# OBDZero User Manual 4.0

OBDZero reads, displays and stores data from the iMiev, CZero and iOn electric cars. The data such as speed and electricity use are available on the car's CAN computer network via a Bluetooth dongle attached to the car's OBD port. OBDZero presents this data in 12 different screens. A 13$^{th}$ screens logs messages between the app, the OBD dongle and the car.  Six screens are intended for use while driving.  These are:

- Wh shows the battery capacity in kWh and the remaining kWh
- Ah shows the battery capacity in Ah and the remaining Ah
- Volts shows the battery volts and the voltages of the highest and lowest cells
- oC shows the average cell temperature and the temperatures of the warmest and coldest cells
- WATTS shows the car's average watts, speed and watt-hours per km.
- DRIVE updates the distance to the next charging station, the difference between the remaining (aka rest) range and the distance to the station, and suggests a speed to the station.

OBDZero saves data in semicolon separated text files, either in the phone's storage or on a SD Card depending on how the phone is set up. The app was developed on an older phone running Android 4.3 with an INTEY OBDII, an inexpensive Bluetooth dongle. It also runs on the more expensive OBDLink LX dongle, though a firmware update of the OBDLink may be necessary. It has been tested on a newer phone running Android 12. The app does not exchange data with the Internet and it does not use GPS. This user manual applies to OBDZero version 4.12 released on July 5, 2024.

### What's new in version 4.12

We have developed as series of functions that permit OBDZero to correctly report the conditions of a iMiev/CZero/ iOn with the Catl NMC93 cells. This was a cooperative effort together with piev and MickeyS70 on the English forum (see https://myimiev.com/threads/main-traction-battery-upgrade-i-miev-using-catl93ah-cells.5458/page-14#post-50617) and Ionysos on the German forum (see https://www.goingelectric.de/forum/viewtopic.php?f=224&t=94422)

Apart from the greater capacity of the NMC93 cell, the primary difference between the standard LEV50 and the NMC93 is a different open circuit voltage (OCV) to state of charge (SoC) curve. Because of this difference the iMiev battery management unit (BMU) cannot measure the actual capacity of the NMC93 cell. OBDZero can. Annex 7 in this manual contains a more detailed explanation.  Information on the practical aspects of the convertion from LEV50 cells to NMC93 cells is available on the forums linked above.

The cell type can be changed between LEV50 and NMC93 in the Initial Values screen.

### Android 11+ phones

Android 10 introduced new rules for storage. In Android 11 following these rules became a requirement for apps on Google Play. Therefore on 11+ phones data is stored in the Download/OBDZero folder. This gives one small difficulty. If OBDZero is uninstalled and then reinstalled it cannot read or delete the previously stored initial values. Therefore on 11+ phones initial values are stored in OBDZero00.ini in the Download/OBDZero folder. With subsequent reinstallations OBDZero00.ini will be incremented to OBDZero01.ini and so on. With many reinstallations there will be many .ini files. These files are small so they do not represent a problem for storage capacity. However if you want to clean up please delete all of the .ini files not just the older .ini files. OBDZero will then create a new OBDZero00.ini file with standard values when run the next time.

## Android 12+ phones

Previous Android versions grant Bluetooth permission automatically. However in Android 12+ OBDZero must be granted "Nearby Devices" permission in order to use Bluetooth. The app will request this permission the first time OBDZero is run on a 12+ phone after an install or reinstall of the app.

## Bluetooth dongles

The dongles I have tested show differences in speed and stability. The app collects data in cycles. For example the OBDLink LX cycle time can be less than 1 sec. For the Vgate Scan the cycle takes between 1.5 and 2.5 sec.   For the INTEY the cycle takes almost 4 sec.

As a rule more expensive dongles work better than less expensive models. For example the inexpensive INTEY runs for some hours before losing contact with the phone. On occasion the INTEY takes more than a minute to connect to the phone and often must be restated before it works properly. Also some dongles show a curious problem in which the cycle time increases during the first 10 min. of operation after which it suddenly returns to normal and stays that way for the rest of the session. I also tested a very inexpensive dongle, a blue ELM 327 mini. It did not work at all.

The Vgate Company sent a number of their dongles for testing and the results are positive. In particular I can recommend the iCar Pro BLE4.0 Dual and the vLink MC. The iCar Pro has a cycle time of 3 to 4 secs while the vLink MC has a cycle time of 1 to 2 sec which is similar to the more expensive OBDLink LX. Both these dongles are quick to pair and link with the telephone and the Bluetooth connection is stable. There are pirate copies of Vgate dongles for sale on the internet. In fact I have used a pirate copy of the Vgate Scan dongle. Tests of the copy and a true Vgate Scan showed that the true Scan is more stable and faster than the copy. When buying a dongle that purports to be manufactured by Vgate, check that Vgate is the supplier.

If there are problems with connection stability, it will be shown in the Info screen as frequent messages such as "Data collector stepped" or "Data collector restarted". At the same time, cycle times can be as long as 30 sec. If WiFi is turned on it may help to turn it off. If this isn't the problem you may need a better dongle.

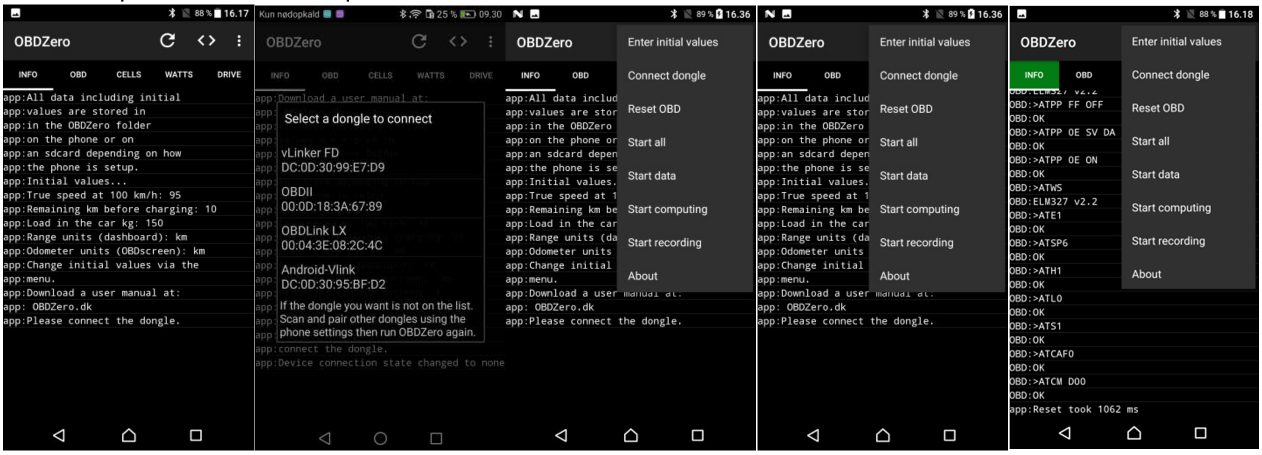
**GitHub**

OBDZero's updated code is available on GitHub via this link:

**https://github.com/DavidCecil50/OBDZero/tree/Android11**

## Startup Instructions

- Plug in the dongle into the OBD connector. It is under the dashboard a little to the left of the steering wheel on left hand drive cars.
- Turn on the car.
- Pair the phone and the dongle using the Bluetooth menu in the phone's setting
- Start OBDZero and stay with the INFO screen
- Open the menu (usually the menu button is three dots in the upper right hand corner)
- Connect the device (dongle). This may require more than one try.
- Once connected open the menu again and reset the dongle.
- Wait for the reset to complete (about 2 sec)
- Open the menu and press one of the Start buttons



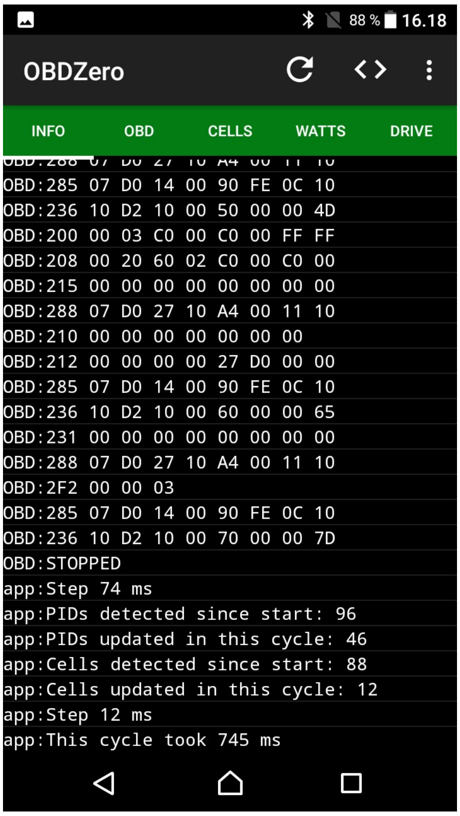

The 5 buttons just below the title are used to move between screens. They change color from black to green not as they are pressed but as the apps functions come online. The buttons and functions are:
INFO: the dongle and the telephone are connected via Bluetooth
Wh: the telephone is receiving data from the car and presenting raw information such as speed.
Ah: the telephone has received data from all the cells of the battery
Watts: the telephone is computing the numbers shown on the watts and drive screens
Drive: the telephone is saving data in semicolon separated text files.

When all the functions are online the INFO screen will look like the screen shown in the next section.

By pressing the <> button in the upper right hand corner of the display, a second and then a third set of screens become available.

If the app has been used recently with the same dongle pressing the restart button will link to that dongle, reset the dongle and start all app functions. It is in the toolbar next to the <> button.

## The INFO screen



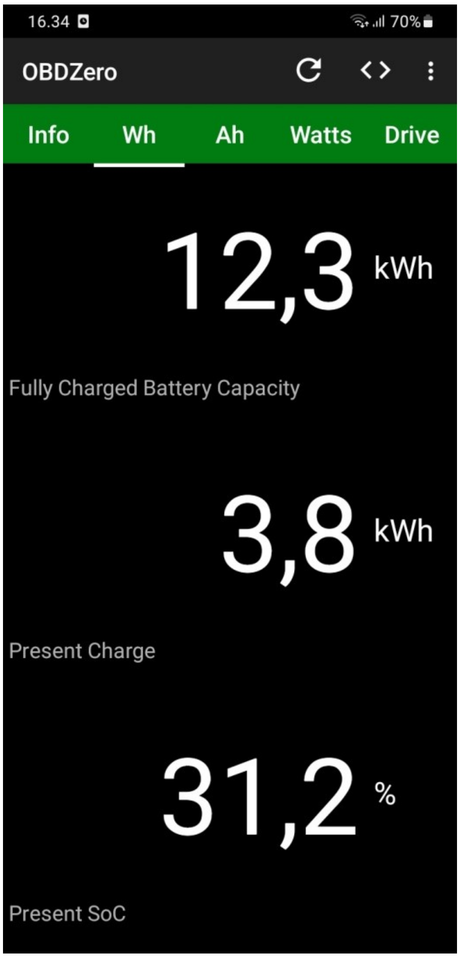

The INFO screen shows messages between the app and the OBD dongle.

Lines that begin with “OBD:” are either commands sent to the dongle or data from the car passed through the dongle. OBD: lines containing “AT” followed by other letters and numbers are commands originally sent from the app and then echoed by the dongle. These commands are found in ELM327DSH.pdf from www.elmelectronics.com.  OBD: lines followed by a three character code and up to 8 two character codes contain the raw data from the car.

Lines beginning with “app:” contain information from the app such as the number of milliseconds required for each step of data collection. The number of unique lines of data collected in each collection cycle, 45 in the example shown, the number of unique cell readings collected in a cycle, in this example 16 and the time required for the cycle, in this example 1352 ms. The dongle in use in this case was the Vgate Scan. Note that not all readings are updated in each cycle. It takes about 5 cycles to update all the readings. However, fast changing values such as speed and amps from the battery are updated in most cycles. Other app: messages are instructions or information to the user. Therefore the INFO screen is the screen to use when starting the app and when diagnosing problems.

When the app is shut down, the first lines during start up and the last lines during shut down shown on the INFO screen are stored in the Info_ file in the OBDZero folder on the phone. This log can be used for troubleshooting communication problems.

**The Wh screen**




The Wh screen shows the capacity and the remaining kWh. The Wh are based on data from the Battery Management Unit (BMU). The Wh screen shows the capacity and the remaining kWh. These numbers are computed based on a SoC computed using the BMU's Ah values (in the next screen) and the nominal values for capacity namely 16 kWh and 50 Ah. In other words:

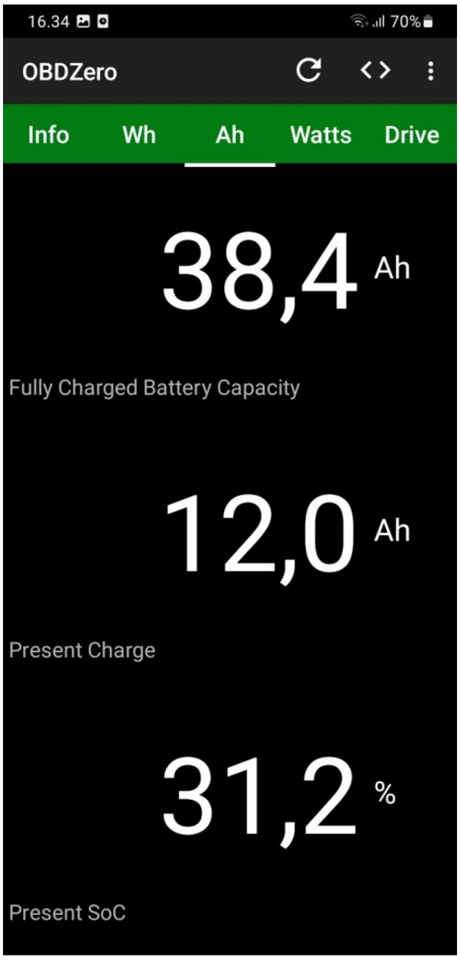
kWh = Ah * 16/50

This is an over simplification but computing a more accurate value based on battery temperature and load characteristics is beyond the present scope of this app.

Based on the capacity and the remaining kWh a more precise SoC is computed and shown on the Wh screen. This value agrees fairly well with the SoC1 value shown on the OBD screen. Note that the SoC is about 10% when the rest range is 0.

To collect the BMU's capacity and the remaining Ah values, OBDZero sets the dongle to a different collection mode then resets the dongle back to normal. This slows the data collection process so this value is only collected at 1 minute intervals.

**The Ah screen**

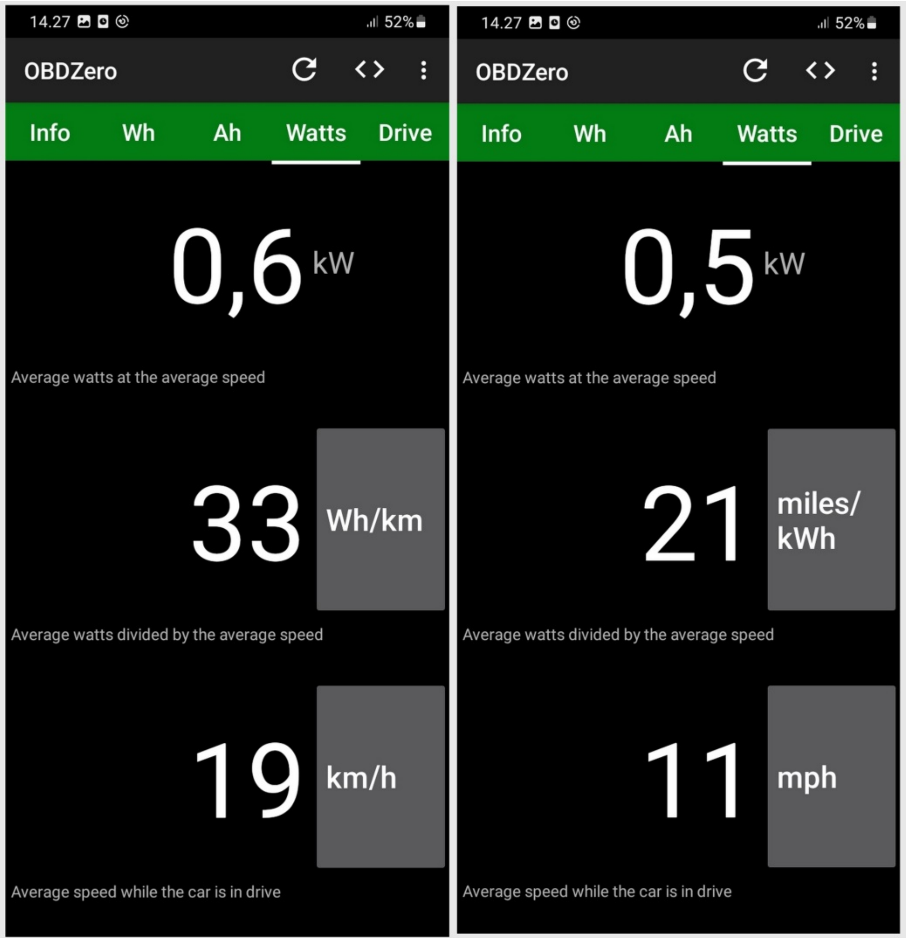



The Ah screens shows data from the Battery Management Unit (BMU). The Ah capacity is the same capacity value as that shown in the CaniOn and the EVBatMon apps. The CAN instructions for the BMU capacity were supplied by anko on https://www.myoutlanderphev.com/forum/viewtopic.php?f=10&t=1796

In the same data string (PID) as the capacity is the car's running account of the Ah in the battery. This is the middle number on the screen. Based on the capacity and the remaining Ah more precise SoC is computed and shown on the Ah screen. This value agrees fairly well with the SoC1 value shown on the OBD screen.

To collect the BMU's capacity and the remaining Ah values, OBDZero sets the dongle to a different collection mode then resets the dongle back to normal. This slows the data collection process so this value is only collected at 1 minute intervals.

### The Watts screen



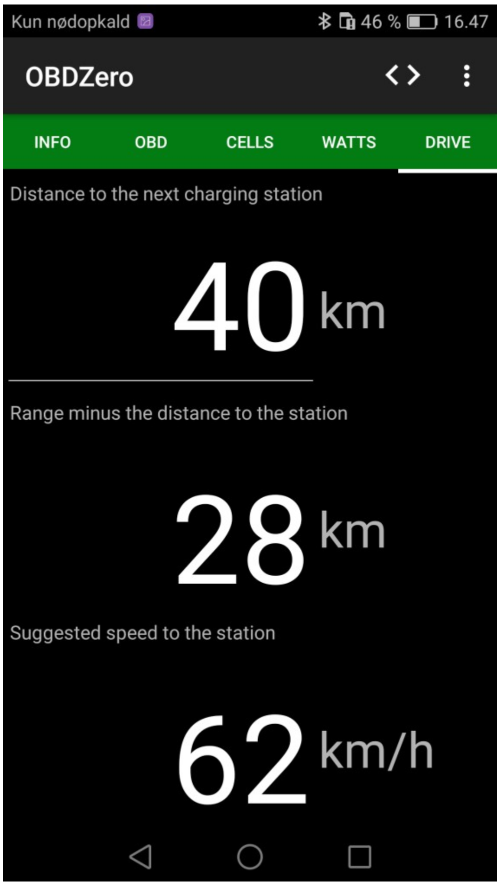

This screen shows average values for the watts from the battery for all purposes, the car's speed and the resulting value for watt-hours per km. By pressing the units buttons Wh/km can be changed to miles/kWh and km/h can be changed to mph. The car's average speed is computed using an exponential filter. This filter is used because it is easy to implement in code. More information on exponential filters can be found on the Internet.

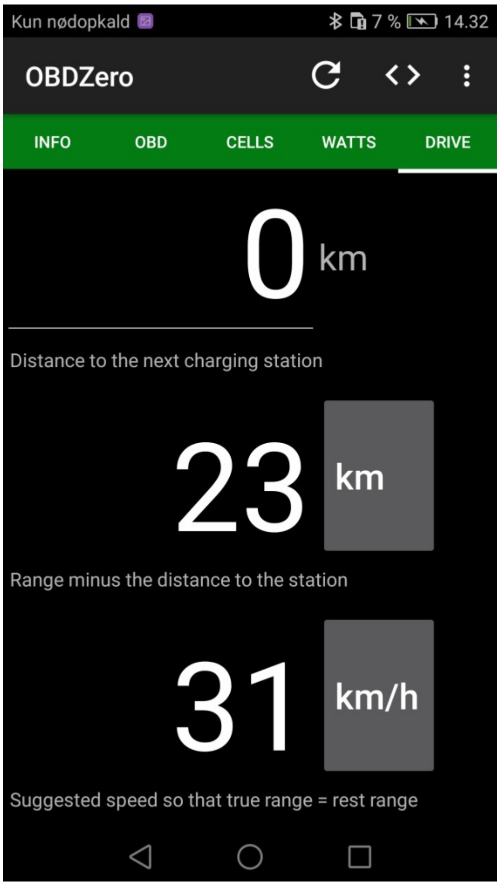
kW are the average measured kW from the battery. As with the speed this average is computed using an exponential filter. To account for changes such as turning the heater on and off the watts for auxiliary functions are subtracted from the measured watts before averaging then added back after averaging. This is the kW shown on the screen. Therefore the kW shown will increase as soon as e.g. the heater is turned on and decrease again as soon as the heater is turned off.

Wh/km is the average watts as shown divided by the average speed. Miles/kwh is the average mph divided by the average kW. Therefore these numbers will also change as e.g. the heater is turned on and off. Furthermore Wh/km will change slightly while city driving as the car accelerates and deaccelerates.

Note that the average speed is updated when the car is in drive. This means that when the car is stopped at lights or by traffic 0 is averaged into the speed and the speed shown will decrease. To avoid unrealistically large values for the Wh/km the average speed doesn't decrease below 1 km/h (0.6 mph).

### The DRIVE screen



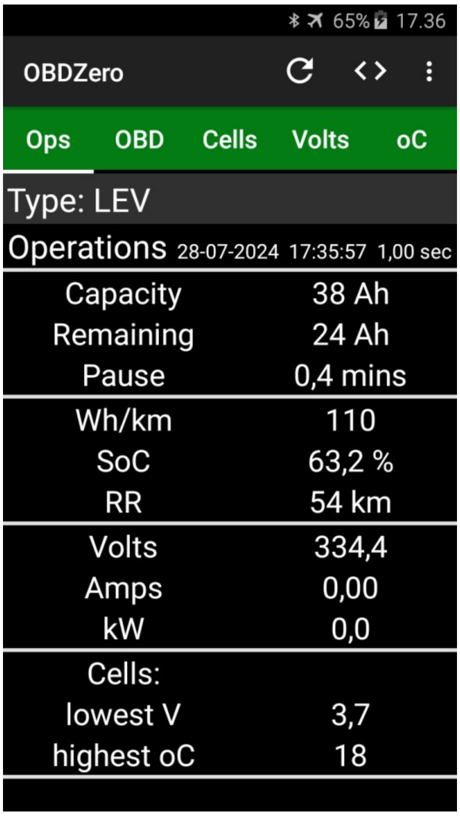

This screen is a trip planning and execution tool. Prior to starting a trip the user enters the expected distance to the next charging station found using e.g. Google Maps. In some cases this may be an out and back distance. The app then computes the difference between the range reported by the car and the distance needed. Hopefully this will be a positive number. While driving, the app computes the distance travelled by multiplying the speed by the time between readings. It then updates the distance to the charging station accordingly. The range minus the distance is also updated.

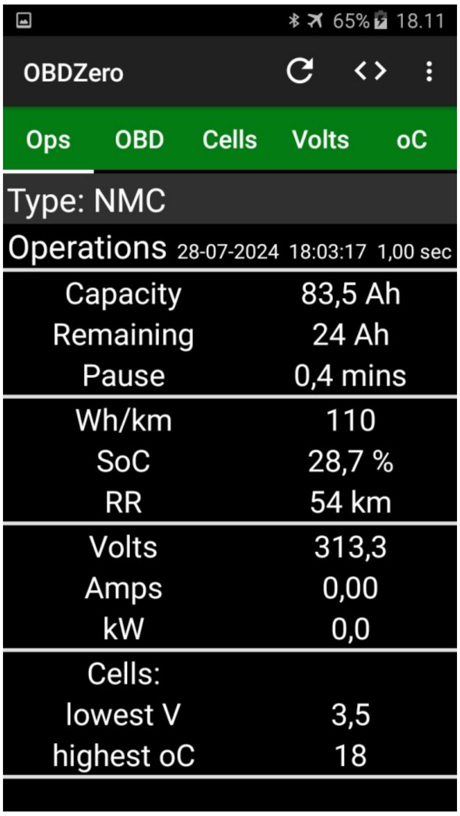
There is no GPS connection so if one deviates from the planned route the distance to the charging station will be in error.

The suggested speed uses a model of the car that computes watts used at any speed. It finds a speed that according to the model will get the car to the charging station. This may be a speed less than the present speed of the car if the battery is running low or it may be a speed greater than the present speed of the car if there are watt-hours to spare. This model is, as with any method of predicting the future, subject to error. To provide some protection against running out of electricity a preferred number of km in reserve when arriving at the charging station can be added. See the Initial Values menu for more on this subject.

As an extra function, when the distance to the next charging station is 0, the speed shown is the speed that will result in the true rest range agreeing with rest range computed by the car. The preferred remaining distance at the charging station is ignored for this computation.

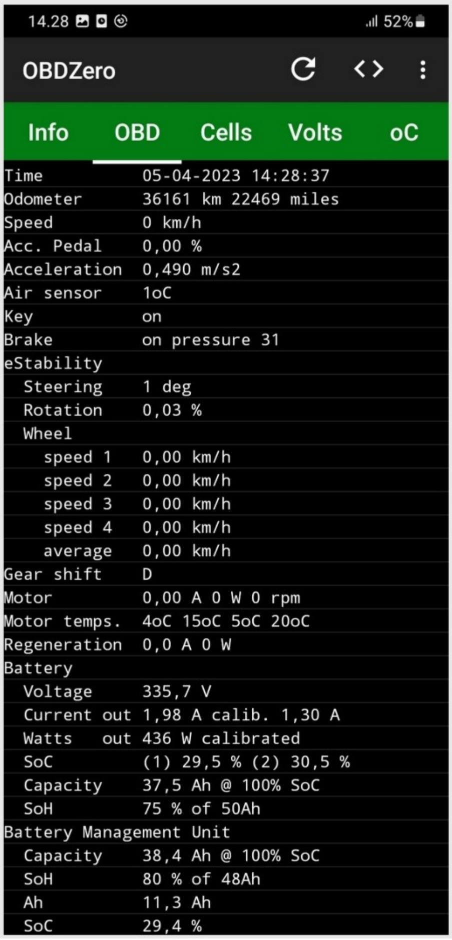
**The Operations Screen**

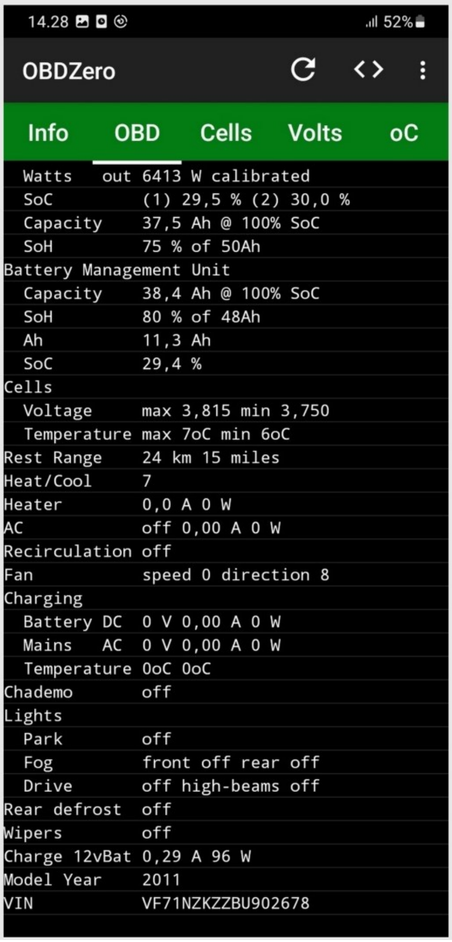



The operations screen provides a key values in one place.

## The OBD screen




By pressing the < > button in the upper right hand corner of the display, a second set of screens becomes available. The INFO screen is the same as before. The OBD screen shows the raw data from the car converted to numbers that are easily understood. The rules for converting this data are not readily available from the car manufacturer. Instead interested owners have spent hours observing the raw data, deducing the rules and publishing them on the Internet. In particular I must thank jjlink, garygid, priusfan, plaes, dax, cristi, SilasAT and kiev all of whom have contributed conversions on http://myimiev.com/forum/. However, there are still many lines of data which we don't know how to interpret yet.

Most of the numbers shown on the OBD screen are self-explanatory. However a few can be confusing. E.g. there are two numbers for battery out amps. The first is the amps computed using the conversion found on the iMiev forum. The second is amps I compute after having calibrated the amps measurement on our car. I believe that this is the more accurate value for all iMiev/CZero/iOns.

There are two values for the State of Charge, SoC1 and SoC2. Both are computed by the car as part of a system for estimating the present fully charged capacity of the battery aka State of Health (SoH). There is some indication that SoC1 is the more accurate of the two but this requires more investigation.

There are also two values for the present fully charged amp hour (Ah) battery capacity. They are labelled Capacity and BMU capa. Capacity is the capacity shown in previous versions of this app. It is computed by the car and coded into a PID. BMU capa is also computed by the car and coded another PID. It is the capacity reported by other apps such as CaniOn and EVBatMon. BMU cap has a resolution of 0.1 Ah while

Capacity has a resolution of 0.5 Ah. This might lead one to believe that BMU is more accurate than Capacity but this remains to be seen.  I would appreciate any feedback regarding the two capacity values.
To collect the BMU capacity OBDZero sets the dongle to a different collection mode, gets the value then resets the dongle back to normal. This slows the data collection process so this value is only collected at 1 minute intervals.

Under each of the two capacities are values for the State of Health of the battery. Under Capacity the SoH is given as a percent of the battery's nominal capacity, 50Ah. The cell used in the battery is the Yuasa LEV50 or LEV50N. 50 Ah is the capacity measured on a new LEV50(N) cell by Yuasa in the lab. Under BMU the SoH is given as a percent of 48Ah.Yuasa measurements also show that the LEV50(N) cell loses about 2 Ah soon after leaving their factory. Therefore 48 Ah is a more realistic value for the new capacity of the battery when the car is ready for delivery. The EVBatMon program computes SoH based on 48 Ah.

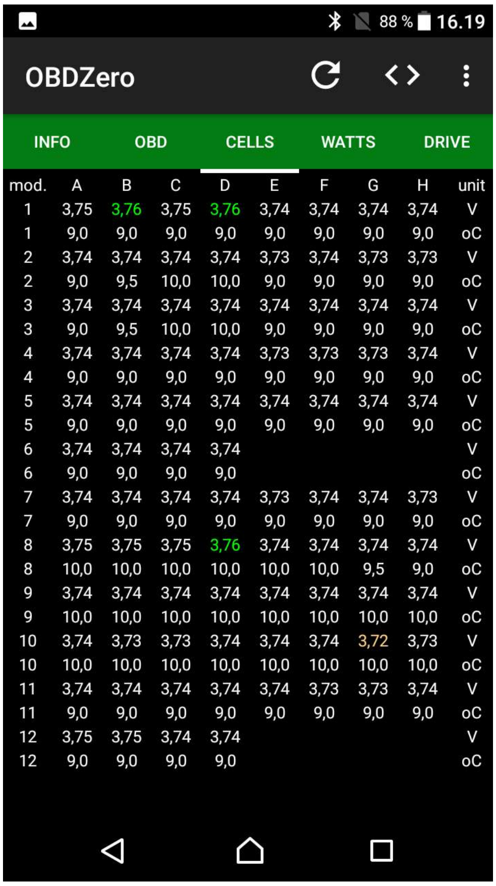
Charge DC is the amps and volts from the car's charging unit to the battery while slow charging. Charge AC is the amps and volts from the mains to the charging unit via the port on the right hand side of the car.

IMPORTANT: If you live in a country where miles are used, the Odometer and Rest Range values shown on the OBD screen may actually be in miles rather than km. This error can be corrected in the Initial Values menu (See the later section on Initial Values). This is important because the app expects values to be in km in its internal calculations. If in miles, many of the calculations will be in error. Once the units are correct the Odometer and the Rest Range will be shown correctly both in km and miles.

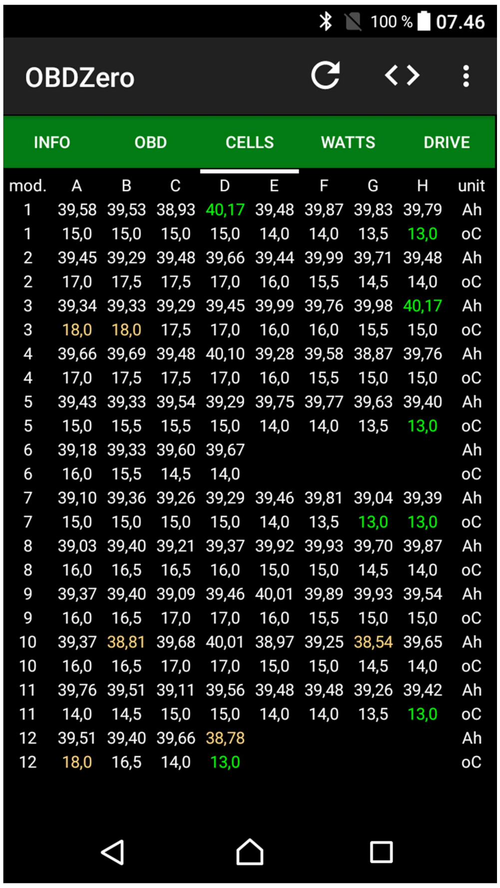
The data shown on the OBD screen is stored in the OBD_ file on the phone.

Annex 2 is a list of OBD data including the rules for converting the raw data on the car's network to the data shown on the OBD screen.

## The Cells screen




| mod. | A | B | C | D | E | F | G | H | unit |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 3,75 | 3,76 | 3,75 | 3,76 | 3,74 | 3,74 | 3,74 | 3,74 | V |
| 1 | 9,0 | 9,0 | 9,0 | 9,0 | 9,0 | 9,0 | 9,0 | 9,0 | oC |
| 2 | 3,74 | 3,74 | 3,74 | 3,74 | 3,73 | 3,74 | 3,73 | 3,73 | V |
| 2 | 9,0 | 9,5 | 10,0 | 10,0 | 9,0 | 9,0 | 9,0 | 9,0 | oC |
| 3 | 3,74 | 3,74 | 3,74 | 3,74 | 3,74 | 3,74 | 3,74 | 3,74 | V |
| 3 | 9,0 | 9,5 | 10,0 | 10,0 | 9,0 | 9,0 | 9,0 | 9,0 | oC |
| 4 | 3,74 | 3,74 | 3,74 | 3,74 | 3,73 | 3,73 | 3,73 | 3,74 | V |
| 4 | 9,0 | 9,0 | 9,0 | 9,0 | 9,0 | 9,0 | 9,0 | 9,0 | oC |
| 5 | 3,74 | 3,74 | 3,74 | 3,74 | 3,74 | 3,74 | 3,74 | 3,74 | V |
| 5 | 9,0 | 9,0 | 9,0 | 9,0 | 9,0 | 9,0 | 9,0 | 9,0 | oC |
| 6 | 3,74 | 3,74 | 3,74 | 3,74 | | | | | V |
| 6 | 9,0 | 9,0 | 9,0 | 9,0 | | | | | oC |
| 7 | 3,74 | 3,74 | 3,74 | 3,74 | 3,73 | 3,74 | 3,74 | 3,73 | V |
| 7 | 9,0 | 9,0 | 9,0 | 9,0 | 9,0 | 9,0 | 9,0 | 9,0 | oC |
| 8 | 3,75 | 3,75 | 3,75 | 3,76 | 3,74 | 3,74 | 3,74 | 3,74 | V |
| 8 | 10,0 | 10,0 | 10,0 | 10,0 | 10,0 | 10,0 | 9,5 | 9,0 | oC |
| 9 | 3,74 | 3,74 | 3,74 | 3,74 | 3,74 | 3,74 | 3,74 | 3,74 | V |
| 9 | 10,0 | 10,0 | 10,0 | 10,0 | 10,0 | 10,0 | 10,0 | 10,0 | oC |
| 10 | 3,74 | 3,73 | 3,73 | 3,74 | 3,74 | 3,74 | 3,72 | 3,73 | V |
| 10 | 10,0 | 10,0 | 10,0 | 10,0 | 10,0 | 10,0 | 10,0 | 10,0 | oC |
| 11 | 3,74 | 3,74 | 3,74 | 3,74 | 3,74 | 3,73 | 3,73 | 3,74 | V |
| 11 | 9,0 | 9,0 | 9,0 | 9,0 | 9,0 | 9,0 | 9,0 | 9,0 | oC |
| 12 | 3,75 | 3,75 | 3,74 | 3,74 | | | | | V |
| 12 | 9,0 | 9,0 | 9,0 | 9,0 | | | | | oC |




| mod. | A | B | C | D | E | F | G | H | unit |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 39,58 | 39,53 | 38,93 | 40,17 | 39,48 | 39,87 | 39,83 | 39,79 | Ah |
| 1 | 15,0 | 15,0 | 15,0 | 15,0 | 14,0 | 14,0 | 13,5 | 13,0 | oC |
| 2 | 39,45 | 39,29 | 39,48 | 39,66 | 39,44 | 39,99 | 39,71 | 39,48 | Ah |
| 2 | 17,0 | 17,5 | 17,5 | 17,0 | 16,0 | 15,5 | 14,5 | 14,0 | oC |
| 3 | 39,34 | 39,33 | 39,29 | 39,45 | 39,99 | 39,76 | 39,98 | 40,17 | Ah |
| 3 | 18,0 | 18,0 | 17,5 | 17,0 | 16,0 | 16,0 | 15,5 | 15,0 | oC |
| 4 | 39,66 | 39,69 | 39,48 | 40,10 | 39,28 | 39,58 | 38,87 | 39,76 | Ah |
| 4 | 17,0 | 17,5 | 17,5 | 17,0 | 16,0 | 15,5 | 15,0 | 15,0 | oC |
| 5 | 39,43 | 39,33 | 39,54 | 39,29 | 39,75 | 39,77 | 39,63 | 39,40 | Ah |
| 5 | 15,0 | 15,5 | 15,5 | 15,0 | 14,0 | 14,0 | 13,5 | 13,0 | oC |
| 6 | 39,18 | 39,33 | 39,60 | 39,67 | | | | | Ah |
| 6 | 16,0 | 15,5 | 14,5 | 14,0 | | | | | oC |
| 7 | 39,10 | 39,36 | 39,26 | 39,29 | 39,46 | 39,81 | 39,04 | 39,39 | Ah |
| 7 | 15,0 | 15,0 | 15,0 | 15,0 | 14,0 | 13,5 | 13,0 | 13,0 | oC |
| 8 | 39,03 | 39,40 | 39,21 | 39,37 | 39,92 | 39,93 | 39,70 | 39,87 | Ah |
| 8 | 16,0 | 16,5 | 16,5 | 16,0 | 15,0 | 15,0 | 14,5 | 14,0 | oC |
| 9 | 39,37 | 39,40 | 39,09 | 39,46 | 40,01 | 39,89 | 39,93 | 39,54 | Ah |
| 9 | 16,0 | 16,5 | 17,0 | 17,0 | 16,0 | 15,5 | 15,0 | 15,0 | oC |
| 10 | 39,37 | 38,81 | 39,68 | 40,01 | 38,97 | 39,25 | 38,54 | 39,65 | Ah |
| 10 | 16,0 | 16,5 | 17,0 | 17,0 | 15,0 | 15,0 | 14,5 | 14,0 | oC |
| 11 | 39,76 | 39,51 | 39,11 | 39,56 | 39,48 | 39,48 | 39,26 | 39,42 | Ah |
| 11 | 14,0 | 14,5 | 15,0 | 15,0 | 14,0 | 14,0 | 13,5 | 13,0 | oC |
| 12 | 39,51 | 39,40 | 39,66 | 38,78 | | | | | Ah |
| 12 | 18,0 | 16,5 | 14,0 | 13,0 | | | | | oC |

For most iMiev/CZero/iOns the battery pack contains 88 cells. There cells are packed together in units of 4 cells. The units are then mounted in modules. 10 modules have 2 units each and two modules, 6 and 12, have one unit each. Cells are referred to by letters from A to H here and in the car's technical manual. The voltage for each cell is measured with a resolution of 0.005 volts however because of the limited width of the screen the voltage is shown with 2 decimals instead of 3.

In each 4 cell unit there are three temperature sensors mounted between the cells. Therefore there are only 3 temperature measurements for 4 cells. The temperatures for the cells shown on the screen are:
Cell A = Sensor 1,
Cell B = (Sensor 1 + Sensor 2)/2,
Cell C = (Sensor 2 + Sensor 3)/2 and
Cell D = Sensor 3
Temperatures for the cells in the second unit in each module are computed in the same way.

In order to make diagnosing of cell problems easier the cells with the highest and lowest values are marked in color such that the highest voltages and lowest temperatures are green and the lowest voltages and highest temperatures are marked with yellow. Red is not used because low voltages and high temperatures are not necessarily indicative of failure. Also to avoid marking insignificant differences the difference between the highest and lowest voltages must be greater than 0.02 volts and the difference between the highest and lowest temperatures must be greater than 1oC before the cells are marked in color.

Some cars built after 2012 have only 80 cells. These cars do not have modules 6 and 12.

Starting with iMievs built in 2016, the manufacture stopped transmitting individual cell data on the car's computer network (CAN). Therefore this data cannot be shown on the Cells screen. According to some owners CZeros and iOns continue to transmit individual cell data for models 2016 and newer.

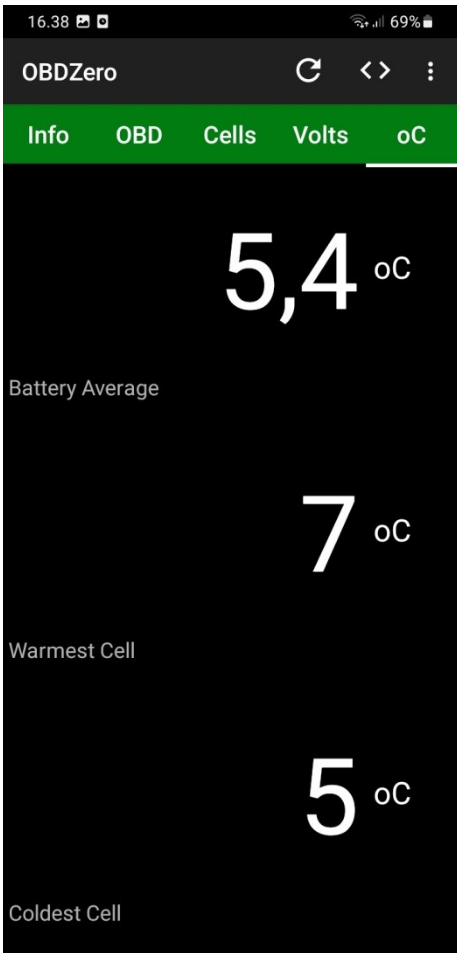
Cell voltages and the computed temperatures are stored in the Cells_ file on the phone. Temperature readings from the sensors are stored in the CellTemperatures_ file.

There are two battery 100% capacity measurements, CAP1 and CAP2. These are described on pages 15 to 18. In both cases the capacities of all the cells are measured. Once a CAP1 or CAP2 measurement has been performed the CELLS screen will now show the capacity of each cell rather than the voltage. This function does not work for iMievs after 2015 where cell data is not transmitted on the CAN network.

### The Volts Screen

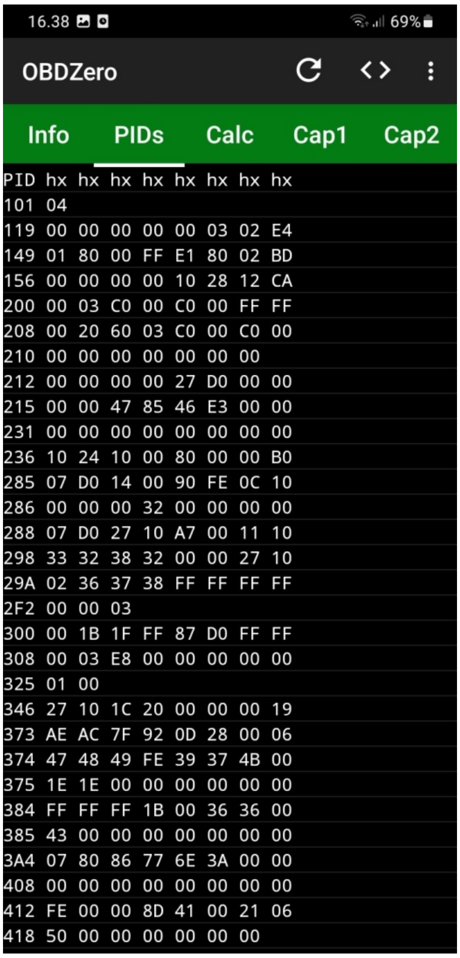

The battery voltage is the same value as shown on the OBD screen. The same is true for the highest and lowest cell voltages. These values are not averaged over time. Therefore they change rapidly with changes in the load on the battery.

**The oC screen**




The temperatures of the warmest and coldest cells are the same values as shown on the OBD screen. These are reported by the CAN network independently of the individual cell data. The battery average is the average of the temperature reported for each cell. As mentioned above, iMievs from 2016 on do not report individual cell data. In which case, the battery average shown on this screen is just the average of the warmest and coldest cells.

**The PIDS screen**




| PID | hx | hx | hx | hx | hx | hx | hx | hx |
|---|---|---|---|---|---|---|---|---|
| 101 | 04 | | | | | | | |
| 119 | 00 | 00 | 00 | 00 | 00 | 03 | 02 | E4 |
| 149 | 01 | 80 | 00 | FF | E1 | 80 | 02 | BD |
| 156 | 00 | 00 | 00 | 00 | 10 | 28 | 12 | CA |
| 200 | 00 | 03 | C0 | 00 | C0 | 00 | FF | FF |
| 208 | 00 | 20 | 60 | 03 | C0 | 00 | C0 | 00 |
| 210 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | |
| 212 | 00 | 00 | 00 | 00 | 27 | D0 | 00 | 00 |
| 215 | 00 | 00 | 47 | 85 | 46 | E3 | 00 | 00 |
| 231 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 236 | 10 | 24 | 10 | 00 | 80 | 00 | 00 | B0 |
| 285 | 07 | D0 | 14 | 00 | 90 | FE | 0C | 10 |
| 286 | 00 | 00 | 00 | 32 | 00 | 00 | 00 | 00 |
| 288 | 07 | D0 | 27 | 10 | A7 | 00 | 11 | 10 |
| 298 | 33 | 32 | 38 | 32 | 00 | 00 | 27 | 10 |
| 29A | 02 | 36 | 37 | 38 | FF | FF | FF | FF |
| 2F2 | 00 | 00 | 03 | | | | | |
| 300 | 00 | 1B | 1F | FF | 87 | D0 | FF | FF |
| 308 | 00 | 03 | E8 | 00 | 00 | 00 | 00 | 00 |
| 325 | 01 | 00 | | | | | | |
| 346 | 27 | 10 | 1C | 20 | 00 | 00 | 00 | 19 |
| 373 | AE | AC | 7F | 92 | 0D | 28 | 00 | 06 |
| 374 | 47 | 48 | 49 | FE | 39 | 37 | 4B | 00 |
| 375 | 1E | 1E | 00 | 00 | 00 | 00 | 00 | 00 |
| 384 | FF | FF | FF | 1B | 00 | 36 | 36 | 00 |
| 385 | 43 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 3A4 | 07 | 80 | 86 | 77 | 6E | 3A | 00 | 00 |
| 408 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 412 | FE | 00 | 00 | 8D | 41 | 00 | 21 | 06 |
| 418 | 50 | 00 | 00 | 00 | 00 | 00 | 00 | |

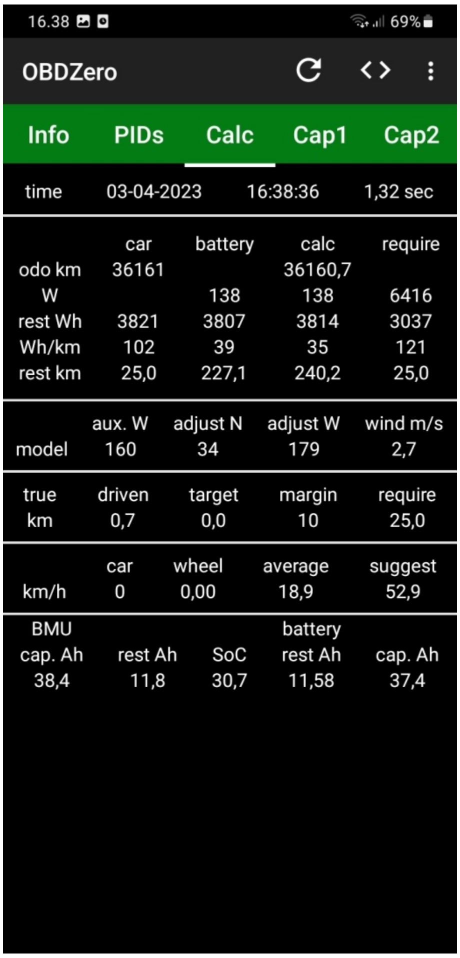
By pressing the  button again, a third set of screens becomes available. The INFO screen is the same as before. The PIDS screen shows all the raw data from the car. Each line starts with a hexadecimal Parameter ID (PID) followed by 1 to 8 bytes of data also in hexadecimal. As mentioned above the meaning of some of this data is as yet unknown. However most of the important data has been decoded. As an example the PID 298 above contains information about the car's motor. PID 298's 7$^{th}$ and 8$^{th}$ bytes encode the motor rpms. 27 hex is 39 decimal and 10 hex is 16 decimal. The rule with these two numbers is:

$$0 = 39*256 + 16 -10000$$

Therefore, the rpms were 0 when this data was recorded. Annex 2 contains a complete list of the conversion rules.

The PIDs are stored in two files on the phone. The PID_ file contains lines in which both the PID and the data are in hex as shown above. In the PIDint_ file the PID is in hex but the data is converted to decimal values.

### The CALC screen




This screen has primarily to do with the model that was mentioned in the description of the Drive screen. Details of the model will have to wait but in short it computes the watts from the battery at any one moment based on:

- Any change in the speed
- Rolling resistance
- Air resistance and
- Auxiliary power

Auxiliary power is used for heating, AC, headlights etc.

Computing these values is fairly straight forward. The challenge is everything OBDZero doesn't know such as the wind direction and speed and whether the car is moving up or down hills. To compensate for this lack of knowledge the model has a 6$^{th}$ input called an error term in modeling theory. Each time the app completes a data collection cycle the model computes the watts it expects the car to use based on the data collected. This is compared with the measured watts and the error term is adjusted. If the measured value is greater than the model value the error term is increased by a small amount and if the reverse is true the error term is reduced by the same small amount. This adjusted error term brings the model watts closer to the measured watts with each cycle.

Using this method the model values are able to follow the measured values quite well. For example in the screen above, there are four values for the remaining watt-hours in the battery (rest Wh). The first value is the rest Wh based on SoC2 and the car's estimate of the rest range. The second rest Wh is based on a sum of the measured watts-hours drawn from the battery. The third rest Wh is based on a sum of the model computed watt-hours drawn from the battery. The numbers aren't exactly the same, it would be surprising if they were, but they are similar and they remain similar over long trips. This indicates that the model is working well. It also indicates that the error term does contain useful information about the factors such as wind speed that we are unable to measure directly. For those of you that are interested, this method of obtaining information is called an “observer” in modeling theory. More information on this subject can be found on the Internet.

Computing the suggested speed shown on the DRIVE screen is the primary reason for the model. Using the model with the error term we can estimate the watts needed to power the car at any speed. We also know the distance to the next charging station so we can estimate the time to that station at any speed. With an estimate of the watts and the time we can compute the watt-hours needed at any speed. Then all we have to do is find the speed for which the watt-hours equal the car’s estimate of the watt-hours remaining in the battery and we know how fast we can drive to the charging station. It’s a bit more complicated than that but the rest is details.

In the CALC screen above, the fourth rest Wh is the model computed watt-hours needed to drive 69.3 km at 73.2 km/h. 69.3 km is the remaining distance to the charging station, 54.3 km, plus 15 km in reserve.

Car rest Wh is based on the SoC 2 (state of charge in %) reported by the car. It is calculated using this equation:

Car rest Wh = (SoC2/100) * cap.Ah * (Volts + Volt0)/2

Where Volts is the present battery voltage and Volts0 is the battery voltage at SoC = 10%. Cap.Ah is the present battery 100% capacity as reported by the car. It is the capacity shown on the OBD screen after the battery voltage. Remember that the last 10% of the battery’s Ah is only available in “turtle” mode. It doesn’t contribute to the rest range shown by the car.

Battery rest Wh is the watt hours based on the sum of the amps used since the OBDZero was started. Battery rest Wh equals Car rest Wh when the app is started. The equation is:

Battery rest Wh = (rest Ah0 – Sum (A * step)) * (Volts +Volts0)/2

Where rest Ah0 is rest Ah based on SoC when the app was started, step is the time step in hours between measurements. Sum (A * step) is the sum of the amp hours used since the app was started.

Model rest Wh is the watt hours remaining based on the model watt hours used. This is a bit more complicated but similar to the Battery rest Wh.

Car Wh/km is the Car rest Wh - 10% of the battery’s capacity in Wh divided by the rest range shown by the car. In reality the car measures the Wh/km continuously and from this computes the estimated rest range. I haven’t found the car’s estimate of Wh/km in the raw data but this is a way of computing it.

Battery Wh/km is the average Watts used divided by the car’s average speed while in drive. You can see the average speed further down the screen. Average Watts are not shown. The Battery Watts shown are the immediate measured Watts.

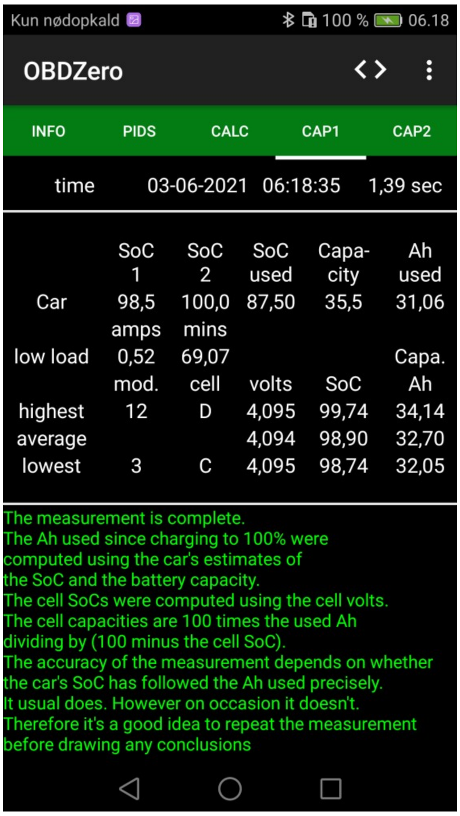
The Car Wh/km and the Battery Wh/km are usually close to each other. However if you turn on e.g. the heater the Car Wh/km increases suddenly. The Battery Wh/km should increase slowly until it is more or less the same as the Car Wh/km.

The Model Wh/km is similar to the Battery Wh/km when the heater is off and similar to the Car Wh/km when the heater is on. Again the model requires much more explanation.

The Car cap. Ah at the bottom of the Calc screen is the battery's present fully charged capacity as reported by the car (Capacity on the OBD screen). The SoC is SoC 2 as reported by the car. The first Ah is the Car cap. Ah times the SoC divided by 100. The second Ah is based on the sum of the Ah used. The last cap. Ah is the second Ah divided by the SoC times 100. This is a check sum and it should be close to the Car cap. Ah. It isn't exactly the same because of measurement errors. However if there is a large difference it may indicate that the car's estimate of the battery's 100% capacity has drifted from the true value. I have observed this drift a number of times and it is usually in the downward direction. In other words the car is under estimating the battery capacity.

The values shown on the CALC screen, the WATTS screen, the DRIVE screen and the CAPA screen are stored in the Calc_ file on the phone.

## The CAP1 screen




This screen contains a procedure for measuring the present 100% capacity of the battery. The car keeps a running estimate of the battery capacity which as a rule is quite good. However under certain circumstances the car’s estimate can be in error. In any event there may be other reasons to want an independent estimate of the present 100% capacity. The procedure for measuring the capacity shown here is much simpler than the procedure performed by the service department at a dealership and therefore less accurate. Computing must be started for this function to work.

The following steps are performed when doing the measurement:

- Before the procedure can begin both SoC1 and SoC2 must be less than 15%, equal to about a rest range of 5 km.
- When the SoC is less than 15% the SoC of all the cells is found. This is based on each cell’s voltage and it requires the load on the battery to be less than 1 amp for 30 min.

This procedure is detailed in the lower half of the CAP1 screen. As each step is completed the next step turns green.

The SoC of a cell is computed using the cell’s voltage and temperature and a model of the lithium cell. To use this model the cell must be close to equilibrium. Equilibrium means no current flowing through the cell, all charges within the cell are balanced and the temperature is constant. In the real world equilibrium is difficult to achieve therefore we approximate it by keeping the amps below 1 for 30 min. During the 30 minutes you will notice that the cell voltage will either increase or decrease to a more or less constant value.

When the measurement is complete the cells with the maximum and minimum capacities and the average capacity of the cells are shown. These capacities are computed by dividing the amp-hours (Ah) drawn from the battery since it was last charged to 100% divided by the SoC drawn from each cell and times 100. The

SoC drawn from the cell is 100% minus the SoC of each cell. The Ah drawn from the battery is the BMU's reported 100% capacity minus the present charge in Ah as shown in the Ah screen described above. The Ah drawn is shown as AhUsed in the first line under the date and time. The accuracy of this measurement depends on whether the car has kept an accurate account of the Ah to and from the battery since it was charged to 100%. Errors though rare do occur which will affect the results. On the other hand, this procedure will give a good relative measurement of the cell capacities. This means that if it shows a 2 Ah difference between the highest and lowest cells then there is about 2 Ah difference in the capacity of the cells. This difference will increase as the battery ages. We have two CZero's, one with a present capacity of 39 Ah and a cell difference of 2 Ah and the other with a present capacity of 34 Ah and a cell difference of 3 Ah.

The best way to determine if one cell is defect is to compare the lowest cell to the next lowest cells. This is possible because OBDZero computes all of the cell capacities. The CELLS screen will show the capacities of each cell when the CAP1 measurement is complete. The app also records the capacities in the Cell_ file on the phone. Annex 3 in this manual describes a method for working with the Cell_ files in the Notepad program.

For iMievs built after 2015 cell data is not available. However the car does provide the voltages of the cells with the highest and lowest voltages. Based on these voltages and the battery voltage CAP1 computes the highest, lowest and average cell capacities. Unfortunately there is no information on which cells have the highest and lowest capacities.

OBDZero cannot change the car's estimate of the battery's 100% capacity. However the the service department at a dealership can. This requires the service center to repeat the measurement and they will probably charger a fee for this service.

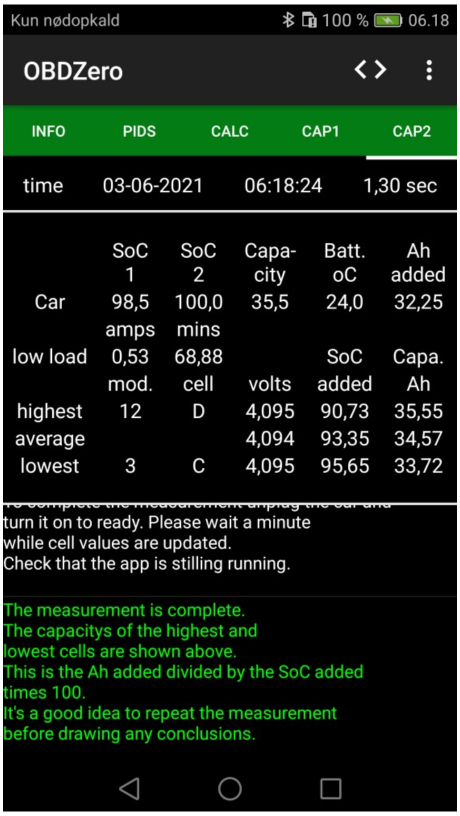
The State of Health (SoH) of the battery is the present Ah capacity divided by the battery's original capacity times 100. The battery's Ah capacity when it left the battery assembly line was 50 Ah (gross capacity). So the SoH of the battery shown above is:

71 % = 100 * 35.5/50

Even though it isn't in use the battery looses capacity in transport to the car factory and in the car on it's way to the dealership. For this reason the highest observed capacity in a car is about 48 Ah or  96% SoH.

Any feedback to me at DavidCecil@outlook.dk about this procedure will be appreciated.

## The CAP2 screen




This screen contains a more extensive procedure for measuring the present fully charged capacity of the battery than that shown on the CAP1 screen. As mentioned before the car keeps a running estimate of the battery capacity which as a rule is quite good but it can be in error. The procedure for measuring the capacity shown here is similar to the procedure performed by the service department at a dealership and is therefore more rigorous and more accurate than the CAP1 procedure described above. The first two steps of the procedure are the same as for CAP1. Therefore you can continue with a CAP2 measurement immediately after seeing the results of the CAP1 measurement. As with CAP1 OBDZero must be computing for this function to work. (See the start instructions section above)

The following steps are performed when doing the measurement:

- Before the procedure can begin both SoC1 and SoC2 must be less than 15%, equal to about a rest range of 5 km.
- When the SoC is less than 15% the SoCs of the cells are measured. This requires the load on the battery to be less than 1 amp for 30 min. The SoC's of the lowest and highest cells are shown.
- Now the car battery must be charged up to 100%.
- Once charging is complete the SoCs of the cells are measured again. As before this requires the load on the battery to be less than 1 amp for 30 min. This 30 min. period starts as soon as charging is complete.
- As a rule the 30 min have passed when one returns to the car after charging. Therefore the only step necessary to complete the measurement is to turn the car on so that the voltages and SoCs of all the cells can be read.

This procedure is detailed in the lower half of the CAP2 screen. As each step is completed the next step turns green.

The cell capacities are the amp-hours (Ah) added to the battery during charging divided by the change in the SoC of each cell and times 100. The app keeps track of both.

The SoC of a cell is computed using the cell's voltage and temperature and a model of the lithium cell. To use this model the amps through the battery must be less than 1 for 30 min. During the 30 minutes you will notice that the cell voltage will either increase or decrease to a more or less constant value.

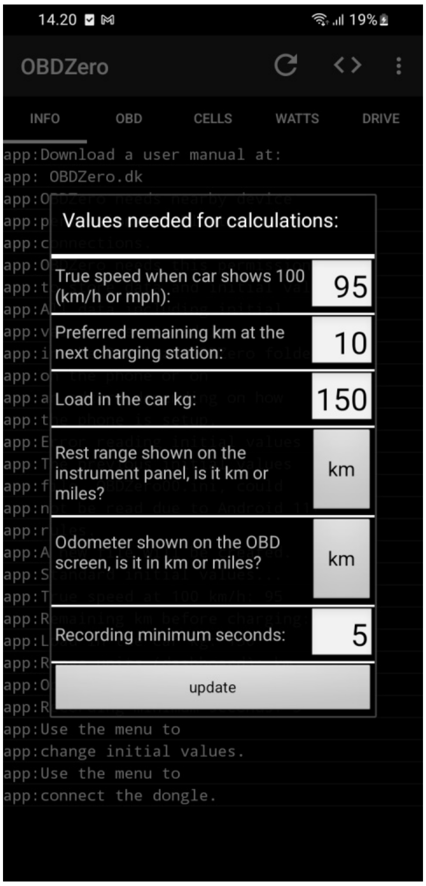
As with CAP1 the best way to determine if one cell is defect is to compare the lowest cell to the next lowest cells. This is possible because CAP2 also computes all of the cell capacities. The CELLS screen will show the capacities of each cell when the CAP1 measurement is complete.  The app also records them in the Cell_ file on the phone. Annex 3 in this manual describes a method for working with the Cell_ files in the Notepad program.

iMievs built after 2015 cell data is not available. However the car does provide the voltages of the cells with the highest and lowest voltages. Based on these voltages and the battery voltage CAP2 computes the highest, lowest and average cell capacities. Unfortunately there is no information on which cells have the highest and lowest capacities.

OBDZero cannot change the car's estimate of the battery's 100% capacity. However the the service department at a dealership can. This requires the service center to repeat the measurement and they will probably charger a fee for this service.

Any feedback to me at DavidCecil@outlook.dk about this procedure will be appreciated.

## Initial Values




This screen shows initial values.

The true speed (@ 100 km/h indicated) was previously used to accurately compute the distance travelled in the Drive screen. It is no longer used other than as information for possible future use.

The car's speed used in all computations is the average wheel speed (PID215 bytes 0 and 1). Experience has shown that there is no measureable error in the wheel speed as there normally is in the indicated speed shown on the instrument panel.

The preferred margin is added internally to the distance to the next charging station shown on the Drive screen. When one arrives at the next charging station the distance should be close to zero and the remaining range should be equal to the preferred remaining range. The preferred remaining km is used to reduce the risk of running out of electricity. It can be set to 0 but I prefer 10 km.

The load in the car is the sum of the passengers and the baggage in the car. The load plus the empty weight of the car is used to calculate the rolling resistance and the watts for acceleration in the car model. Errors in this value are usually small in relation to the total weight of the car. However when the car is fully loaded this can be significant.

The app will work satisfactorily with these values under most conditions. However the accuracy of many of the calculations mentioned previously will improve if these values are updated.

For cars sold in countries where miles are in common use the rest range shown on the instrument panel can be in miles rather than km. As yet there is no simple way for OBDZero to detect if this is the case. Many of the functions in the app depend on knowing if the rest range is in miles or km.  This is particularly true for the Drive function. Therefore the driver must tell the app which it is. By pressing the button to the right of the text the units can be changed from km to miles and back again.  Note that the preferred remaining km doesn't change when the units for the rest range change. The driver must convert his or her preferred miles to km before entering this value.

In this version of OBDZero it is also possible to change the units of the Odometer. If you are in doubt as to whether the units must be changed check the OBD screen. If the values shown there are correct both for miles and km then everything is in order.

By setting the recording minimum seconds to a number greater than the data cycle time (1 to 4 sec depending on the dongle) the amount of data in the text files saved to phone can be reduced. The app accepts values between 0 and 60 sec. At 60 sec. there will be 60 values saved for each variable per hour. There are 85 difference variables recorded in the OBD text. Therefore the OBD text file will grow by 5100 lines per hour when this value is set to 60.

If the minimum recording time is 0 sec all data is stored.

In Android versions 10 or less the initial values are stored in the OBDZero.ini file in the OBDZero folder. New rules for storage were implemented in Android 11. For 11+ phones initial values are stored in the Download/OBDZero folder. This gives one small difficulty. If OBDZero is uninstalled and then reinstalled it cannot read or delete the previously stored initial values. Therefore on 11+ phones initial values are stored in Download/OBDZero/OBDZero00.ini. With subsequent reinstallations OBDZero00.ini will be incremented to OBDZero01.ini and so on. With many reinstallations there will be many .ini files. These files are small so they do not represent a problem for storage capacity. However if you want to clean up please delete all of the .ini files not just the older .ini files. OBDZero will then create a new OBDZero00.ini file when run the next time.

### Corrections and additions to this user manual

If you have questions or you believe there are errors in this manual please contact me.
There are many subjects that require further explanation e.g. the model of the car or the model of the cell. I plan on adding annexes to the manual with more detailed descriptions including the equations used. This will take time.


### Acknowledgements and references

I have already mentioned the information on the rules for the conversion of PIDs to data made available by users on the iMiev forum. I couldn't have written this app without their help.

The code for OBDZero is built on the Bluetooth terminal program, BlueTerm, by pymasde.es found on GitHub and dated the 7th May 2014

The technical manuals for the iMiev can be found at http://mmc-manuals.ru/manuals/i-miev/online/Service_Manual/2013/index_M1.htm

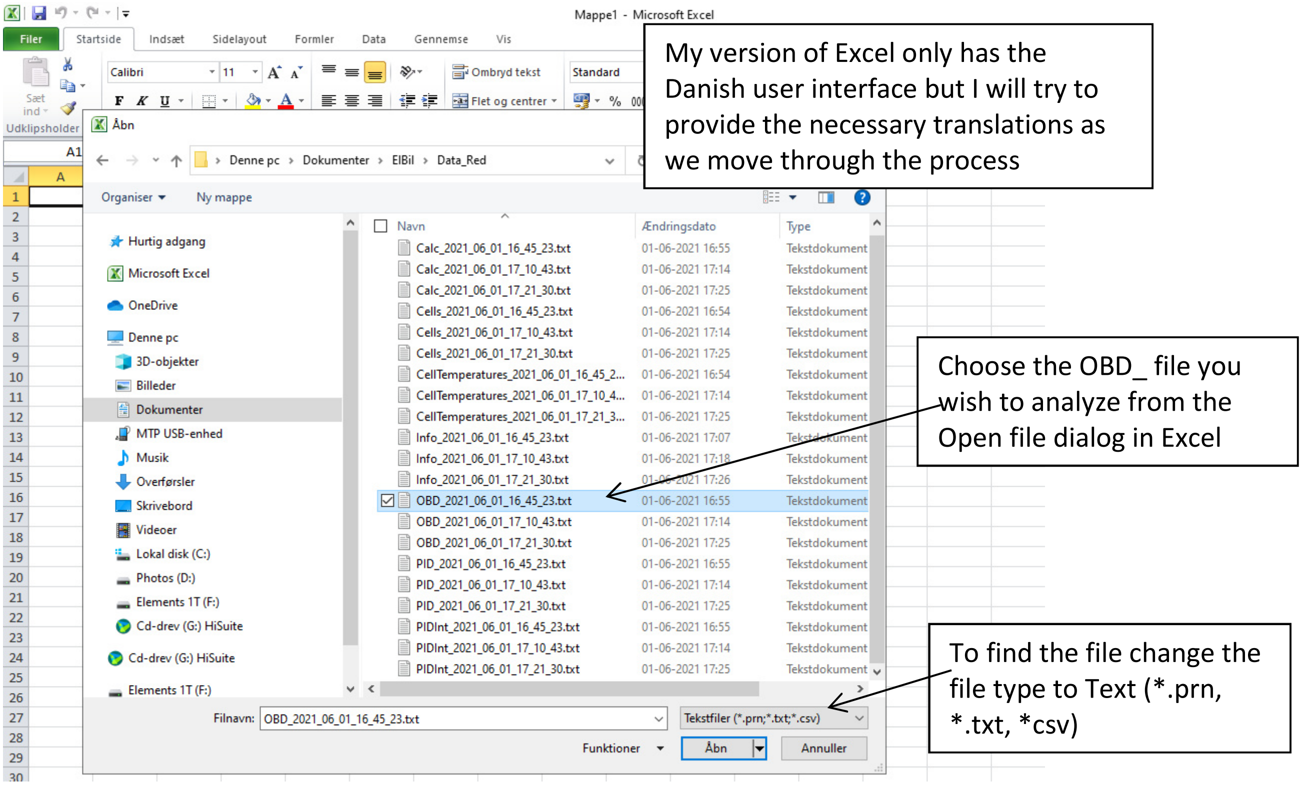
The AT commands for the OBD dongles are found in ELM327DSH.pdf from www.elmelectronics.com

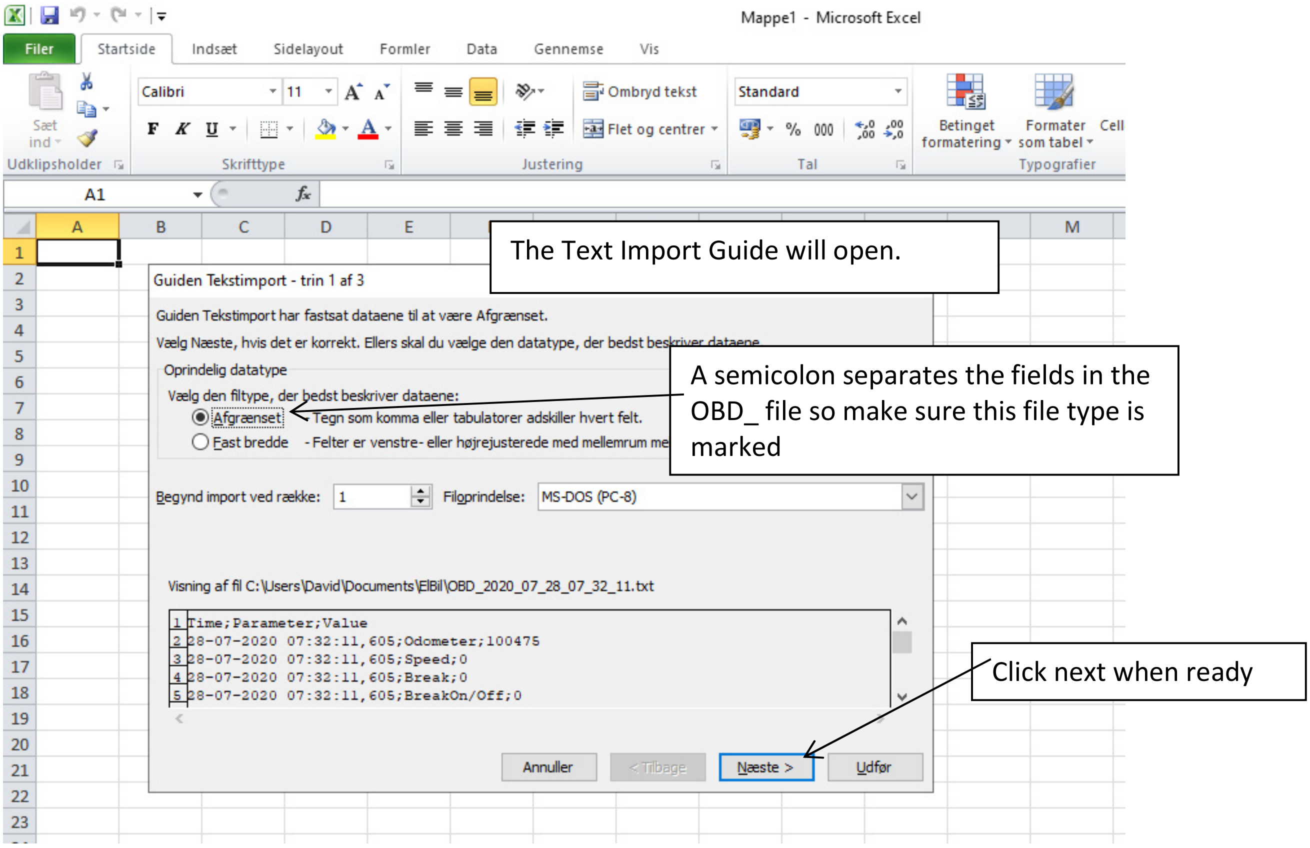
A manual for the OBDLink LX can be found at https://www.scantool.net/downloads/98/stn1100-frpm.pdf

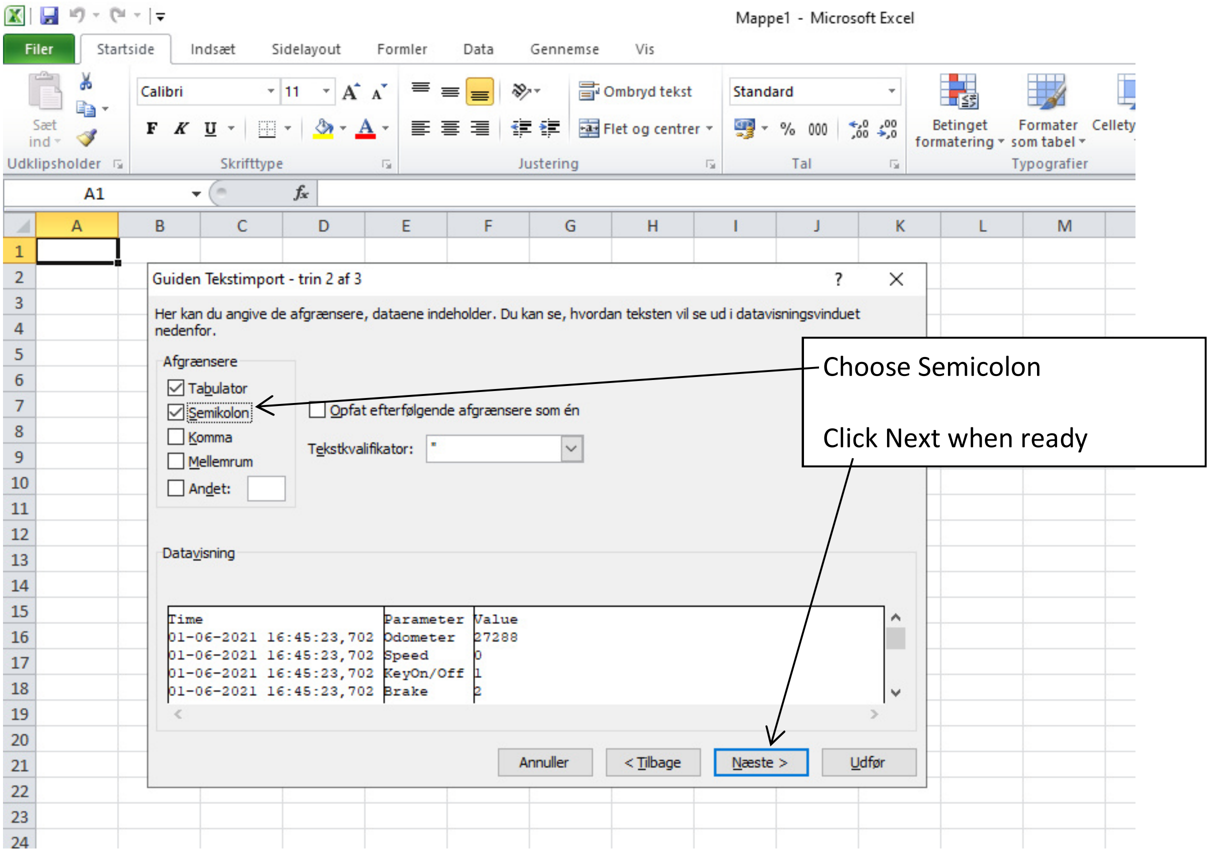
Thanks to the following users of OBDZero who have pointed out errors and made helpful suggestions: Richi, Andon, Kruno, Jason, Albert, Paolo, Florian, DginiSG, Greg, Toby, Tibor, Antonio, David, PTX and Allan.

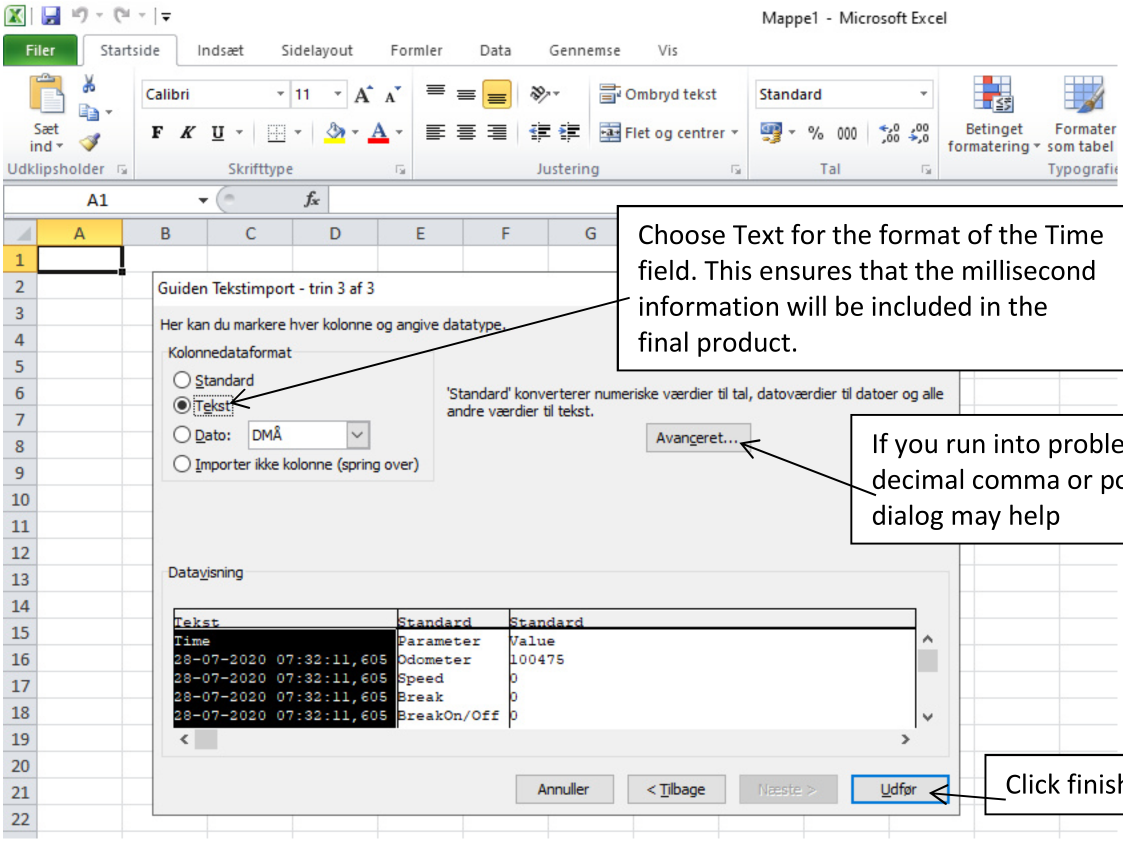
## Annex 1 Working with the OBD_ file in Excel on a Windows PC

The first step is to copy the data files from your phone to a folder on your computer. Connect the phone to the PC with a USB cable and choose file transfer. Different phones have different ways to choose file transfer. Open Windows Explorer, find the OBDZero folder on the phone and then copy the txt files to a folder on the PC.  Once there the OBD_ file can be opened in Excel.




The Text Import Guide will open.
A semicolon separates the fields in the OBD_ file so make sure this file type is marked
Click next when ready
Mappe1 - Microsoft Excel
Guiden Tekstimport - trin 1 af 3
Guiden Tekstimport har fastsat dataene til at være Afgrænset.
Oprindelig datatype
Vælg den filtype, der bedst beskriver dataene:
Afgrænset
- Tegn som komma eller tabulatorer adskiller hvert felt.
Fast bredde
Begynd import ved række:
Filoprindelse:
MS-DOS (PC-8)
Visning af fil C:\Users\David\Documents\ElBil\OBD_2020_07_28_07_32_11.txt
Time;Parameter;Value
28-07-2020 07:32:11,605;Odometer;100475
28-07-2020 07:32:11,605;Speed;0
28-07-2020 07:32:11,605;Break;0
28-07-2020 07:32:11,605;BreakOn/Off;0
Annuller
< Tilbage
Næste >
Udfør

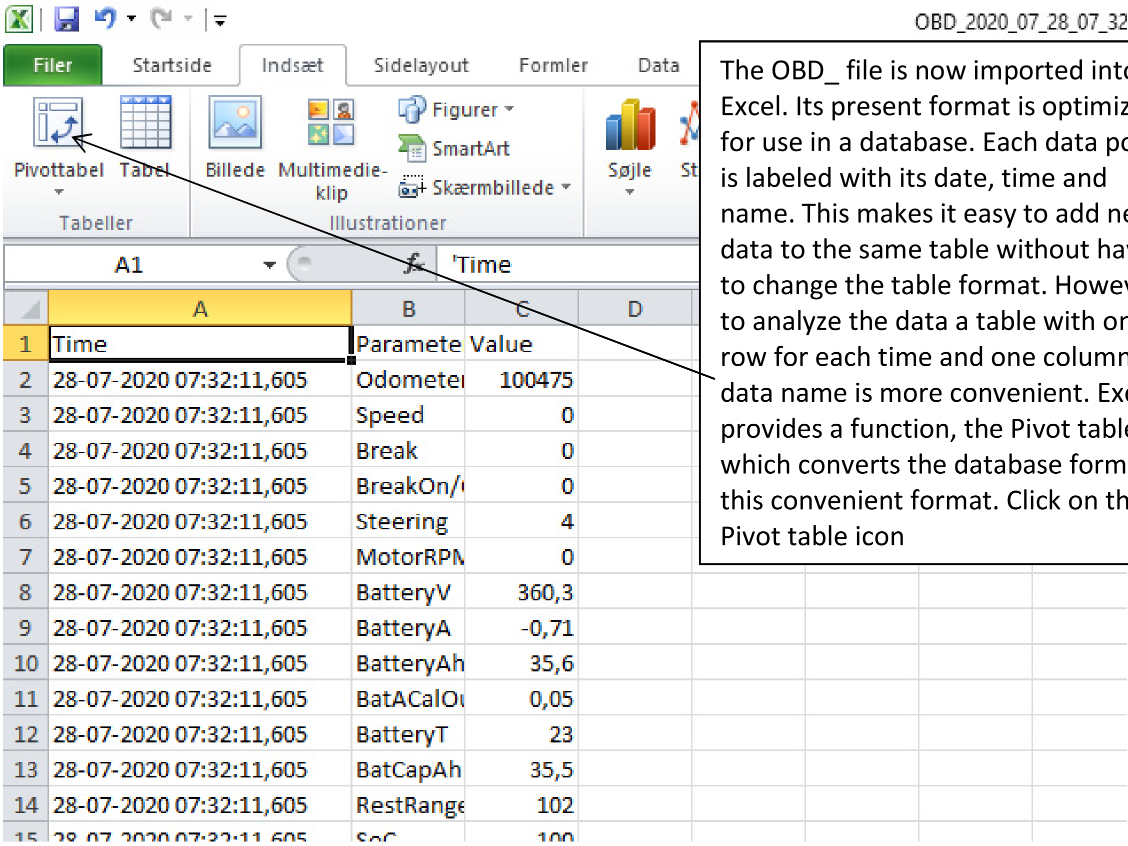

Choose Semicolon
Click Next when ready
Mappe1 - Microsoft Excel
Guiden Tekstimport - trin 2 af 3
Her kan du angive de afgrænsere, dataene indeholder. Du kan se, hvordan teksten vil se ud i datavisningsvinduet nedenfor.
Afgrænsere
Tabulator
Semikolon
Komma
Mellemrum
Andet:
Opfat efterfølgende afgrænsere som én
Tekstkvalifikator:
Datavisning
Time
Parameter
Value
01-06-2021 16:45:23,702
Odometer
27288
Speed
KeyOn/Off
Brake
Annuller
< Tilbage
Næste >
Udfør



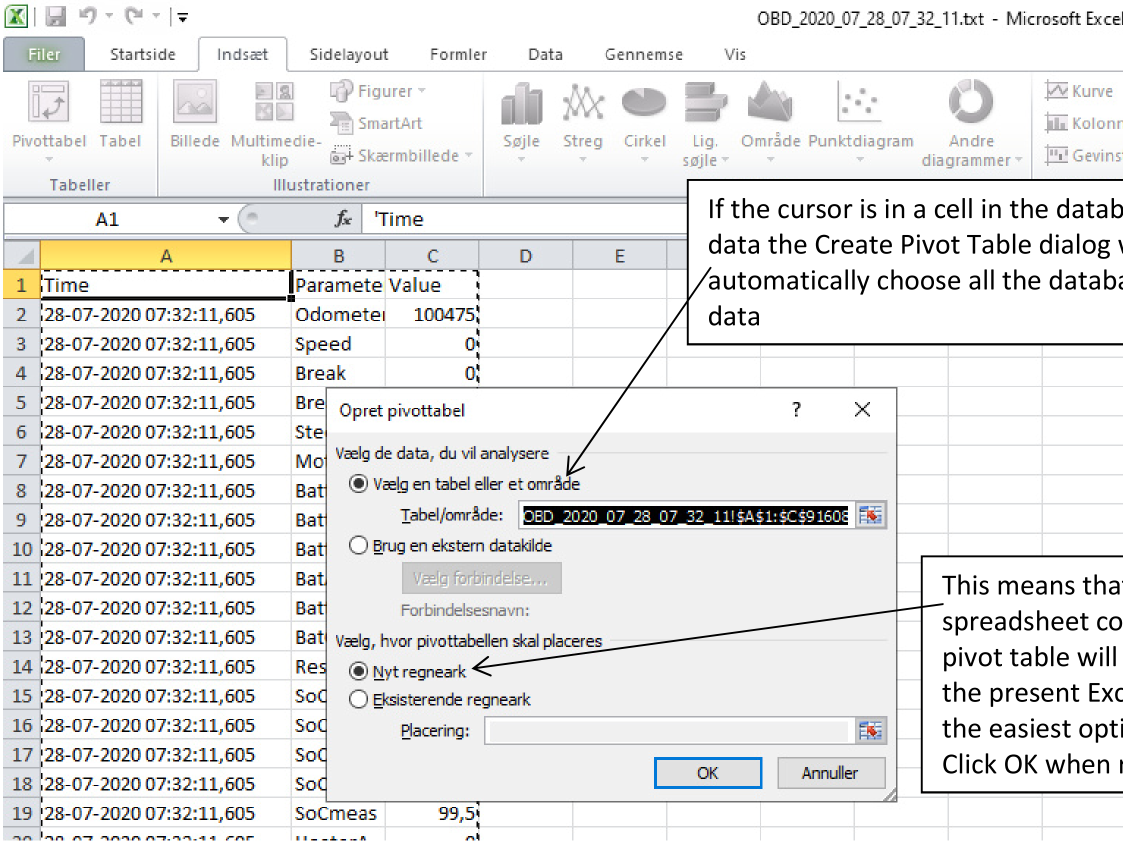

Choose Text for the format of the Time field. This ensures that the millisecond information will be included in the final product.

If you run into problems with the decimal comma or point the Advanced dialog may help

Click finish when ready.

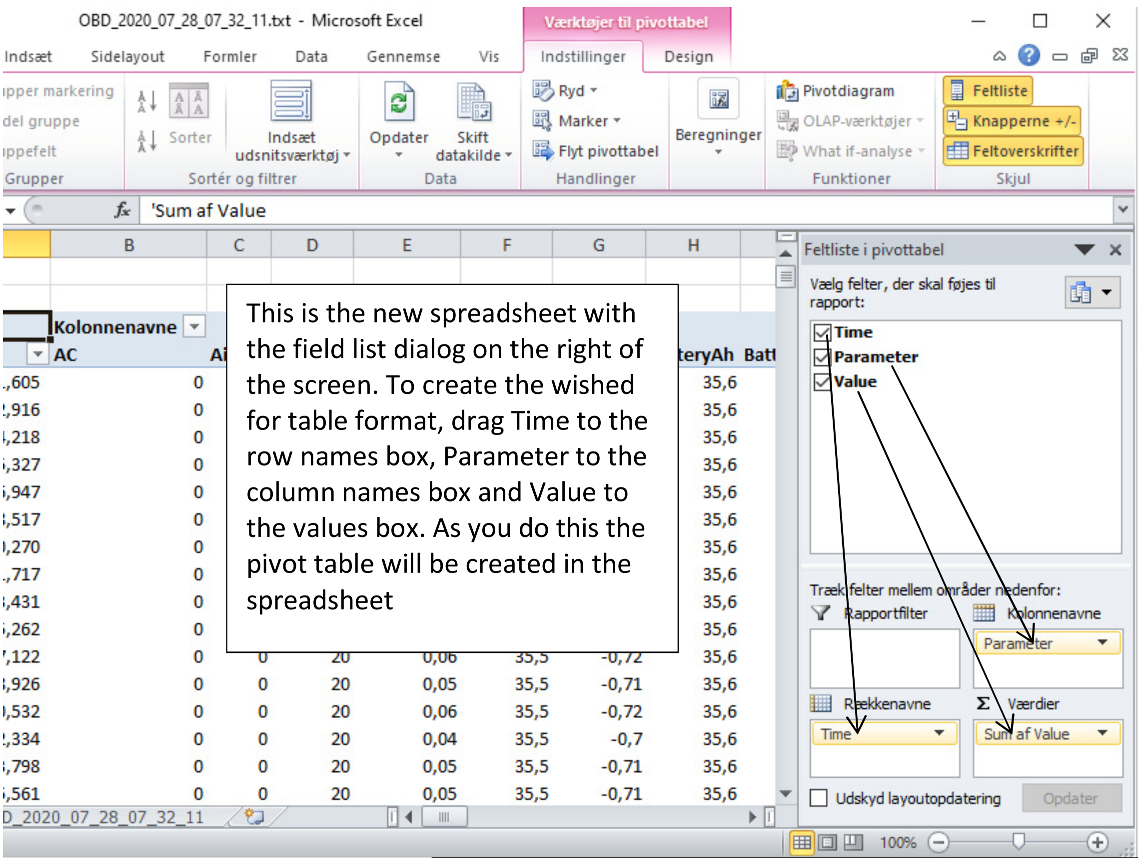



The OBD_ file is now imported into Excel. Its present format is optimized for use in a database. Each data point is labeled with its date, time and name. This makes it easy to add new data to the same table without having to change the table format. However to analyze the data a table with one row for each time and one column for data name is more convenient. Excel provides a function, the Pivot table, which converts the database format to this convenient format. Click on the Pivot table icon



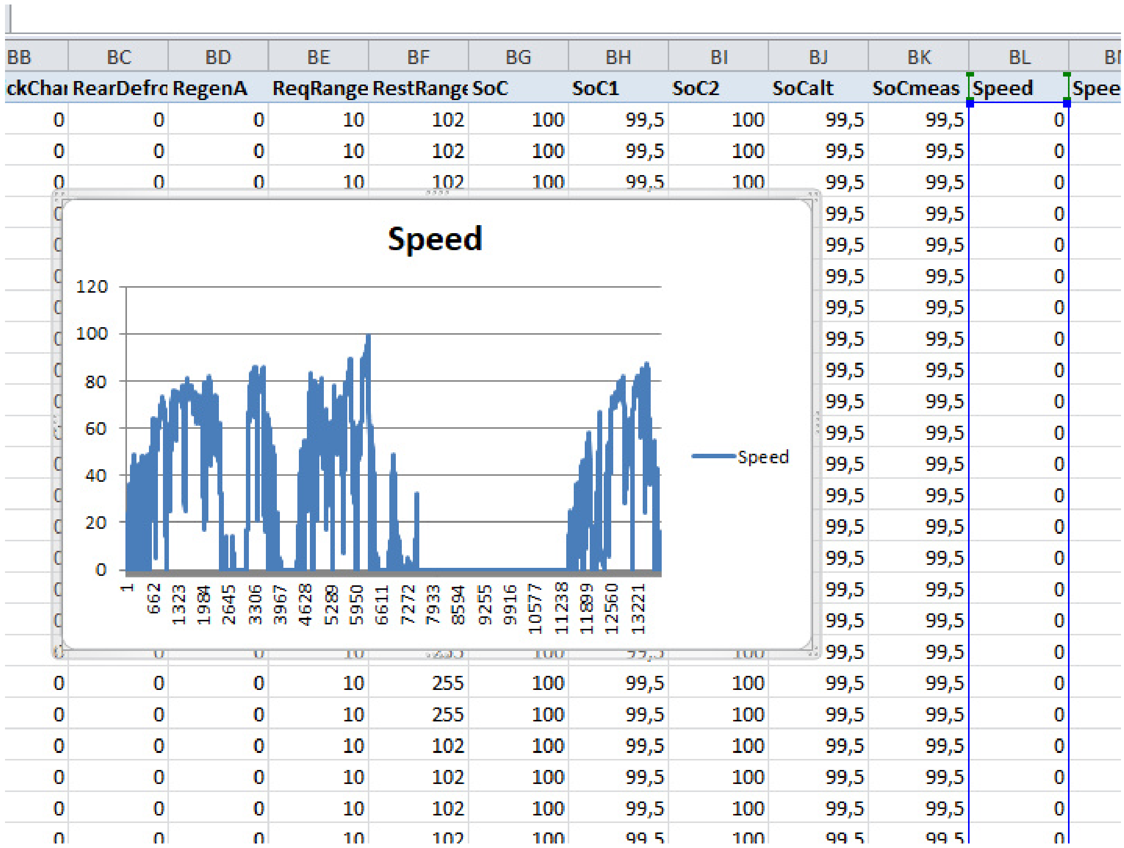

If the cursor is in a cell in the database data the Create Pivot Table dialog will automatically choose all the database data

This means that a new spreadsheet containing the pivot table will be created in the present Excel file. This is the easiest option.
Click OK when ready.



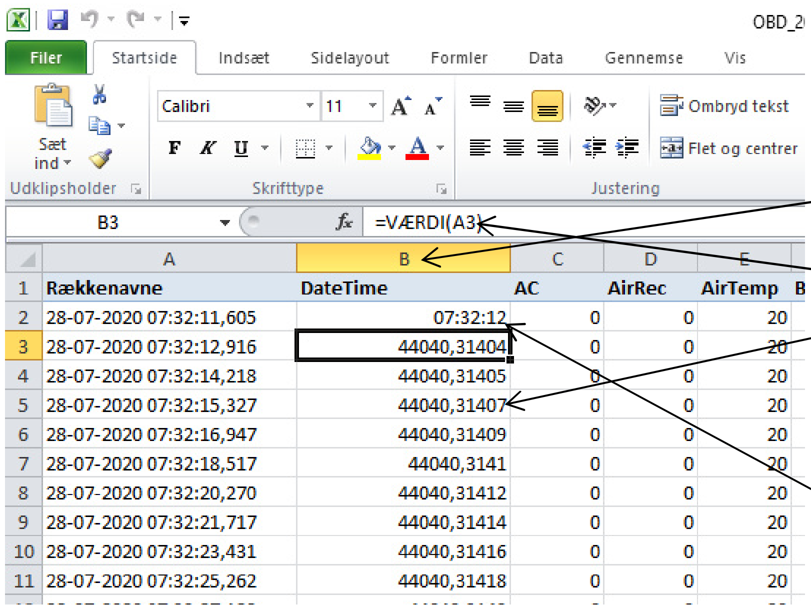

This is the new spreadsheet with the field list dialog on the right of the screen. To create the wished for table format, drag Time to the row names box, Parameter to the column names box and Value to the values box. As you do this the pivot table will be created in the spreadsheet

Note that value is the sum of the value. It isn't possible to choose the value alone and in this case it makes no difference. There is only one value for each time and parameter name so the sum of the value is the same as the value.




Once the pivot table is finished I like to copy the result to a third spreadsheet before creating graphs of the data. This is because the pivot table has some built-in rules that make it different from normal columns of data in Excel. Don't copy the totals along the bottom row and the right hand column of the pivot table. They have no meaning and they only create problems when graphing the data.

There is one last step and that is converting the Time from a text to a number in Excel's date and time system.




| | Rækkenavne | DateTime | AC | AirRec | AirTemp |
|---|---|---|---|---|---|
| 2 | 28-07-2020 07:32:11,605 | 07:32:12 | 0 | 0 | 20 |
| 3 | 28-07-2020 07:32:12,916 | 44040,31404 | 0 | 0 | 20 |
| 4 | 28-07-2020 07:32:14,218 | 44040,31405 | 0 | 0 | 20 |
| 5 | 28-07-2020 07:32:15,327 | 44040,31407 | 0 | 0 | 20 |
| 6 | 28-07-2020 07:32:16,947 | 44040,31409 | 0 | 0 | 20 |
| 7 | 28-07-2020 07:32:18,517 | 44040,3141 | 0 | 0 | 20 |
| 8 | 28-07-2020 07:32:20,270 | 44040,31412 | 0 | 0 | 20 |
| 9 | 28-07-2020 07:32:21,717 | 44040,31414 | 0 | 0 | 20 |
| 10 | 28-07-2020 07:32:23,431 | 44040,31416 | 0 | 0 | 20 |
| 11 | 28-07-2020 07:32:25,262 | 44040,31418 | 0 | 0 | 20 |

In the spreadsheet shown above I have added a column between columns containing the row names (A) and the first column of data (B). In this column I used the value function to convert the date and time in column A to the Excel date series number in the new column, column B. These are the numbers starting in the third row and copied down to convert all the times. Finally I changed the format of the first cell under DateTime to a time format. This can also be copied.

Using the DateTime column together with a data column you can create xy graphs with time on the x axis.

While we are on the subject of time there is a limit we need to be aware of. In Excel 2010, the version I use, there are 1048576 rows. In the OBD_ file there are 85 data points computed in each data cycle. The fastest dongles have a cycle time of about 1 second. 1048576 divided by 85 times 1 second is 12336 seconds or 3.4 hours. If all the computed values are recorded in the OBD_ file, it would grow beyond the number of rows in Excel 2010 in just 3½ hours.  As mentioned in the section on Initial Values the amount of data can be reduced by setting the minimum recording time to a value greater than the cycle time. With the max of 60 sec. 24 hours of data can easily be imported into and worked with in Excel 2010.

## Annex 2 OBD data defined

The following is a description of the data recorded in the OBD_ file

| Parameter | Units | Description | Source | Conversion | Notes |
|---|---|---|---|---|---|
| AC | on/off | Airconditioning | PID 3A4 | byte(0) bit 128 | |
| ACAmps | Amps | Airconditioning amps | PID 384 | (byte(0) * 256 + byte(1))/1000 | |
| AirRec | on/off | Air recirculation | PID 3A4 | byte(0) bit 64 | |
| AirTemp | oC | Air temperature | Initial values | | |
| BatACalOut | Amps | Amps out of the battery calibrated | PID 373 | -(byte(2)*256 + byte(3) -32700)/100 | 4 |
| BatCapAh | Amp-hours | Battery 100% capacity | PID 374 | byte(6)/2 | |
| BatteryA | Amps | Amps in to the battery (both + & -) | PID 373 | (byte(2)*256 + byte(3) -32768)/100 | 4 |
| BatteryT | oC | Average cell temperature | Cells averaged | | |
| BatteryTmax | oC | Maximum cell temperature | PID 374 | byte(4) – 50 | |
| BatteryTmin | oC | Minimum cell temperature | PID 374 | byte(5) – 50 | |
| BatteryV | Volts | Battery volts | PID 373 | (byte(4)*256 + byte(5))/10 | |
| Brake | unknown | Brake pressure | PID 208 | byte(3) | |
| BrakeOn/Off | on/off | Brake | PID 231 | byte(4) | |
| CellVmaxCell | number | Max volts cell number | Cells sorted | | 3 |
| CellVmaxMod | number | Max volts cell module | Cells sorted | | 3 |
| CellVmaxTemp | oC | Max volts cell temperature | Cells sorted | | 3 |
| CellVmaxVolt | Volts | Max volts cell volts | Cells sorted | | 3 |
| CellVminCell | number | Min volts cell number | Cells sorted | | 3 |
| CellVminMod | number | Min volts cell module | Cells sorted | | 3 |
| CellVminTemp | oC | Min volts cell temperature | Cells sorted | | 3 |
| CellVminVolt | Volts | Min volts cell volts | Cells sorted | | 3 |
| CellVsum | Volts | Sum cell voltages | Cells summed | | 3 |
| Charge12Amps | Amps | 12volt battery charger amps in | PID 384 | byte(3)/100 | |
| ChargeADC | Amps | Charging unit amps to battery | PID 389 | byte(2)/10 | 4 |
| ChargeTemp1 | oC | Charging unit temperature 1 | PID 389 | byte(3) – 50 | |
| ChargeTemp2 | oC | Charging unit temperature 2 | PID 389 | byte(4) – 50 | |
| ChargeVAC | Volts | Charging unit volts from mains | PID 389 | byte(1) | |
| ChargeVDC | Volts | Charging unit volts to battery | PID 389 | 2*byte(0) + 1 | |
| ChargeAAC | Amps | Charging unit amps from mains | PID 389 | byte(6)/10 | |
| FanDirect | number | Air direction control position | PID 3A4 | byte(1) low nibble | |
| FanMax | on/off | Max fan | PID 3A4 | byte(0) bit 32 | |
| FanSpeed | number | Fan speed | PID 3A4 | byte(1) high nibble | |
| Gear | number | 3 = P/N, 1 = Reverse, 4 = Drive | PID 285 | see note 2 | 2 |
| Gear418 | number | ASCII for P, R, N, D, B | PID 418 | see note 2 | 2 |
| Heat/Cool | number | Temperature control position | PID 3A4 | byte(0) low nibble | |
| HeaterA | Amps | Heater amps | PID 384 | byte(4)/10 | 4 |
| KeyOn/Off | on/off | Key on (either ready or not ready) | PID 412 | byte(0) = 254 | |
| LDrive | on/off | Headlights | PID 424 | byte(1) bit 32 | |
| LFrontFog | on/off | Front fog lights | PID 424 | byte(0) bit 8 | |
| LHigh | on/off | Highbeams | PID 424 | byte(1) bit 4 | |
| Loadkg | Kg | Passengers and baggage | Initial values | | |
| LPark | on/off | Parking lights | PID 424 | byte(1) bit 64 | |
| LRearFog | on/off | Rear fog lights | PID 424 | byte(0) bit 16 | |

| Parameter | Units | Description | Source | Conversion | Notes |
|---|---|---|---|---|---|
| Margin | Km | Preferred km at next charging station | Initial values | | |
| MotorA | amps | Motor current | PID 696 | (byte(2)*256 + byte(3) - 500)/20 | 4 |
| MotorRPM | rpm | Motor rpm | PID 298 | byte(6)*256 + byte(7) – 10000 | |
| MotorTemp0 | oC | Motor temperature 0 | PID 298 | byte(0) – 50 | |
| MotorTemp1 | oC | Motor temperature 1 | PID 298 | byte(1) – 50 | |
| MotorTemp2 | oC | Motor temperature 2 | PID 298 | byte(2) – 50 | |
| MotorTemp3 | oC | Motor temperature 3 | PID 298 | byte(3) – 50 | |
| Odometer | Km | Total km driven | PID 412 | byte(2)*256*256 + byte(3)*256 + byte(4) | |
| PIDCount | number | Number of new PIDs received | program counter | | |
| QuickCharge% | % | Chademo percent | PID 697 | byte(1) | |
| QuickChargeA | amps | Chademo current | PID 697 | byte(2) | 4 |
| QuickChargeOn/Off | on/off | Chademo connected | PID 697 | byte(0) | |
| RearDefrost | on/off | Rear window defrost | PID 424 | byte(6) bit 8 | |
| RegenA | amps | Regeneration amps (negative) | PID 696 | (byte(6)*256 + byte(7) - 10000)/5 | 4 |
| RestRange | Km | Car's estimate of remaining km | PID 346 | byte(7) | |
| SoC1 | % | State of Charge 1 | PID 374 | (byte(0) - 10)/2 | |
| SoC2 | % | State of Charge 2 | PID 374 | (byte(1) - 10)/2 | |
| SpdShown | km/h | Indicated speed | PID 412 | byte(1) | |
| Speed0 | km/h | wheel speed | PID 215 | (byte(0)*256 + byte(1))/128 | |
| Steering | degrees | Steering wheel position | PID 236 | (byte(0)*256 + byte(1) -4096)/2 | |
| WindWiper | on/off | Windshield wipers | PID 424 | byte(1) bit 8 | |
| BMUCapAh | Ah | Battery capacity at 100% SoC | PID 762 byte(0) = 36 | (byte(3)*256 + byte(4))/10 | 5 |
| BMURemAh | Ah | Battery remaining Ah | PID 762 byte(0) = 36 | (byte(5)*256 + byte(6))/10 | 5 |
| VIN first 7 char. | | | PID 6FA byte(0) = 0 | Char(i) = ASCII(byte(i)) | 6 |
| VIN last 7 char. | | | PID 6FA byte(0) = 1 | Char(i) = ASCII(byte(i)) | 6 |

Note 1
Each PID contains 1 to 8 bytes. The first byte is numbered 0 and the last byte is numbered 7. When on/off information is code it is usually coded in the bits of a byte, a bit equal to 1 means on. The bits are numbered in the table above by positional value in the byte. Of the 8 bits in a byte bit 128 is the most significant and bit 1 the least significant. The low nibble in a byte is the number represented by the 4 least significant bits and the high nibble is the number represented by the 4 most significant bits.

Note 2
In previous versions the gear position was obtained from PID 285. If PID 285 byte(6) equals 12 then the position is either Park or Neutral with no indication of which. If byte(6) equals 14 and byte(7) equals 16 then the position is Drive. If byte(6) equals 14 and byte(7) doesn't equal 16 then the position is Reverse.

SilasAT found that PID418 byte(0) is the ASKII number for the letters P, R, N, D, and B. This has been added to the program. However the gear based on PID 285 has been retained for continuity. The new gear positon is called Gear418.

Note 3
The coding of the individual cell information is found in PIDs 6E1-6E4. Byte(0) is the module number. See annex 3 for a more detailed description of the cell data.

Note 4

The BatteryA, the current to the battery can be both positive and negative. The formula for BatteryA is from the iMiev forum. It is the original formula for the amps measurement published in 2012. See: https://myimiev.com/forum/viewtopic.php?p=4805#p4805

The sign of BatteryA is in relation to the battery so charging is positive and discharging is negative.

BatACalOut, the calibrated current, is based on newer research in 2023. See: https://myimiev.com/forum/viewtopic.php?p=45916#p45916

With BatACalOut the current to the battery is 0.68 amps larger than BatteryA and the current from the battery is 0.68 amps smaller than BatteryA. As most of the computations in OBDZero are from the perspective of the motor and auxiliary functions, the sign of the current is reversed relative to BatteryA. Therefore discharge is positive and charging is negative.

MotorA, motor current is always positive and RegenA, the regeneration current is always negative. HeaterA, heater current and ACAmps, air-conditioning current are both positive. However this convention doesn't apply to ChargeADC, charging unit current to the battery and QuickChargeA, Chademo current to the battery both of which are also positive.

Note 5

To collect the BMU's capacity and the remaining Ah values, OBDZero sets the dongle to a different collection mode then resets the dongle back to normal. This slows the data collection process so this value is only collected at 1 minute intervals. The CAN instructions for the BMU capacity were supplied by anko on https://www.myoutlanderphev.com/forum/viewtopic.php?f=10&t=1796

Note 6

The VIN number is shown on the OBD screen but it isn't saved in the OBD_ text file

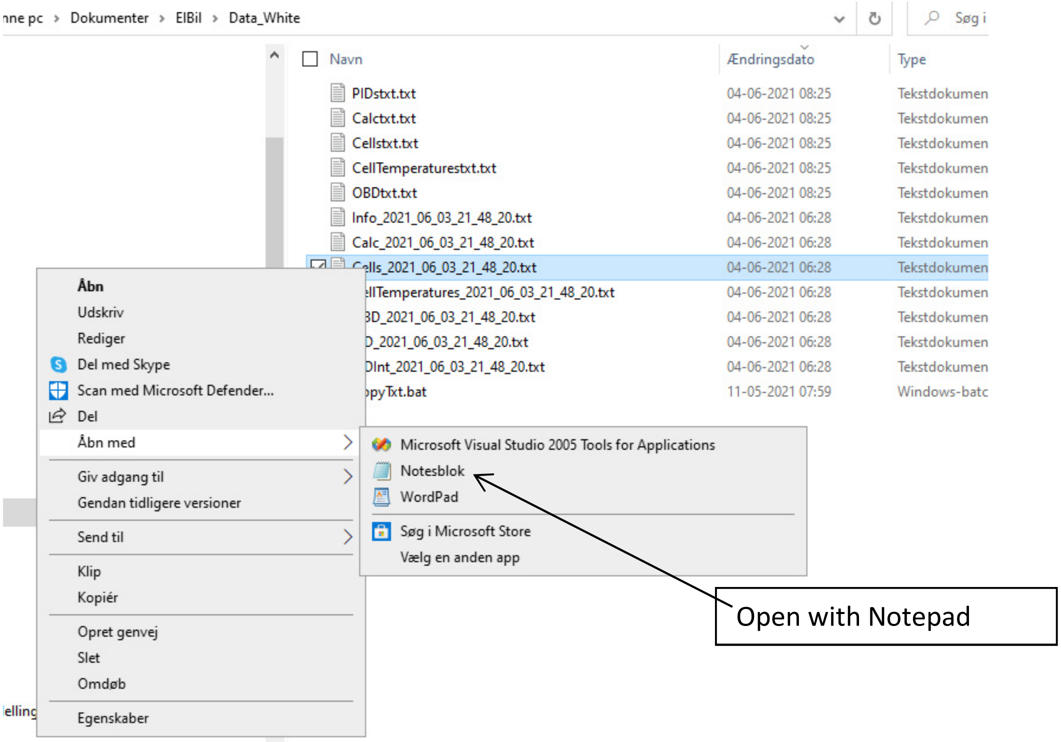
Note 7

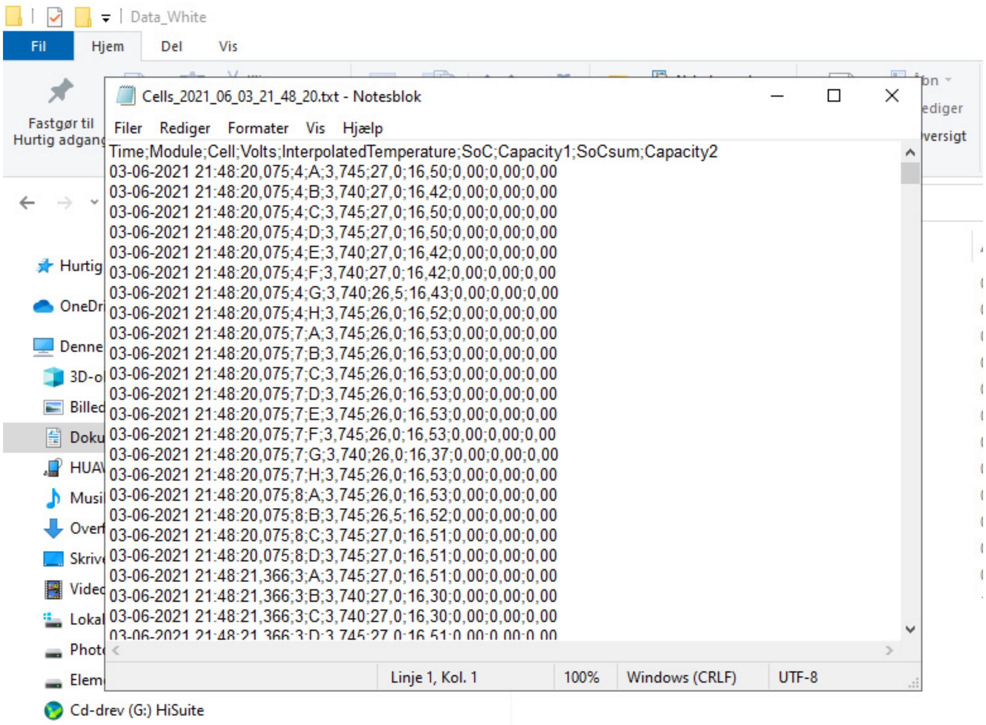
Earlier versions of OBDZero also saved the following parameters. These parameters have been replaced as shown in the table.

| **Parameter** | **Replaced by** | **Reason** |
|---|---|---|
| Break | Brake | Spelling error |
| BreakOn/Off | BrakeOn/Off | Spelling error |
| SoC | SoC2 | More specific |
| SoCalt | SoC1 | More specific |
| SoCmeas | SoC1 | More specific |
| CellVmaxVolts | CellVmaxVolt | Shorter, saves space in the database |
| CellVminVolts | CellVminVolt | Shorter, saves space in the database |
| ReqRange | Margin | More specific |
| BatteryAh | BatCapAh * SoC2/100 | Compute when needed, saves space |
| ACW | ACAmps * BatteryV | Compute when needed, saves space |

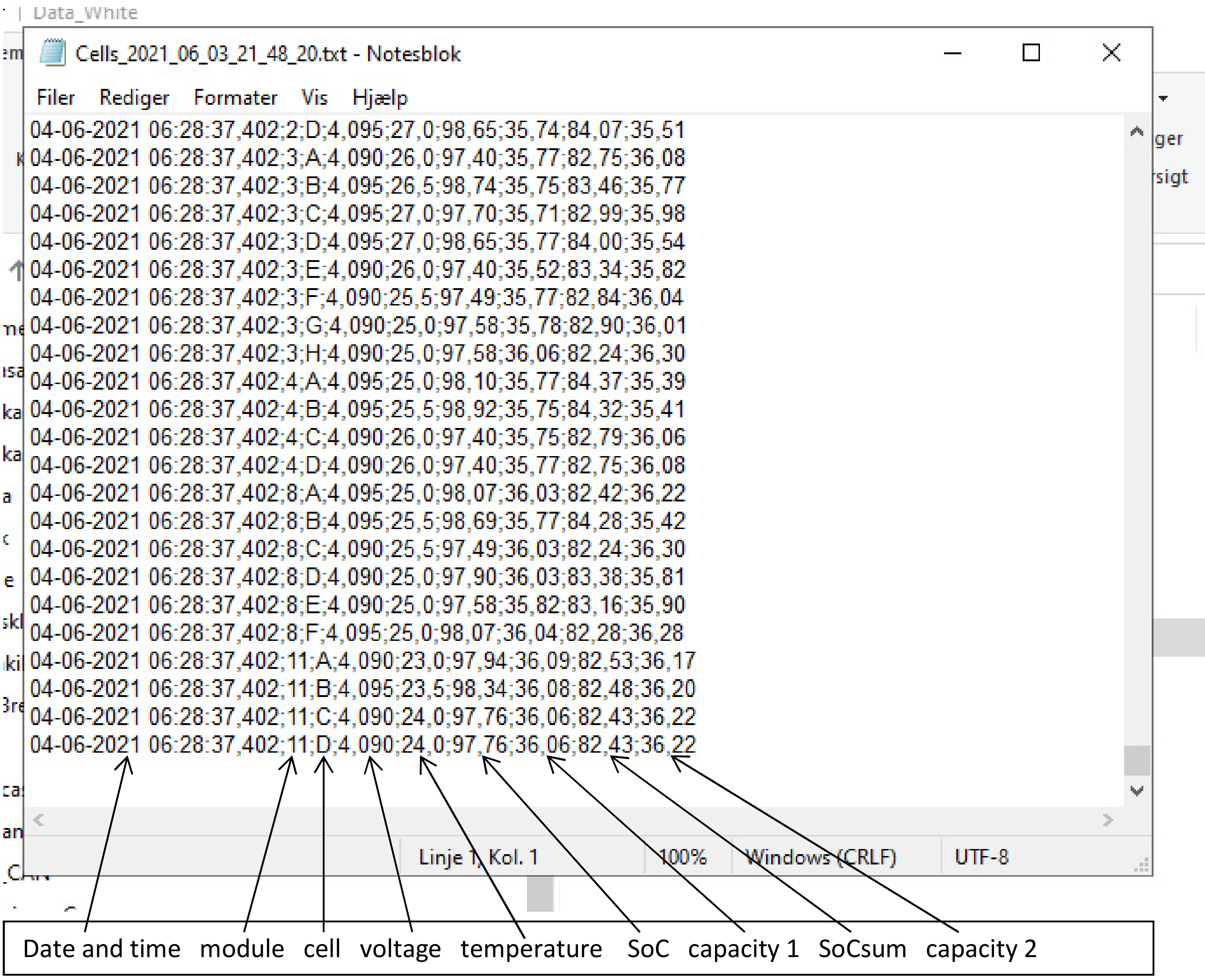
## Annex 3. Reading the capacities of individual cells in the Cell_ text file

Once CAP1 or both CAP1 and CAP2 measurements are complete and the app is closed the capacity results can be found at the end of the Cells_ file. As with the OBD_ file, the first step is to copy the data files from your phone to a folder on your computer. Connect the phone to the PC with a USB cable and choose file transfer. Open Windows Explorer, find the OBDZero folder on the phone and then copy the txt files to a folder on the PC. Once there the Cells_ file can be opened in Notepad. In this case, Notepad is easier than using Excel.



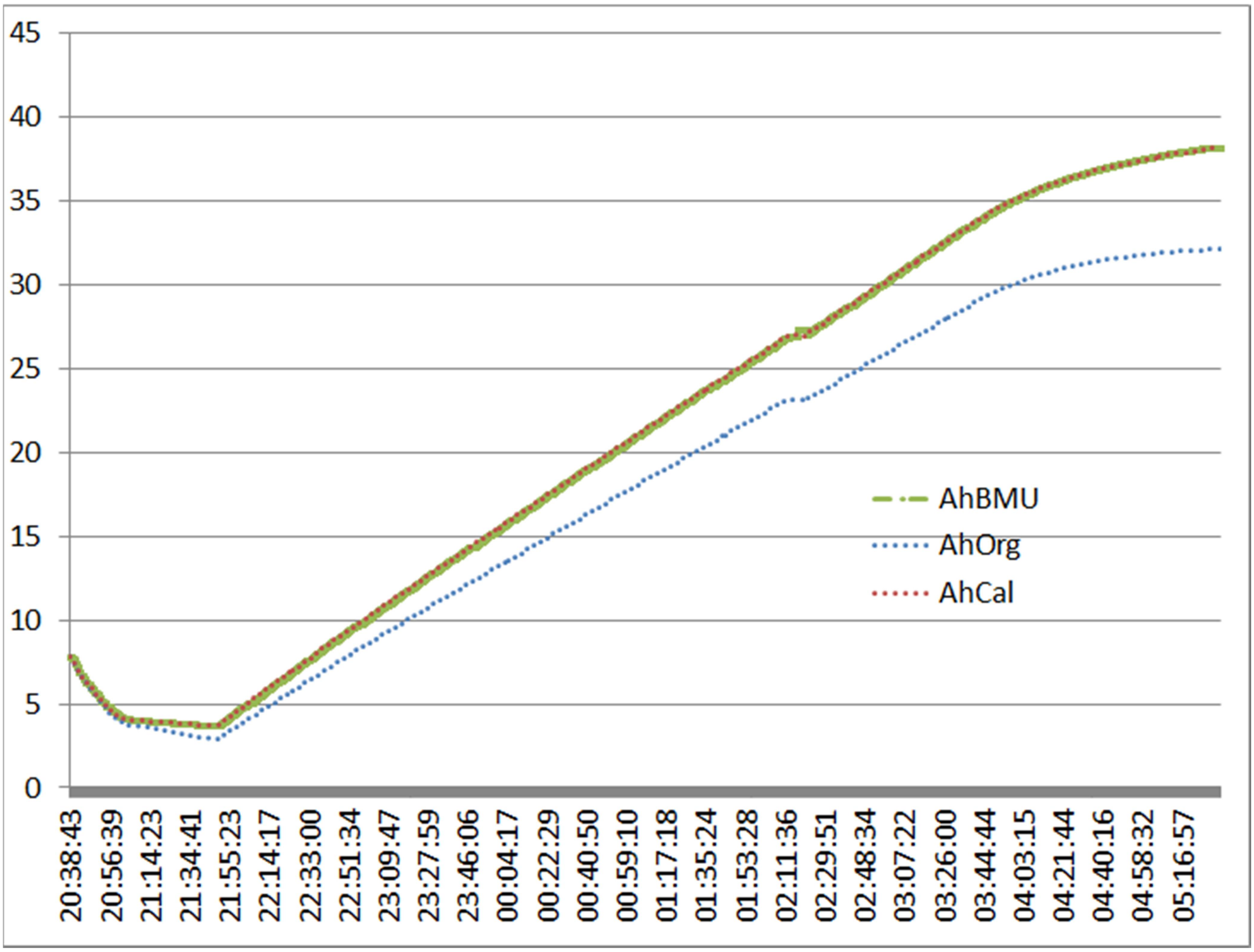
Scroll down to the end of the file and note of the values for each cell. As a rule you will find all 88 (80) cells in the last 10 time steps.




The Date and Time is day, month, year, hour, minute,  and seconds with microseconds. The SoC is the present value. If only CAP1 was performed then this will be less than 15%. If both CAP1 and CAP2 were performed then SoC should be close to 100%. Capacity 1 is the result of the CAP1 function. SoCsum is used to compute capacity 2. Lastly capacity 2 is the result of the CAP2 function. If only CAP1 was performed then these last two number with be 0 or close to 0. Capacity 1 and capacity 2 are almost always different. Capacity 2 is more accurate than capacity 1. However, both can be used to compare the capacities of the cells.

OBDZero cannot change the car's estimate of the battery's 100% capacity. However the the service center at a dealership can. This requires the service center to repeat the measurement and their will probably a charger a fee for this service.

## Annex 4. Computing the Amps to and from the Battery




This graph shows a slow charging of the battery recorded a couple of months ago. The green line is the BMU's present charge in amp hours (AhBMU) using data from PID 762. The red (AhCal) and blue (AhOrg) dotted lines are computed present charges in Ah based on the amps to and from the battery.

AhOrg is the accumulated Ah using this formula for the amps to and from the battery:
PID 373: (((D3 * 256) + D4) - 32768) * 0.01 = Pack Amps In

AhCal is the accumulated Ah using this formula:
PID 373: (((D3 * 256) + D4) - 32700) * 0.01 = Pack Amps In

To compute the accumulated Ah for the red and blue lines, I first computed the added Ah in each time step by multiplying the time step in hours with the average of the amps at the start and finish of the time step. Then I started both the red and the blue lines in the graph with the AhBMU start value of 7.8 Ah. Finally I added the step Ah to the previous sum of Ah for each step from 20:38:43 to 05:35:02. At 05:35:02 the BMU Ah was 38.2, AhOrg was 32.12 and AhCal was 38.19.

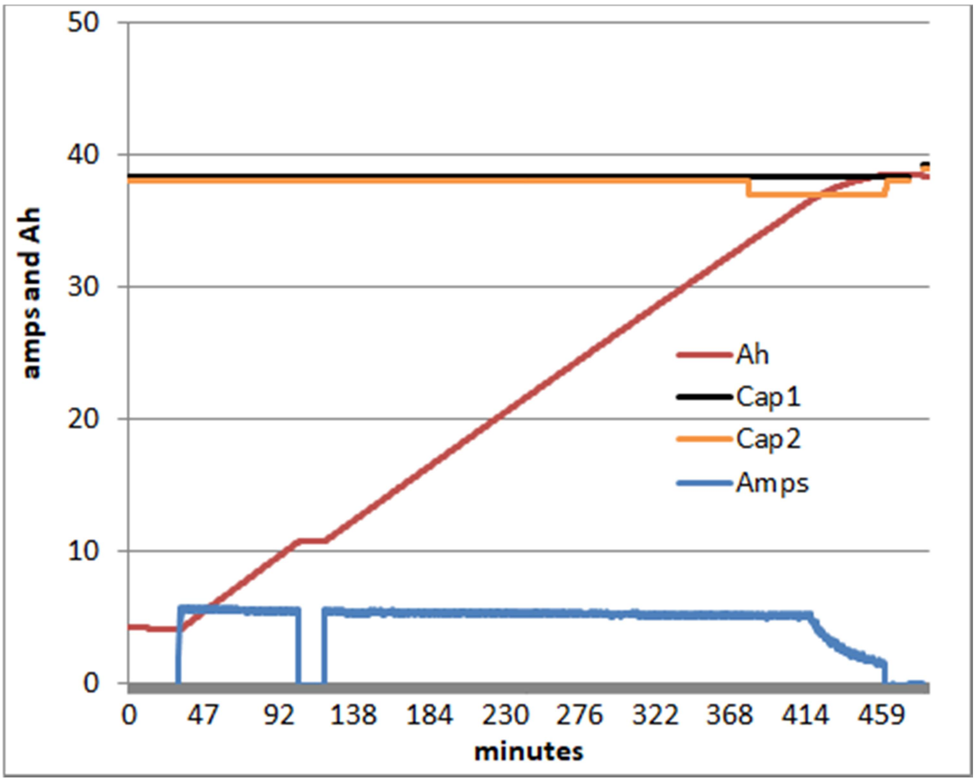
It is clear that the accumulated Ah based on the AhCal formula agrees much better with the BMU Ah. Therefore this is the formula for the battery amps that is used in OBDZero.

I used the formula for AhOrg in the first versions of OBDZero. I believe this formula came originally from a file named iMievFormProgram.txt that I found back in 2016. There is no author given.

In 2018 I argued for a correction of + 0.66 amps based on a comparison of the accumulated Ah when charging and discharging. https://myimiev.com/forum/viewtopic.php?p=36989#p36989
I now know that +0.68 is the better correction. This gives the same result as the AhCal formula above. A correction of only +0.66 gives an Ah of 38.02 compared to the BMU value of 38.2. Not bad but not as good as AhCal.

**Annex 5. What happens when the i-MiEV, CZero and iOn are slow charged**.

Sometimes when our cars are slow charged from SoC less than SoC 20% to 100%, the battery capacity is measure by the BMU. Here is a detailed description of the process based on a slow charge I recorded a month ago.



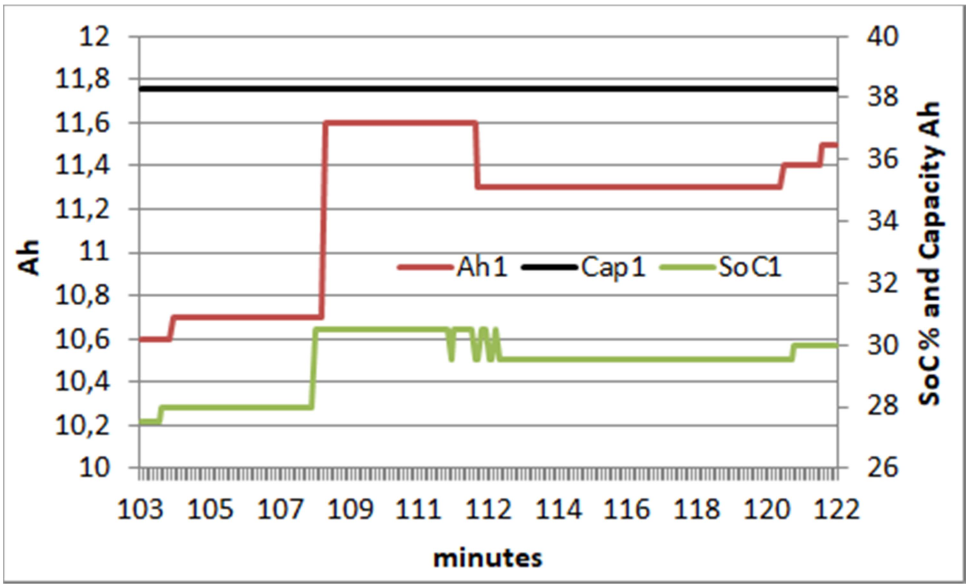

Slow Charge May 2023

To start with here are some **definitions**.
Cap1 is the capacity reported by CaniOn and EVBatMon. OBDZero reports it as the Battery Management Unit (BMU) capacity. Ah1 is the present charge in Ah in the battery corresponding to Cap1. OBDZero and EVBatMon reports this but not CaniOn. SoC1 is the State of Charge equal to 100 * Ah1/Cap1. SoC1 is reported by EVBatMon as SoC. It is reported by OBDZero as SoC1. CaniOn does not report it. Range is computed using Cap1, Ah1 and SoC1.

Cap2 is a second capacity computed by the BMU. It isn't reported by CaniOn or EVBatMon. It is reported by OBDZero as just the battery capacity.  SoC2 is the SoC reported by CaniOn but not EVBatMon. It is reported in OBDZero as SoC2. Ah2 is the present charge in the battery corresponding to Cap2. It can be computed by Ah2 = SoC2 *Cap2/100. I have not found a PID containing Ah2 but I'm fairly sure that the BMU keeps an account of Ah2.

Now the **slow charge data**
During the slow charge shown in the graph above and before SoC1 reached 28 %, Ah1 and Ah2 increased by counting the amp-hours to the battery. This is called Coulomb counting. The BMU computes SoC1 and SoC2 according to:
SoC1 = 100 * Ah1 / Cap1
SoC2 = 100 * Ah2 / Cap2
Cap1 was 38.3 Ah and Cap2 was 38 Ah.

On this day the charging pause that we often see at about SoC 30% began when SoC1 was 28%. Looking at other slow charges the pause can begin between 24% and 37%. (Even though charging starts at less than 24% the BMU often waits until SoC1 is about 70% before pausing the charge. I don't know why.) Studying the data It appears that the trigger is not the SoC but a battery voltage of about 339 V while charging.



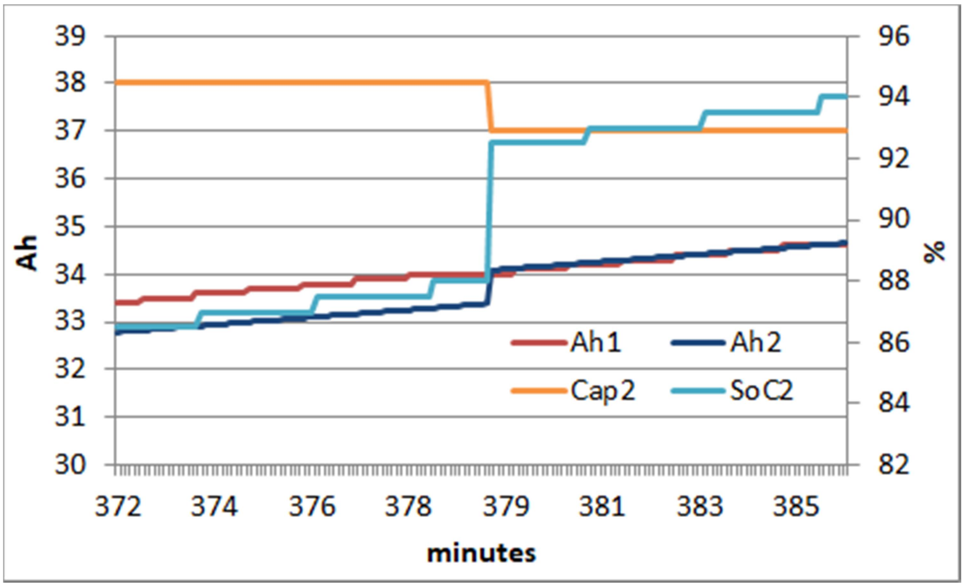

During the pause shown here between 105 minutes and 120 minutes, charging stopped and the amps went from about 5 down to about -0.5. While the current was low, the BMU measured the open circuit voltage (Voc) and computed the true SoC of the battery. SoC1 is adjusted to the true SoC using a simple function such as:
SoC1 = a* Voc + b
The factors a and b are specific to the LEV50(N) cell but I don't know what they are yet.
Cap1 didn't change and Ah1 was updated according to:
Ah1 = SoC1 * Cap1/100
In this case SoC1 was 29.5 % when the pause ended and
Ah1 = 29.5 * 38.3 / 100 = 11.3 Ah
Studying other slow charges, SoC1 varied between 29 % and 32 % at the end of the pause.

Ah2 continued to increase by Coulomb counting but changed only slightly because the current from the battery was very small. Cap2 didn't change and
SoC2 = 100 * Ah2 / Cap2

Other things like cell balancing may occur during the pause but I believe the primary purpose of the pause is to measure Voc and compute the true SoC of the battery.

After 15 minutes charging resumed and both Ah1 and Ah2 increased by Coulomb counting as before the pause. The BMU also computed the SoCs as before:
SoC1 = 100 * Ah1 / Cap1

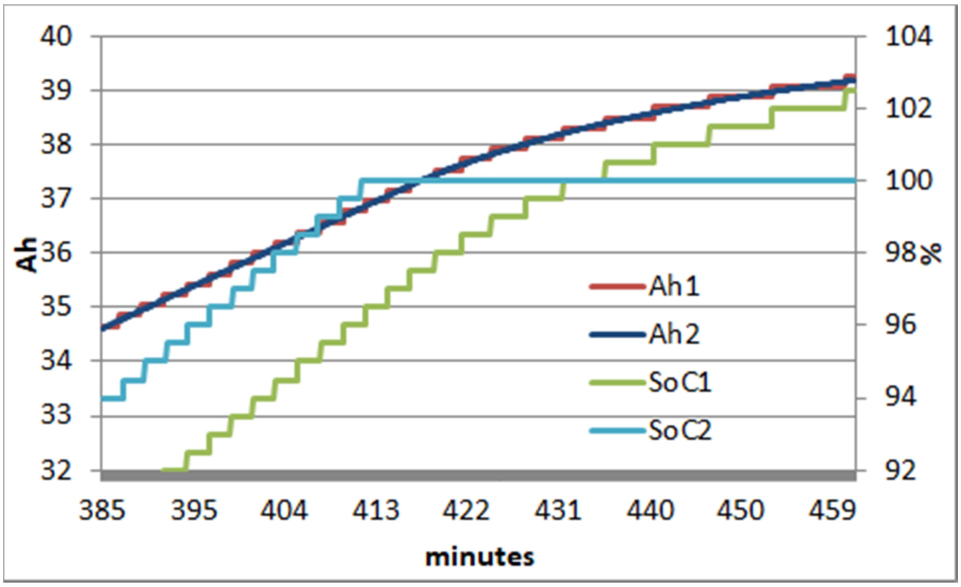
SoC2 = 100 * Ah2 / Cap2



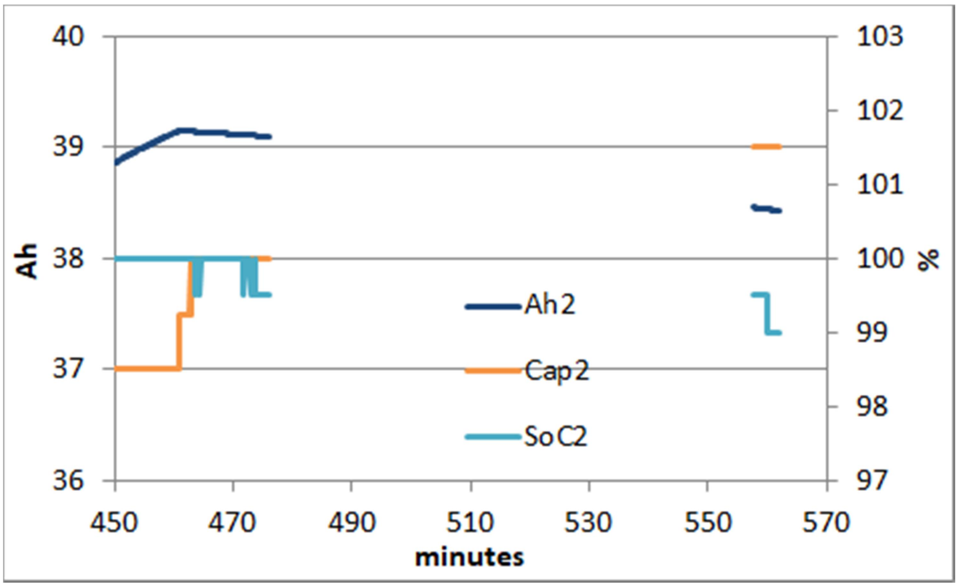

At 379 minutes shown in the graph above SoC1 was 89%. Both Ah2 and Cap2 were updated. Cap2 decreased from 38 Ah to 37 Ah and Ah2 was set equal to Ah1
Then SoC2 was updated according to the new values for Ah2 and Cap2.
SoC2 = 100 *Ah2 / Cap2
In this case SoC2, the light blue line, increased from 88 to 92.5%

At the same time, there were no changes in Cap1, SoC1 or Ah1.
This change always occurs when SoC1 is 89% during a slow charge. However, the changes in SoC2 and Cap2 are often too small to see.

After that, Ah1 and Ah2 were equal and both increased by Coulomb counting. SoC1 and SoC2 were computed by the BMU according to:
SoC1 = 100 * Ah1 / Cap1
SoC2 = 100 * Ah2 / Cap2

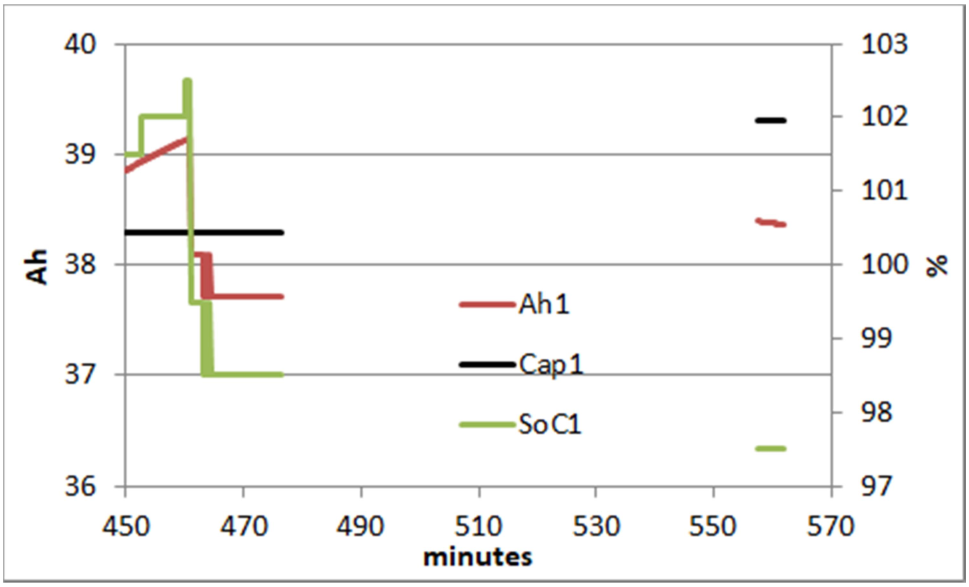


When SoC2 reached 100% at 461 minutes, it stopped increasing but Ah2 continued to increase due to Coulomb counting. Ah1 also increased by Coulomb counting and Ah1 and Ah2 were still equal. SoC1 also continued to increase beyond 100% according to
SoC1 = 100 * Ah1/Cap1.




When balancing of the cells finished at 461 minutes, charging stopped. However the capacity measurement wasn't complete. All PIDs continued to flow on the CAN network for another 15 minutes. Ah2 decreased due to Coulomb counting but only slightly because the current from the battery was very small. During this time the BMU was busy completing the measurement. Among other things, Cap2 increased slowly from 37 to 39 Ah in order to match Ah2. I don't know why Cap2 didn't change immediately to Ah2 but instead increased slowly to Ah2 over more than 15 minutes. The change in Cap2 occurred before data stopped but my app didn't record this change until 558 minutes, when I turned the car on and full data transmission resumed. By this time Ah2 and SoC2 had also been updated and
SoC2 = 100 * Ah2 / Cap2

During the period from 476 minutes and 558 minutes there were only 4 PIDs transmitted on the network at 1 minute intervals. These didn't contain data related to the charging process. The reason for the 82 minute gap was that full data transmission stopped at 6 o'clock in the morning. It took me 82 minutes to wake up, put on some clothes, drink a cup of tea and then go out to the car and turn it on.  When I turned on the car, all PIDs became available on the CAN network again.




This graph shows Ah1, SoC1 and Cap1 over the same period as the previous graph. During the 15 min. after charging at 461 minutes and until data transmission stopped at 476 minutes the current was about -0.5 amps, the BMU measured the Voc and computed SoC1 according to:
SoC1 = c*Voc + d
Like a and b above c and d are specific to the LEV50(N) cell and I don't know what they are yet.
Cap1 doesn't change and Ah1 is computed from SoC1 and Cap1.
Ah1 = SoC1 *Cap1 /100

When full data transmission restarted at 558 minutes, both Cap1 and Ah1 had been updated. Cap1 increased from 38.3 Ah to 39.3 Ah, a 1 Ah hop.

How the new **capacity was computed**
I believe the measured capacity was computed based on SoC1 and Ah2 at the end of the pause at 100% shown above and Ah1 and SoC1 at the end of the 30% pause shown in the second graph from the top. This next equation is the simplest way to compute the capacity:

Capacity = 100 * (Ah2_100 – Ah1_30) / ( SoC1_100 – SoC1_30)
Capacity = 100 * (39.1 – 11.3) / (99.5 - 29.5) = 39.7 Ah

The reason Ah2_100 was used rather than Ah1_100 is that Ah2 _100 minus Ah1_30 was the actual number of Ah added between the SoC 30% and 100%. At the same time SoC1_100 – SoC1_30 was the true change in the SoC between 30% and 100%. Dividing the true change in the Ah with the true change in the SoC and multiplying by 100 gives 39.7 Ah, the measured capacity. This is greater than 39.3 Ah, Cap1 after being updated. I believe the BMU limits increases in the capacity to 1 Ah. Therefore Cap1 only increased from 38.3 to 39.3. This limit prevents possible errors in the measurement from causing wild changes in the capacity.

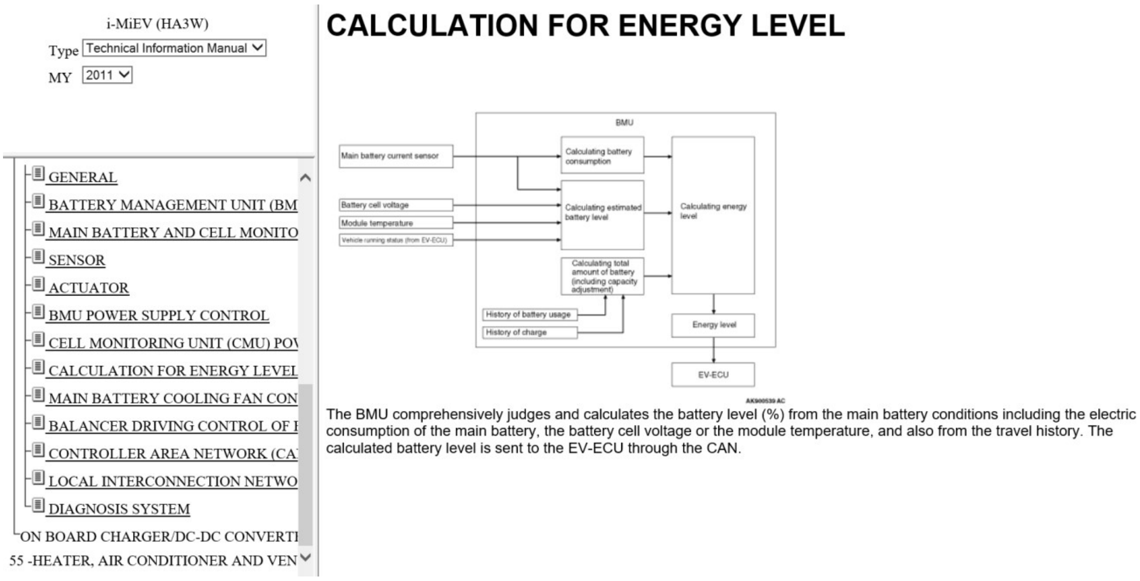
After 476 minutes, the measurement was complete. From then on, Ah1 and Ah2 were equal and they decreased by Coulomb counting. SoC1 and SoC2 were computed according to:
SoC1 = 100 * Ah1 / Cap1
SoC2 = 100 * Ah2 / Cap2

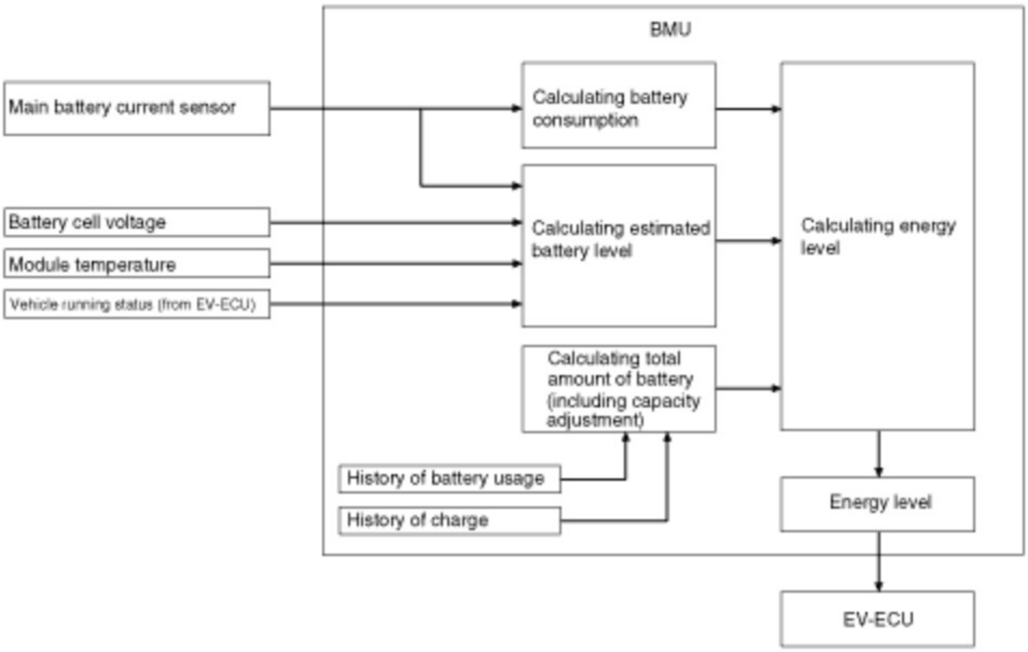
I don't know why this process is so complicated. If the only objective was to measure the capacity then it could be much simpler. One secondary objective could be to have a SoC, in this case SoC2, that doesn't exceed 100%. This then would be the SoC displayed on the left hand side of the instrument panel.

## Annex 6. The iMiev/CZero/iOn method for computing the capacity of the main battery

The capacity of a lithium ion battery powering an electric car deteriorates over time and with use. Therefore the Battery Management Unit (BMU) of an electric car must be able to measure the present fully charged capacity (just capacity from now on) of the car's battery. Predicting the car's range depends on an accurate measurement of the capacity and the actual state of charge (SoC).

This diagram from the iMiev Technical Information Manuel outlines the iMiev method of measuring the capacity and the SoC. This method also applies to the iOn and CZero cars.



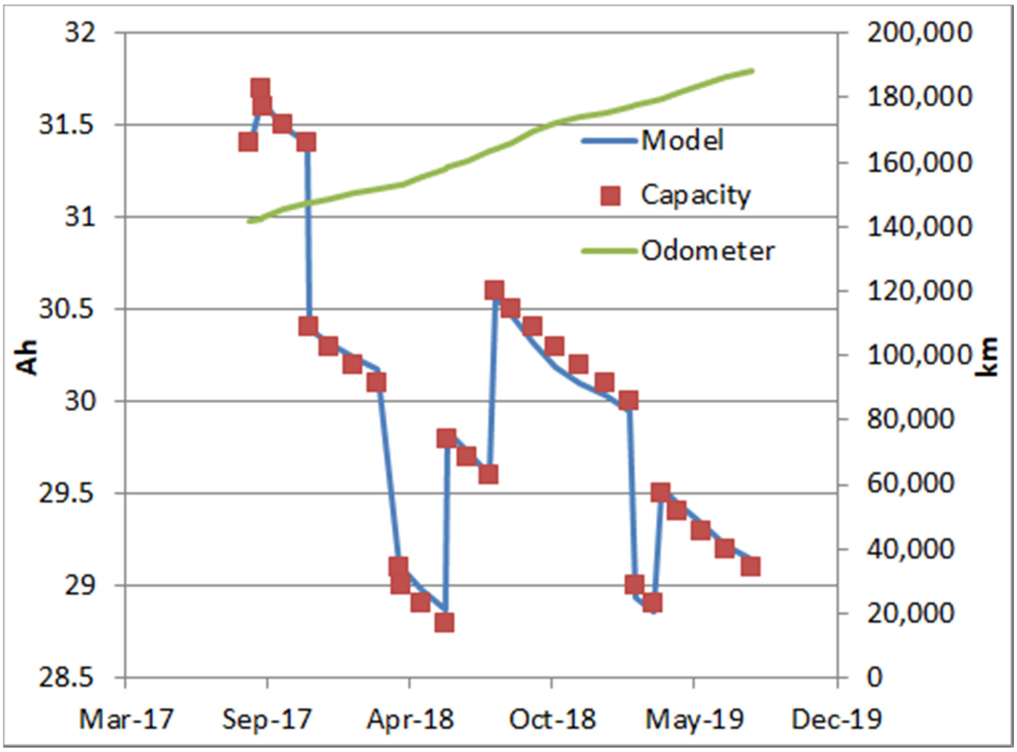




The diagram shows three calculations feeding information to a calculation of the energy level (SoC) in the battery. The manual contains no further information on how these calculations are performed.

The top calculation “Calculating battery consumption” shown in the diagram receives a signal from the main battery current sensor. This system probably keeps an account in ampere-hours (Ah) of the current to and from the battery. This is called coulomb counting.

Capacity is calculated by the bottom calculation named “Calculating total amount of battery” (CTAB). The following is an analysis of this capacity calculation based on data collected by two of us, joloeber and iOnier, both contributors to the German GoingElectric forum.
https://www.goingelectric.de/forum/viewforum.php?f=222

It is important to keep in mind in the following, that we are reverse engineering a calculation of battery degradation programmed into the BMU by Mitsubishi engineers prior to the car leaving the factory. We are not calculating the true capacity of the battery.




joloeber iMiev 2011

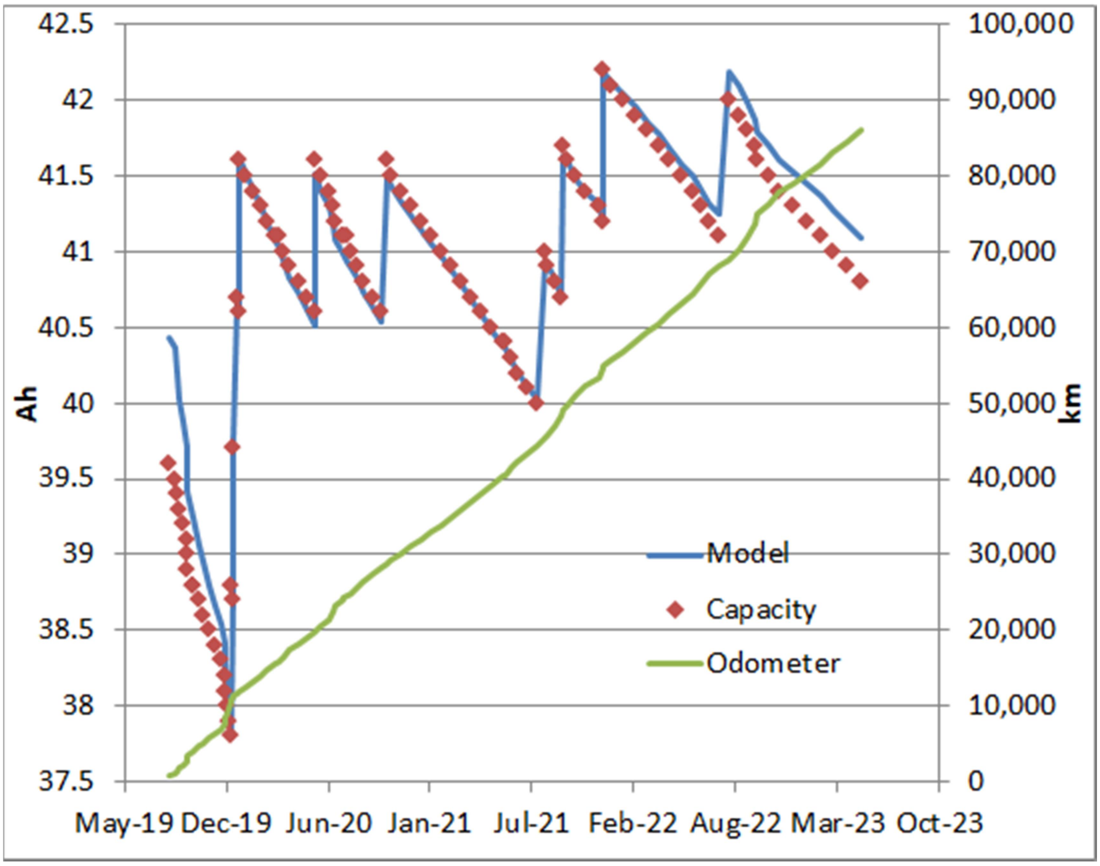
This first graph shows the data record by joloeber from his 2011 iMiev using the CaniOn app. For clarity the capacity is shown as a time series. However the change in the capacity follows the car’s mileage more closely than the age of the car. The model shown in the diagram is:

Ah = 48 - 0.0388 * sqrt(km) plus or minus 1 Ah corrections.

Note that this agrees with the input “History of battery usage” to the CTAB shown in the BMU diagram. The correction agrees with the “capacity adjustment” in the CTAB. We believe this capacity adjustment comes from the capacity measurement done while slow charging. This is the “History of charge” input.

The measurement done while slow charging, was described in another post. It requires the SoC to be less than 70% before charging and 100% after charging.

The next graph shows capacity data from joloeber’s second car, a 2015 iMiev. This was also collected using CaniOn.

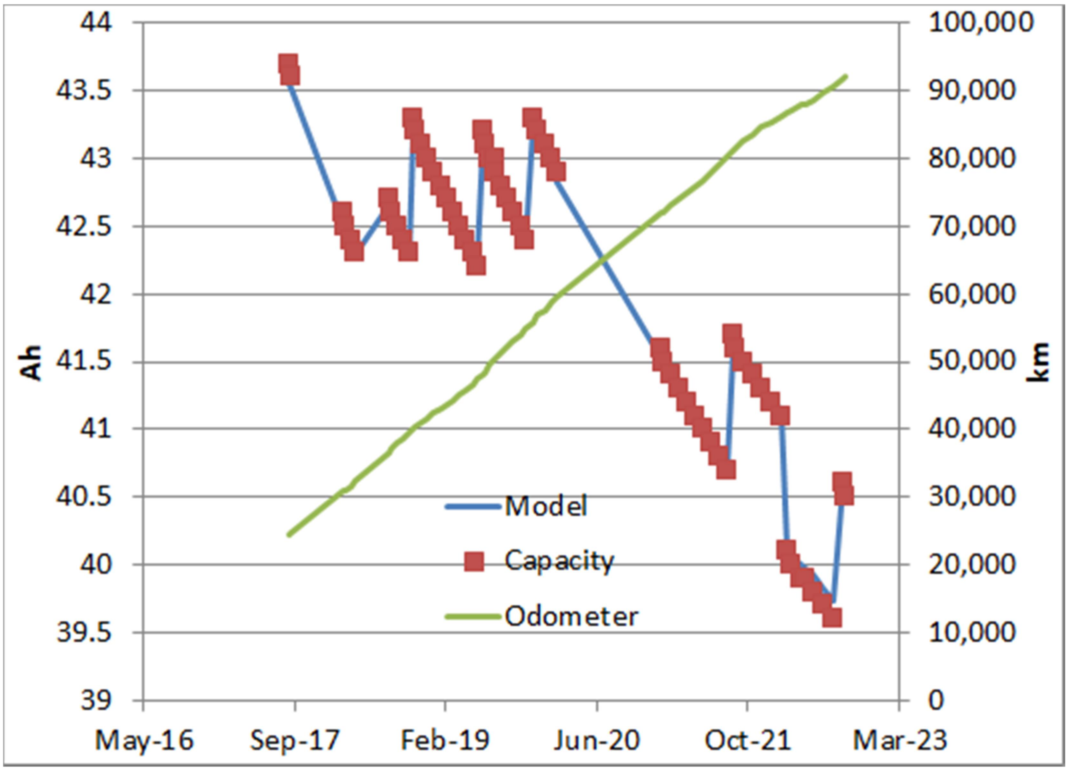



joloeber iMiev 2015

The model in this case is:
Ah = 48 – 0.0355 * sqrt(km) plus minus 1 Ah corrections

Note that there is poor agreement between the model and the capacity for the first series of data ending in mid-December 2019. This may be due to the long time between the build in 2015 and the first km driven in 2019. This requires further analysis. The factor 0.0355 is similar to the 0.0388 for the 2011 iMiev. Considering the accuracy of the models the factor could be the same for both cars.

Also interesting is the rapid correction of the capacity from 37.8 Ah to 41.5 Ah in the space of one month starting in mid-December. This occurred in 1 Ah steps, the largest correction permitted by the BMU calculation.



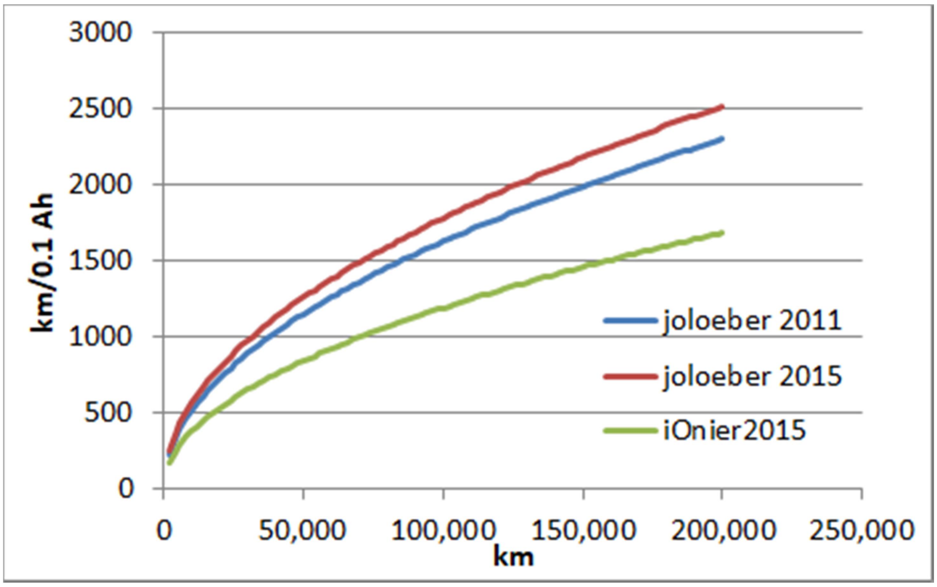
iOnier iOn 2015

This last data set is from iOnier's 2015 iOn collected using CaniOn. The model is:

Ah = 48 - 0.054 * sqrt(km) plus minus 1 Ah corrections

In this model the factor 0.054 is significantly larger than the factors for joloeber's cars. The difference is probably due to the difference between the km driven per day. IOnier drives an average of 38 km/d while joloeber drove 68 km/d in his 2011 car and drives 61 km/d in his 2015 car. (Datom, in his analysis of the data on the German GoingElectric forum, included km/d as a factor.) IOnier's factor is larger because the BMU also takes time and aging into account when computing the capacity. In other words there are more days per km for iOnier's car than joloeber's and the factor in this model for iOnier's car must be larger to account for this.

This means that the correct model should include both the age of the car and the number of km driven. We have not found this model yet. More data from cars with different km per day would help.



This graph shows the average number of km between 0.1 Ah downward steps of the capacity computed by the BMU. This graph was computed using the models show above.

The graph shows that iOnier's iOn loses 0.1 Ah after fewer km than joloeber's iMiev because it takes iOnier longer to drive those km. Again time plays a role in the BMU capacity calculation as well as km driven. It also shows that the Mitsubishi engineers that designed the system expected the rate of degradation to slow as the battery aged.

These observations have some interesting consequences. Of these the most important is that the battery capacity deterioration is in fact much slower than the Mitsubishi engineers expected at least for iMiev/iOn/CZeros in Germany. This is indicated by the frequent 1 Ah upward corrections in the capacity shown in the graphs above. The reason for this may be that the engineers had to account for cars operating at higher temperatures than the average in Germany.

Another important consequence can be seen in the data from joloeber's iMiev 2015. This car spent almost 3 years from its build date to its first use. During that time it may not have been charged properly such that the BMU could measure the capacity. However the BMU deterioration model was running all the time, steadily reducing the BMU's capacity estimate. As soon as joloeber began charging the car normally the BMU was able to correct its very low capacity estimate to a much more reasonable value.

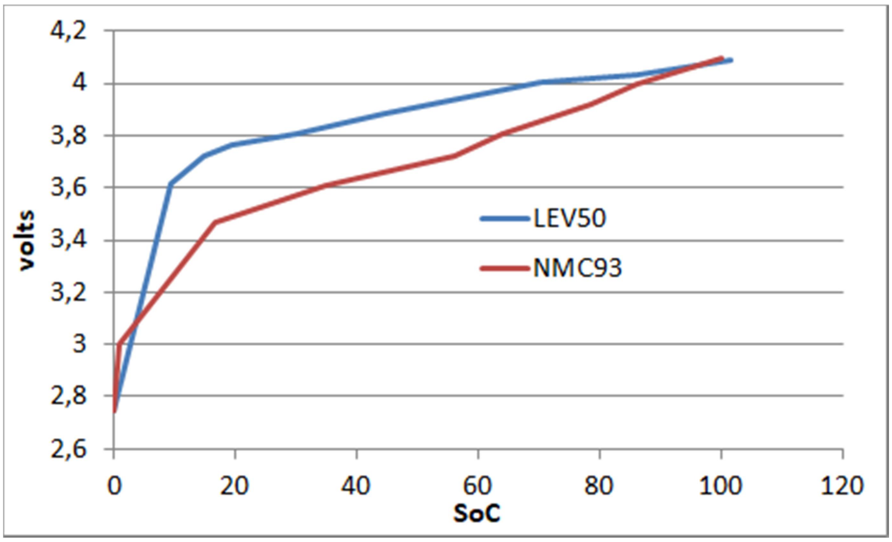
In conclusion it is necessary for the BMU calculation of the battery capacity that the car be slow-charged from less than 70% SoC to 100% on a regular basis. This insures that the BMU measures the capacity and can make the necessary adjustments to its preprogrammed capacity estimate. If the measurement isn't performed because e.g. the car isn't charged to 100% or the car is only charged via the Chademo port, the BMU capacity will slowly and regularly decrease until it is well below the true capacity of the battery.

joloeber, iOnier and OBDZero
May 2023

**Annex 7. The difference between the LEV50 and the NMC93 cell**

This graph below shows the low current voltage to state of charge (SoC) curves for the LEV50 and the NMC93 cells. The low current voltage is very close to the open circuit voltage (OCV). For all types of cells the OCV is determined by the SoC. Therefore if you know the OCV, you know the SoC.

Even though both cell types are lithium ion cells their chemistries are quite different. The LEV50 is a lithium magnesium oxide cell while the NMC93 is a lithium nickel magnesium cobalt cell (NMC). This is the reason for the difference between the two curves. This difference is important because the iMiev CZero iOn battery management unit (BMU) uses the low current voltage to determine the SoC of the cell or cells. This is an important step when measuring the present 100% capacity of the battery.




Graph: The blue line for the low current voltage of the LEV50 is based on data I collected from my CZero 2011. The data for red low current voltage curve was collected by piev (see the myiMiev forum).

A number of owners have replaced the LEV50 cells with NMC93 cells. They discovered that the BMU didn't measure the correct capacity for these new cells. This is because of the difference between the two curves in the graph. The 100% SoC voltage is 4.1 volts for both cells and so is the 0% SoC voltage, 2.75 volts, however at 3.75 volts the SoC is 30% in the LEV cells and 60% in the NMC cells.

From Annex 5 we know that the BMU computes the capacity based on the Ah charged to the battery between SoC 30% and 100% and it assumes that the SoC is 30% when the low current voltage is 3.75. This is the equation for the LEV50.

Capacity = 100 * (Ah2_100 – Ah1_30) / ( SoC1_100 – SoC1_30)

However for the NMC cell the BMU computation is in error because the SoC is in fact 60% when the voltage is 3.75. Here is the computation which results in a 100% capacity of a new NMC93 cell of about 49 Ah instead of 85.

48.6 Ah = 100*(85-51)/(100 -30)

Note 85 Ah and not 93 Ah because the NMC93 cell only has 93 Ah when charged to 4.25 volts. In the iMiev the cell is charged to 4.1 volts therefore its max capacity when new is about 85 Ah.