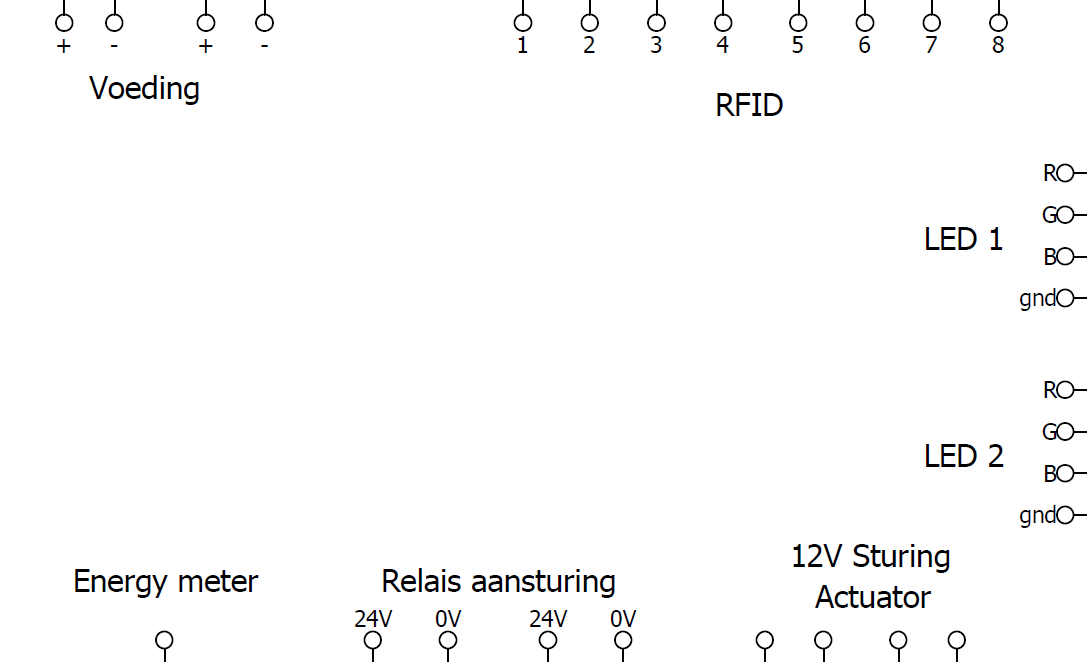
RFID need to place at least 10 cm away from metal case. If not the magnetic energy will transfer to the metal case.

#### Appendix 1. Schematics

List of Schematics:

1. Complete electrical system overview (part 1)
2. Complete electrical system overview (part 2)
3. Charge controller system overview
4. EVshield schematics
5. Locking actuator
6. Particle Photon pinout

****

2

**SDA2**

**SDA1**

**RESET**

**3.3V**

**GND**

**MISO**

**MOSI**

**SCK**

0``

24V

5V

**Particle Photon**

* 3.3V
* GND
* RX
* TX
* D7
* D6
* D5
* A7
* A6
* D2
* D4
* A2
* A1
* A0
* 3.3V
* GND
* A4/MISO
* A5/MOSI
* A3/SCK

**DC/DC converter 24🡪12**

* +
* -

**Wifi to Ethernet bridge**

* Eth

**Olimex**

* RST
* WKUP
* A0
* A1
* A2
* A3
* A4
* RX
* TX
* GND
* 3.3V
* +
* -
* Ethernet
* PD7
* PD8?(TX?)
* PD9?(RX?)
* 5V
* GND
* 3.3V
* GND
* PD6
* PD5
* PD4
* PD3

D8

D10

D12

D13

5V

3.3V

GND

**EVshield**

R LED1

G LED1

B LED1

GND LED1

R LED2

G LED2

B LED2

GND LED2

* 24V +
* 24V -
* A0
* A1
* D8
* D10
* D12
* D13
* 5V
* 3.3V
* GND






* 24V +
* Relay1
* 24V +
* Relay2
* D0
* D1
* D2
* D3
* GND
* Vcc
* NO1
* COM1
* NC1
* NO2
* COM2
* NC2
* NO3
* COM3
* NC3
* NO4
* COM4
* NC4

00

**RS485 to UART**

* GND
* VCC
* RO
* DI
* DE
* RS 485: A,B,GND

**Relay board**

Figure 3: Overview of control system (Charge Controller)

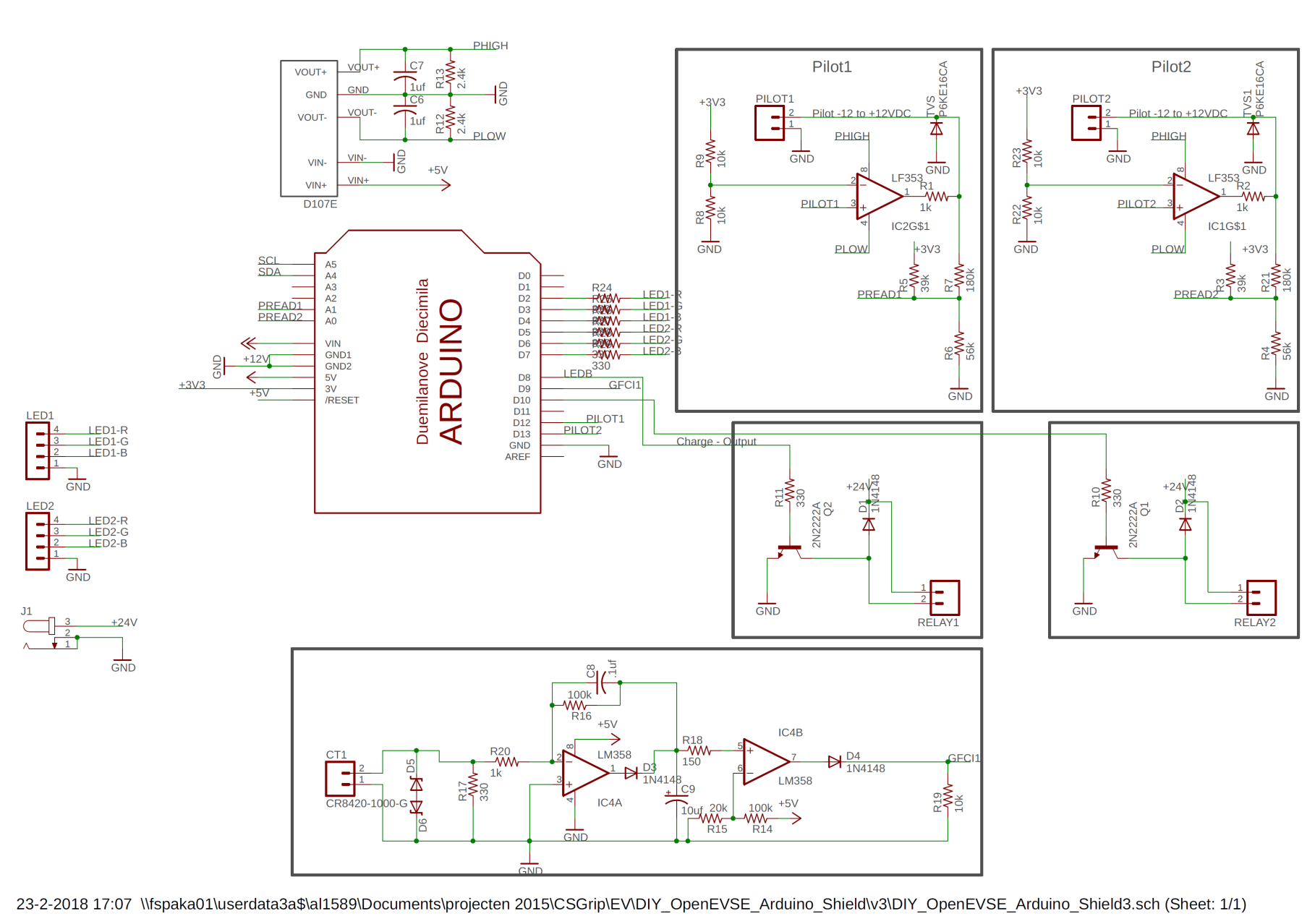


Figure 4: Schematic of EV shield

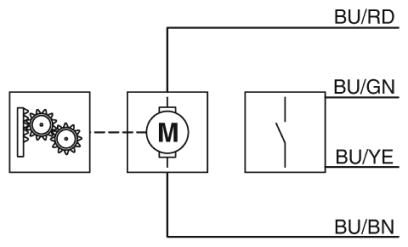
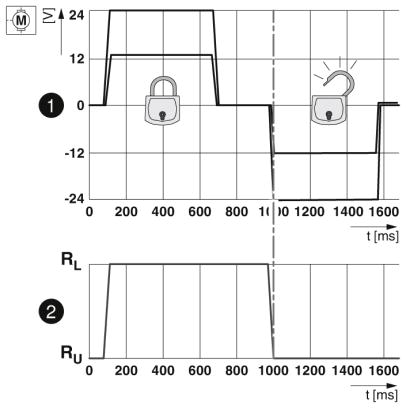
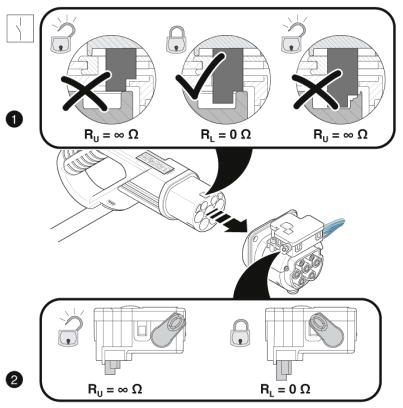
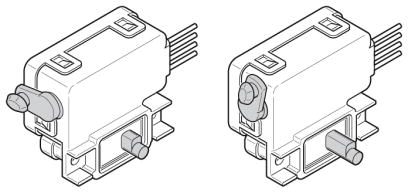
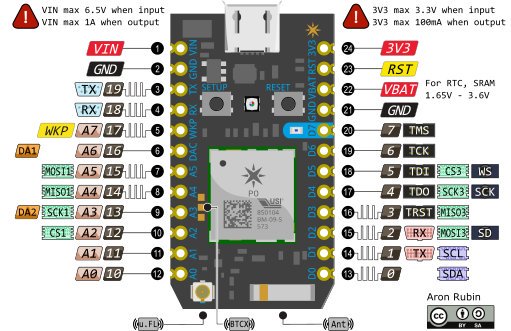


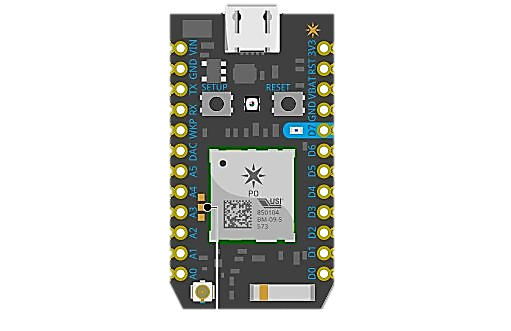
Figure 5: Control and contact information of the locking actuator

Figure 6: Connectors, wiring and pinout of Particle Photon/Particle Photon Shield

**Photon**

**Shield**

* Reset
* 3.3V
* 5V
* GND
* GND
* Vin
* A0
* A1
* A2
* A3
* A4
* A5
* RX
* TX
* D2
* D3
* D4
* D5
* D6
* D7
* D8
* D9
* D10
* D11
* D12
* D13
* GND
* AREF



* WKUP
* D0
* D1
* D3
* SDA2
* SDA1
* RESET
* 3.3V
* GND
* MISO
* MOSI
* SCK

#### Appendix 2. BOM list:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Partlist | | | |  |  |  |
| number | quantity | Part | Description | | | |
|  |  |  |  | | | |
|  |  |  |  | | | |
|  |  |  |  | | | |
|  |  |  |  | | | |
|  |  |  |  | | | |
|  |  |  |  | | | |

#### 

Appendix:

Web calculation:

The system we use is a 3phase system. We have 3 live wires, with 3 voltages and 3 currents. It should be summed in order to get total power. You can ignore the scaling by PF (power factor) in the general formula since it is almost 1.

Power = ampere\_L1\*volt\_L1(\*PF)  +   ampere\_L2\*volt\_L2(\*PF)  +   ampere\_L3\*volt\_L3(\*PF);

Or another possibility

Power = (Energy[now]-Energy[last\_measurement])  /  (now-last\_measurement);

powerKW = power / 1000 (KW) is correct.

kWhNeeded  = AvailableBatteryCapacityToCharge = (100%-UserEnteredSoC) \* Batterycapacity\_kWh;

estimate = kWhNeeded / powerKW (hours) is correct.