Politecnico di Torino

Master Thesis in Electronic Engineering

Analysis and development of Battery Management System for electric vehicles

Author: Floridia Stefano

Professor: Passerone Claudio

Supervisor: Leandro Malara

Academic year 2021/2022

Contents

Acronyms

ADC Analog to Digital Converter. 19

BMS Battery Management System. 5, 6, 9, 16, 18–21, 24, 46

BP Battery Pack. 5, 8, 9, 11, 12, 19, 21, 22, 30, 46, 47

CAN Controller Area Network. 7

CC Counting Coulomb. 26

CRC Cyclic Redundancy Check. 8

DLC Data Length Code. 8

DMA Direct Memory Access. 54, 55

EChM Electrochemical Model. 28, 29

ECM Electrical Cicruit Model. 28

ECU Electronic Control Unit. 6, 19, 21, 46

EV Electric Vehicle. 5, 6, 9, 11, 13, 23, 24, 30

GLVS Grounded Low Voltage System. 8

HEV Hybrid Electric Vehicle. 5, 6, 9, 24

HV High Voltage. 8, 9, 19, 21

HVIL High-Voltage Inter Lock. 9, 21

I2C Inter Integrated Circuit. 6

IDE Identifier extension. 8

IMD Isolation Monitoring Device. 8

LV Low Voltage. 9, 21, 22

MCU Microcontroller. 48

NTC Negative Temperature Coefficent. 19

OCV Open Circuit Voltage. 26–28

RMSE Root Mean Square Error. 39, 41, 43, 45, 46

SAR Successive Approximation Register. 19

SoC State of Charge. 5, 18, 20, 23–30, 32, 33, 39, 46, 58–60, 62

SoH State of Health. 20, 23, 24, 30, 62

SPI Serial Peripheral Interface. 6

UART Universal Asynchronous Receiver Transmitter. 6, 21, 50–52, 54, 56

VCU Vehicle Contol Unit. 6, 8, 21

List of Figures

Abstract

Our times are characterized by numerous electronic applications that are changing the lifestyle of many. There are devices designed for various situations and one of the biggest challenges is making sure these applications last as much as possible. For this purpose, rechargeable batteries are the best solution until now, guaranteeing a permanent and clean energy storage if we talk about the automotive industry. In fact, batteries are starting to be the most common electrical energy storage system for an Electric Vehicle (EV) or Hybrid Electric Vehicle (HEV). In Electric Vehicles, a Battery Pack can provide several hundred Volts, since various functions within a car depend on it, including powering the electric motor itself. The key aspect of all these applications is that the batteries must be constantly monitored, to have better energy consumption or to take safety measures.

A Battery Management System (BMS) is an integrated system, designed to closely monitor battery circuits, implementing protective measures in case the battery is operating outside its normal operating area, but also acquiring data that can be used for better power management. The State of Charge (SoC) is one of those data that must be tracked in a battery to optimize its performance since it is essential to know how much capacity is available.

The purpose of this thesis work is to describe a set of functions of the Battery Management System, with particular attention to the development of an algorithm for the evaluation of the State of Charge, but still having the possibility of being able to apply these functions to generic management systems of a battery in vehicles. The system used for data validation is the one developed for the Formula SAE "Squadra Corse PoliTo" prototype.

CHAPTER 1

Battery Management System

1.1 Introduction

Inside Electric Vehicles, the battery pack contains one or more ECUs that are able to perform for monitoring and protection for the vehicle and the user. This system is what is usually referred to as Battery Management System (BMS) made of a group of ECUs exchanging information in a Master-Slave fashion. The **Slave** system is a group of ECUs connected to each other forming a Daisy Chain and connected with the Master using standard digital protocols like UART, SPI, I2C. Its work is to perform data acquisition from the battery such as Voltage, Current and Temperature, and sometimes it is integrated with a circuitry for balancing. **Cell Balancing** is a very useful technique that can be implemented in order to increase the available capacity of the battery, extending its life. It is a typical solution for multi-cells systems including laptops, phones, HEV and EV.

The **Master** is the main ECU, this why the device is referred most of the time as the actual BMS, performing the most important algorithms according to the received data and it is connected to the majority of protection devices inside this circuitry. The BMS exchanges information with the vehicle, usually communicating with a Vehicle Contol Unit (VCU) that elaborates what is important to know about the Battery. The most common communication protocol implemented for information exchange in automotive, between ECUs,

is the Controller Area Network (CAN) [16]. It is a multi-master bus which has good noise immunity and allows for a large number of node connected, consisting of only a twisted-pair cable (differential communication) that is terminated on the two sieds by a 120 Ω resistances avoiding signal reflection that would bring further interference [20]. **Figure 1.1** provides an example of CAN network with 3 nodes.

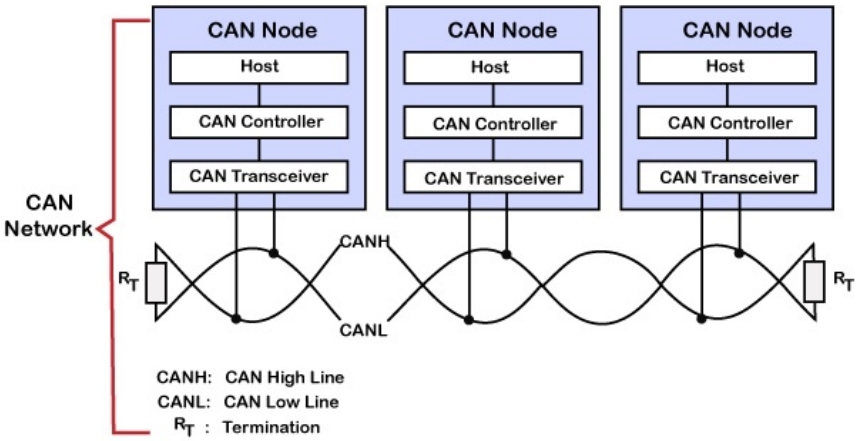


Figure 1.1: CAN bus topology

CAN protocol is defined in the ISO-11898 [4] as CSMA/CD+AMP, that is:

- **Carrier-Sense Multiple-Access**: every node can access the bus, waiting for a certain amount of time before trying to send a message;
- **Collision Detection** and **Arbitration on Message Priority**: each node can detect a collision between the actual bus state and the attempted transmission, by a predefined message priority.

The protocol consists of frames subdivided in fields, as shown in **Figure 1.2**, sent at a maximum speed rate of 1 Mbit/s.

- **Arbitration field**: The Identifier, whose length may be 11 or 29 bits, is the actual element that defines the priority of a message, the lower it is the higher is the priority. RTR is a 1-bit length for remote request, it is always '0' in data frames.

- **Control field**: this is the field where the length and the type of frame are defined. The IDE is 1-bit long, it defines whether the Identifier is:
  - **Standard**: 11 bit long
  - **Extended**: 29-bit long

  The Data Length Code is 4-bit long portion which defines the number of Bytes of the Data field

- **Data field**: It is the portion containing the actual data bytes to be sent, its length goes from 0 to 8 bytes

- **CRC and ACK fields**: CRC is used to detect data corruption during transmission. ACK is the receiver acknowledgment

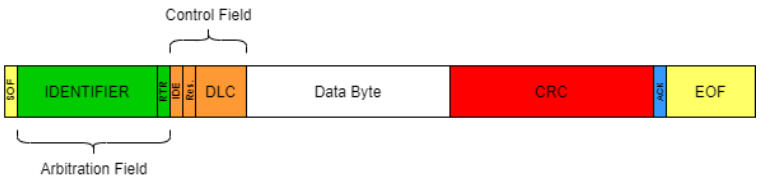


Figure 1.2: Standard CAN message frame

For protection purposes the High Voltage Battery circuit is implemented with devices that may be controlled by the BMS master.

- **Isolation Monitoring Device (IMD)** which make sure that the Galvanic Isolation between HV and the Grounded Low Voltage System (GLVS) is $\geq 500\Omega/V$ in AC and $\geq 100\Omega/V$ in DC. It can operate on its own and perform protection procedure or it can be connected to the Master sending a signal when the Battery Pack is no more safe.

- **HV Contactors**. These are switches that provide fast switching and there must be one for each terminal, Positive and Negative. In this way it is assured that the vehicle is switched off, whenever it is requested by the VCU, since the car is stopped, or because an hazard occurred. There's a third relay that is the one involved in the **precharge** procedure. Power supply going out from the Accumulator needs to go through an

Inverter in order to reach the Motors, since they work in AC. Inverter are provided with Capacitive elements that are used for energy storage. With High Voltage systems, inrush currents may stress a lot the capacitive components during power up. The solution adopted was to have a parallel channel in which a resistive component is connected limiting the amount of current delivered to the load.

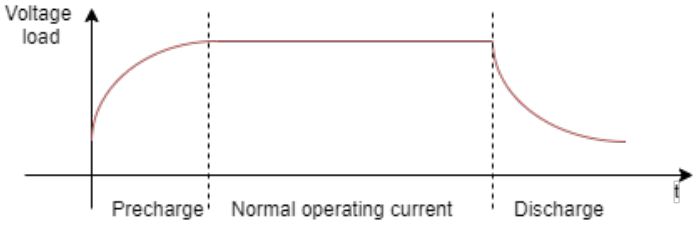


Figure 1.3: Battery Voltage characteristic

The outcome, illustrated in **Figure 1.3**, is that during power-up, the BMS closes the Negative and Precharge relay and the output Voltage increases slowly until inrush current has subsided. Then Positive relay is closed in order to provide the right amount of current during operating stage.

- **Fuses** that are used for cable management. They may be connected to both or one of the Terminals for over-current protection.
- **HVIL**. High-Voltage Inter Lock loop is a safety feature typical of Hybrid Electric Vehicles and Electric Vehicles. HVIL is a LV powered cable that inside a BP is connected between the HV connector and the BMS. If the connection is interrupted the BMS goeas in safe mode, opening every contactors in order to cut-off any kind of supply. This is a very important feature that come in handy during maintainance on High Voltage components or in situation like a high-voltage connection becomes loose, disconnected or damaged during the operation of the vehicle.

Figure 1.4 shows a simplified architecture of the circuitry embedded in a Battery Pack, including those elements enlisted previously.

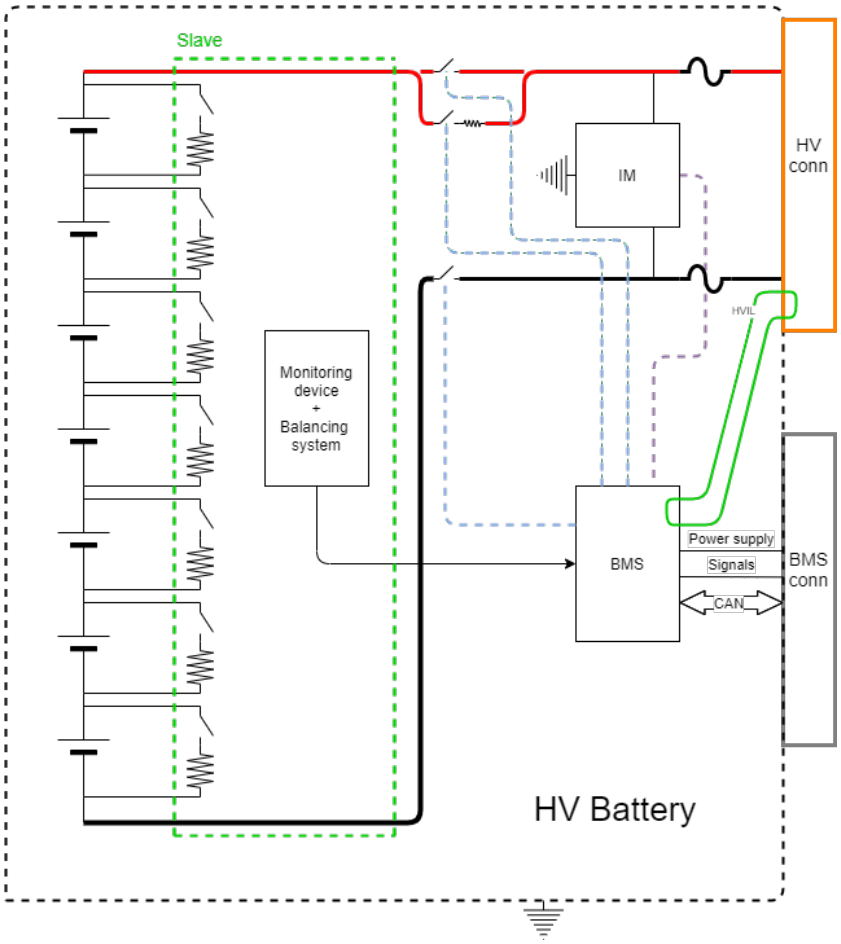


Figure 1.4: Battery schematic

1.2 Accumulator design

Before going on talking about the functionalities of a BMS, it is important to know something about batteries. A Battery is a device that transforms chemical energy into electrical energy and it is rechargeable. The basic unit inside Battery for Electric Vehicles is a cell; to obtain proper voltages and currents, multiple cells are connected following the $xSyP$ **configuration**.

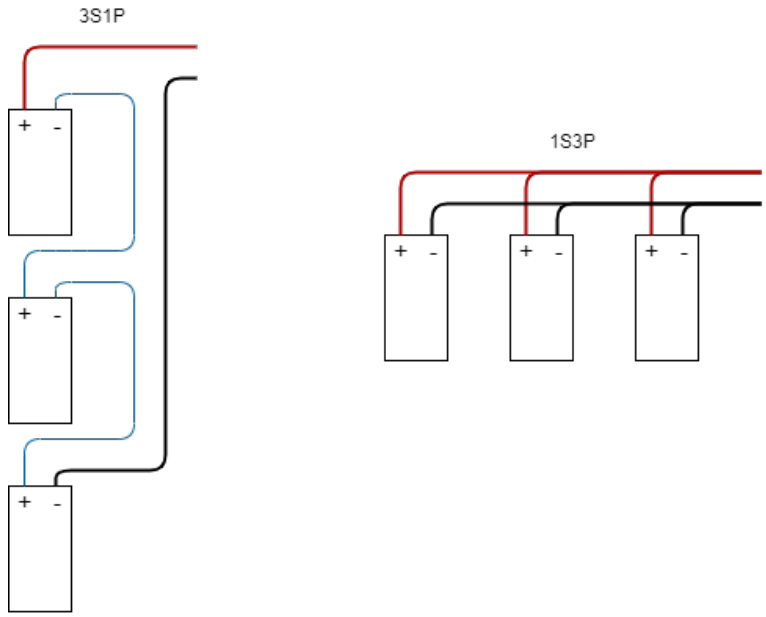


Figure 1.5: Parallel (on the right) and Series (on the left) configurations

Looking at **Figure** 1.5, in parallel configuration the Output voltage is still the same while the Capacity increases. The opposite occurs with series configuration. Usually a set of cells in parallel is called **Group** connected in series forming a **Module**. A set of module in series assemble a **Battery Pack**. This is important for the Design since whichever it is the number of parallel cells in a group or the number of groups in series in a module, the amount of Energy a Module can deliver can change. $\mathrm{E}_{module} = gV_{cell} \times nC_{cell}$ where $g =$ **groups in series in a module** and $n =$ **cells in parallel in a group**. If you take into account that a Battery Pack contains m number of modules, the amount of Energy delivered is $\mathrm{E}_{BP} = m\mathrm{E}_{module}$.

So a Battery Pack having $xSyP$ configuration means that $x = g \times m$ and $y = n$. The example shown in **Figure 1.6** [22] is a single Module of a Tesla Model S which has a configuration of $6s74p$. It is usually covered by a metallic plate, typical for accumulator designs, called **Bus Bar** and it is the element working as connecting node between two groups in series.

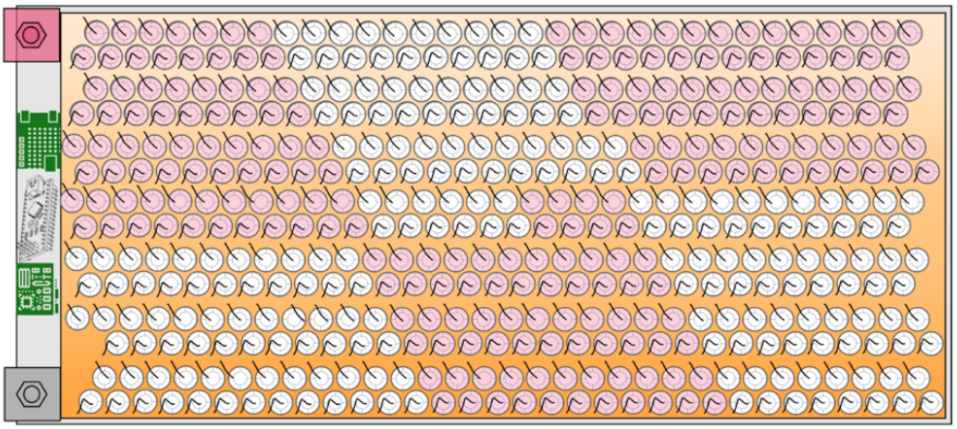
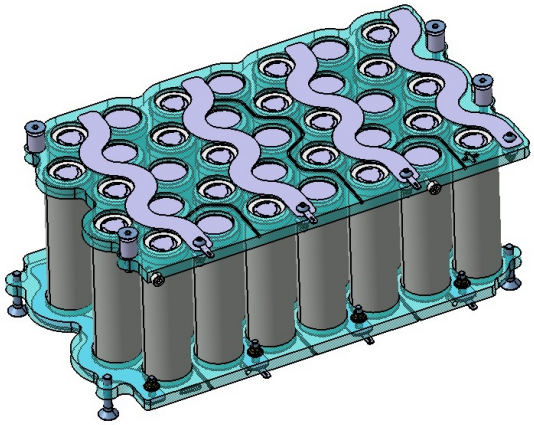


Figure 1.6: Tesla Model S Module: 6 groups containing 74 cells in parallel

Single rechargeable cells must be chosen taking into account different main features

- **Capacity** [Ah]: the amount of current deliverable by the cell completely charged
- **Gravimetric Energy density** [Wh/Kg]: the amount of electric energy stored per unit mass
- **Power density** [w/Kg]: the amount of stored power per unit mass
- **Nominal Voltage** [V]: Voltage of a fully charged cell
- **number of Cycles**: number of charge-discharge full cycles before the cell starts to deteriorate
- **recharge time**: the lower, the better
- **cheap, safe and reliable**

Lithium ions cells are the most employed for low power solutions like smartphones and lately their appearance is common when designing accumulators for EVs.



ADVANTAGES	DRAWBACKS
High Energy density	Variable performances depending on the quality of production
High Power density	Variable Safety depending on the cathode
High number of cycles (500 - 1000)	Variable Costs depending on the cathode

Figure 1.7: Li-ion cells features

The Anode of a Li-ion cell is made of Carbon while the cathode is a Li-metal Oxide, which means that, whatever the metal chosen for the Cathode, the cell performance, costs and safety features may change as shown in **Figure 1.8** [2]

	$LiCoO_2$	$LiMn_2O_2$	$LiFePO_4$
Nominal Voltage [V]	3.9	3.7	3.3
Max charge Voltage [V]	4.2	4.2	3.6
Number of cycles	500	500-1000	1000-2000
Energy density [Wh/Kg]	155	100-120	160
Thermal runaway [°C]	150	250	270
Safety	Low	-	Good
Cost	High	Low	-

Figure 1.8: Li-ion batteries comparison

There exist some alternatives to Li-ion cells that have been exploited, some of them are still used, like **Lead Acid** batteries. They are often used as auxiliary batteries for vehicles, in order to supply low voltage circuitry, but not involved in the traction system. Their Energy density is low and the Nominal Voltage is usually around 2V, nevertheless they are very durable and reliable. **Nickel** batteries are another example, their Energy density is higher then the previous one, but not as much as Li-ion batteries. Less durable than Lead-acid batteries but their charge time is very low. Nickel-Cadmium batteries are now out of production due to their toxicity. Li-ion batteries, compared with all these alternatives, are far better in terms of Energy density, since it is possible to deliver the same amount of Energy with far less unit mass, as shown in **Figure 1.9** [22], saving space and improving the vehicle design.

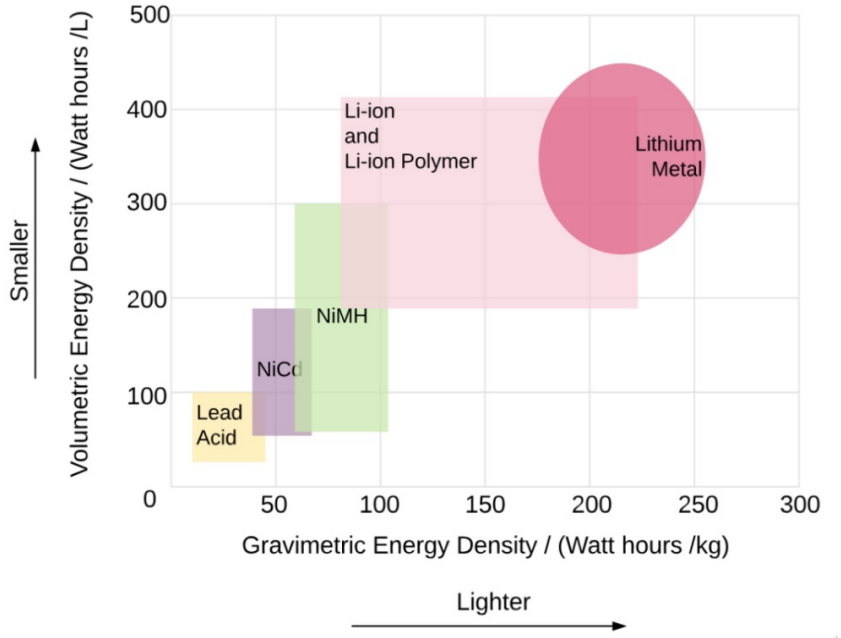


Figure 1.9: Cell comparison

Nevertheless Li-ion cells may be very dangerous if not monitored correctly. For multi-cells systems, even if cells are all the same in terms of the product, they have differences. Some cells may have different reaction to temperature changing, some of them may discharge faster or charge slower, all of this leads to cells having not equal output voltages. Let's consider Lead-Acid batteries, as already said they are very durable and strong, so in order to completely charge the battery, even if some cells reach a full state of charge, it possible to wait for other cells to reach the same condition, given that the disequilibrium is not so wide. For Li-ion cells this kind of solution may lead to thermal runaway damaging the battery. The same is valid in case of discharging of operation, if a cell goes far beyond the lower limit the reaction is far more dangerous, bringing the cell to short circuit [2]. The issue concerns a lot for elements connected in series, for cells in parallel inside a group tend to **self balance** themselves having the same voltage drop, exchanging charge.

Cell Balancing is a solution implemented by the BMS, monitoring cells during charge and discharge operations, equalizing the charge reaching as much as possible the maximum battery capacity, avoiding cell damage and extending the battery life. Cell balancing methods have been classified in two classes:

- **Passive balancing**: it is the easiest method to implement that commonly works on cell voltage indicator to its status of charge. The classic method is based on discharging cells dissipating charge upon a load placed in parallel. There are systems stopping the charging operation, discharging the cells with higher voltage until it is equalized with weaker cells. There are also systems that are able to do that without stopping the charge, until all cells are fully charged, bypassing the cell with higher voltage or discharging them on a load. These common methods are very wasteful, since it implies energy dissipation, and reaches balance after a number of cycles, due to the fact that voltage is not a good indicator for defining how charged is a cell since they may have same capacity but different voltage drop, because of different internal resistance. **Figure 1.10** shows a typical example of passive balancing. When the controller measures a cell voltage higher then the others, it opens the MOSFET connecting the cell to the resistance in parallel, called bleeding resistors [21].

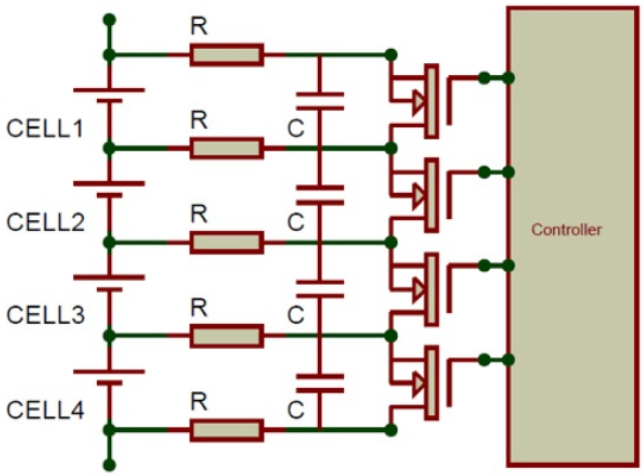


Figure 1.10: Passive cell balancing

- **Active balancing**: it implements capacitive, inductive circuits or dc/dc converters for charge exchange between cells. Far more efficient then the previous one, shifting energy where is needed with infinitesimal dissipation. Capacitors and inductors work as charge storage and of course it is very time consuming since charge is distributed between cells sequentially. Since it needs more components than a simple load it's more expensive and harder to implement. There are a variety of methods based on the concept just introduced. **Figure 1.11** shows a simple schematic of one of the most common technique called Flying capacitors.

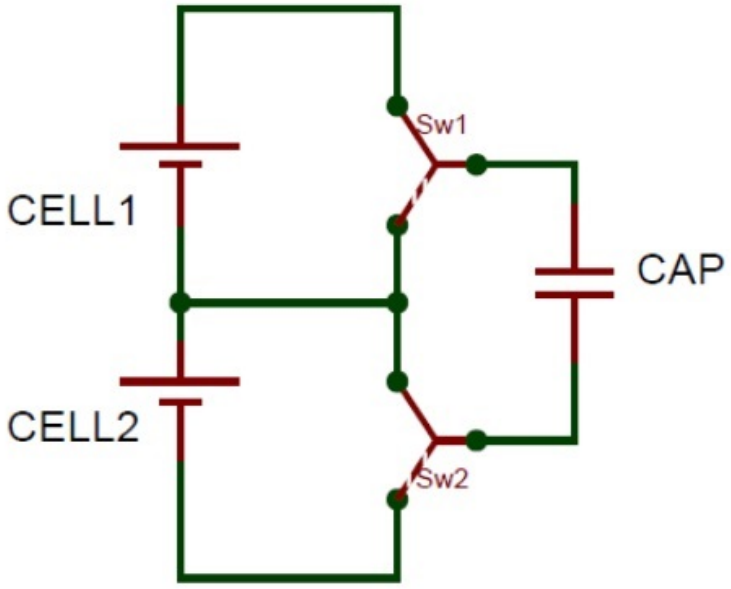


Figure 1.11: Active cell balancing

The capacitive elements are the one that are physically shifting energy from one cell to the next one, this implies a sequential action between adjacent cells, very time consuming considering also the time needed for the capacitor to be charged. This circuit have been improved in terms of spent time and efficiency having different configuration involving DC-DC converters like buck-boost or fly-back converters, but increasing a lot the circuitry complexity. Nowadays software controlled solution tried to solve the problem aiming to reduce Hardware usage, gaining in space and complexity [14].

The idea is based on cell bypassing having as decision criterion the SoC of the single cell calculated by the BMS. While charging, cells that reaches full charge are bypassed, letting other cells to join. When discharging cells with low SoC are disconnected, having minimal loss since those cells have very low capacity.

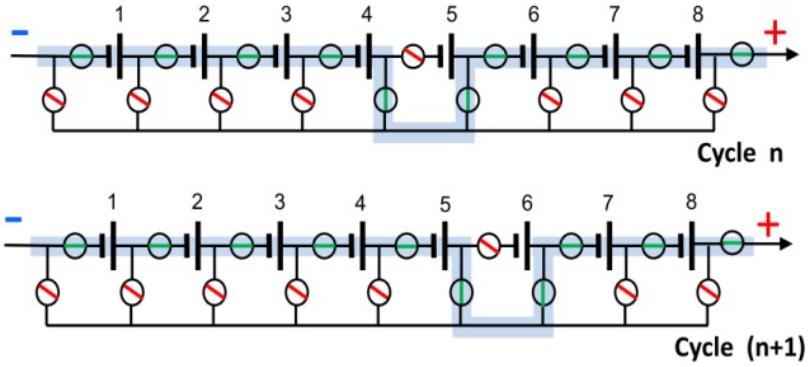


Figure 1.12: Software controlled balancing schematic

1.3 BMS functionalities

The main functions of a BMS are:

- **Battery monitoring**: the ECU is constantly controlling those features of a battery that are very important in order to know in which state the accumulator is. Voltage, current and temperature is what the Master needs in order to perform its calculations. For this thesis work, the slave system has an ECU using the $bq76pl455$ micro-controller, which is able to measure up to 16 cell connected in series and it is possible to connect up to 16 devices in Daisy chain. The acquisition is driven by a 14-bit Successive Approximation Register (SAR) ADC, then the value is stored into a 16 bit register ready to be sent to the master [16]. The same acquisition system is used for temperature measurement, with the slight difference that this time the micro can handle only 8 analog inputs, that are designed as auxiliary channels. NTC thermistor are placed in strategic positions inside the module or the Battery Pack in order to observe better temperature mutations. The NTC works as a resistance which decreases its value exponentially as temperature increases. Mathematically speaking this behaviour is described as

$$R_{NTC} = R_0 e^{\beta(\frac{1}{T} - \frac{1}{T_0})} \qquad with \qquad R_0 = R(@T_0 = 25C)$$

  The analog value measured by the ADC is the voltage drop on the NTC itself. The simpler way to connect the NTC is to use a pull-up or pull-down resistors, but there are Hardware solutions that may help forcing a linear relation between temperature and voltage acquired by the slave. Current sensing is usually done exploiting Faraday's law of electromagnetic induction, as illustrated in **Figure 1.13** [15]. These kind of sensors are designed to have a circular element, in which the magnetic field is induced, looping around the HV cable and having two terminals forcing a voltage upon a plate, called Hall plate, connected to an operational amplifier whose output voltage is the one read directly by the Master for data managing.

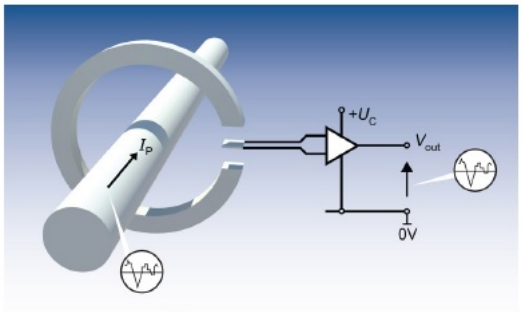


Figure 1.13: Current sensor with Hall effect

- **Battery protection and optimization**: Smart battery usage is a key feature a Battery Management System in order to keep the battery vehicle alive as much time as possible. As already said battery cells, especially Li-ion cells, are very dangerous if not closely monitored risking reaction that can damage the battery, the vehicle and therefore the user. Battery must perform safety action when over-current, over-voltage or over-temperature occurs, but it should be far better if it is possible to prevent those events keeping an eye on the status of the battery. The same data monitored is useful for battery status data evaluation like SoC or SoH, which are good parameters for knowing how much charge can be delivered to single cells, how much capacity the battery is still able to guarantee or how old is the battery in terms of number of operative cycles, since cells performance degrades with it. Usually a BMS should keep the battery to work inside an operative range where damage possibility is lower, moreover it is common practice that battery manufacturers provide data sheets with operative range smaller then the actual capacity of the cell in order to reduce risks further more [1] [9].

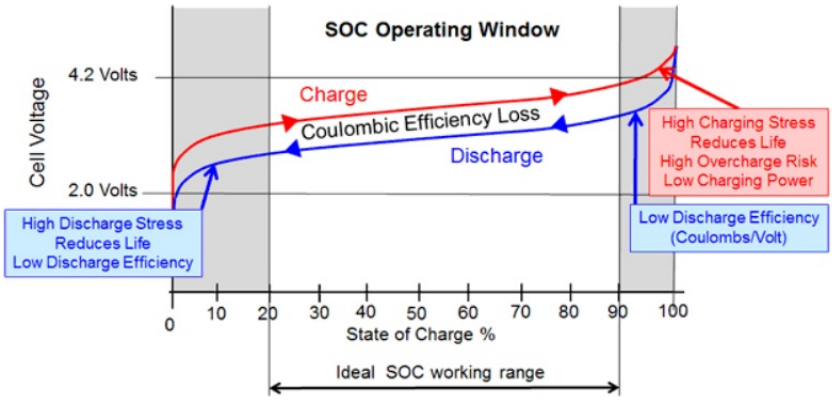


Figure 1.14: Cell operative range

With a good evaluation of cell status, the balancing operation is definitely improved letting the master to equalize the working range to each group of cells connected in series.

- **Communication**: continuous exchange of information is vital for a BMS, communicating to the vehicle useful data for the VCU or the user. For example alerting when the battery charge is complete or it needs charge. It is also helpful for diagnosis purposes, since the master is able to distinguish different kind of hazard, like over-current or HV connector not placed correctly with HVIL open, so whenever a problem occurs it is possible to discover what caused that faulty behaviour. CAN bus is the most used for Automotive purposes due to its robustness to noise and ability to cover long distances with very low probable data communication errors, mostly used in order to connect ECUs belonging to different subsystems inside the vehicle. Inside the battery circuitry, simple communication signals, belonging to the Low Voltage system, are very often hardwired like supply signals, the HVIL presence or contactors status (open/closed). More complex information exchange, like measured cell voltages and BP temperatures or cell balancing commands, are managed by digital communication protocols like UART.

For this thesis work the system implemented, shown in **Figure 1.15**, is far simpler then the one shown before (see §**1.4**), also because the Battery Pack monitored this time is a Low Voltage accumulator (output voltage $< 60V_{DC}$, according to ISO-6469 [5]), so some of those system protection components and strategies presented previously are not needed.

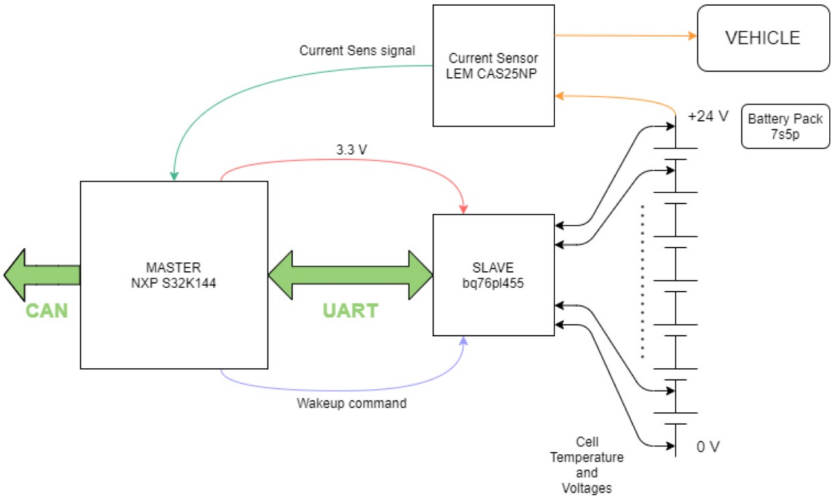


Figure 1.15: Low voltage battery management system

CHAPTER 2

State of Charge

2.1 Definition

State of Charge (SoC) represents the available battery capacity with respect to its rated capacity. The preferred SoC reference should be the rated capacity of a new cell rather than the current capacity of the cell. This is because the cell capacity gradually reduces as the cell ages. For example, towards the end of the cell's life its actual capacity will be approaching only 80% of its rated capacity and in this case, even if the cell were fully charged, its SoC would only be 80% of its rated capacity [10]. State of Charge (SoC) is one of the most important states that need to be monitored to optimize the performance and extend the lifetime of batteries, its evaluation is not direct as it involves to take under consideration voltage, current and temperature of a cell. In EVs, the number of battery has series-parallel combinations to match the load requirement [7]. Due to manufacturing procedures, not all cells simultaneously attain full voltage during charging resulting in voltage imbalance among different cells and, consequently, lower capacity from the entire battery. Knowing the amount of energy that a battery can deliver is vital for EVs applications requiring to constantly monitor the actual capacity of the cell in order to evaluate the actual and practical capability of the battery in order to fit the different road conditions and driving patterns of the vehicle. Another parameter important to evaluate is the **State of Health (SoH)** whose purpose is to provide an indica-

tion of the performance which can be expected from the battery in its current condition or to provide an indication of the how much of the useful lifetime of the battery has been consumed and how much remains before it must be replaced. In critical applications such as standby and emergency power plant the SoC gives an indication of whether a battery will be able to support the load when called upon to do so. Knowledge of the SoH will also help the engineers to anticipate problems to make fault diagnosis or to plan replacement. This is essentially a monitoring function tracking the long term changes in the battery [11]. A BMS must be not only a protection circuit but also a thorough and accurate device that can predict the SoC, SoH, capacity, and available power to increase the efficiency and the safety of the battery. By continuously measuring current, voltage, and temperature in batteries, the aforementioned parameters can be estimated. The estimation of the SoC is key in a BMS, but its online and accurate estimation remains a challenge due to strong nonlinear and complex electrochemical reactions in the battery and because of battery characteristics change with aging [3] [13]. There is a dependency between the actual capacity of a cell and the rate at which the battery is charged and discharged, due to the fact that a cell's internal electrochemical reactions take some time, this means that the battery cannot easily follow instantaneous charge-discharge pulses, as it happens in EV and HEV. The consequence of this is that when charging is complete and a load is applied to the battery to discharge it, there is a delay before the full current can be delivered through the load. This phenomenon is called **Hysteresis** and leads to SoC evaluation errors, even for direct methods like Counting Coulomb, that involves the monitoring of instantaneous capacity. There's still the possibility to minimize this error taking in account the behaviour, implicating more complex calculation [10].

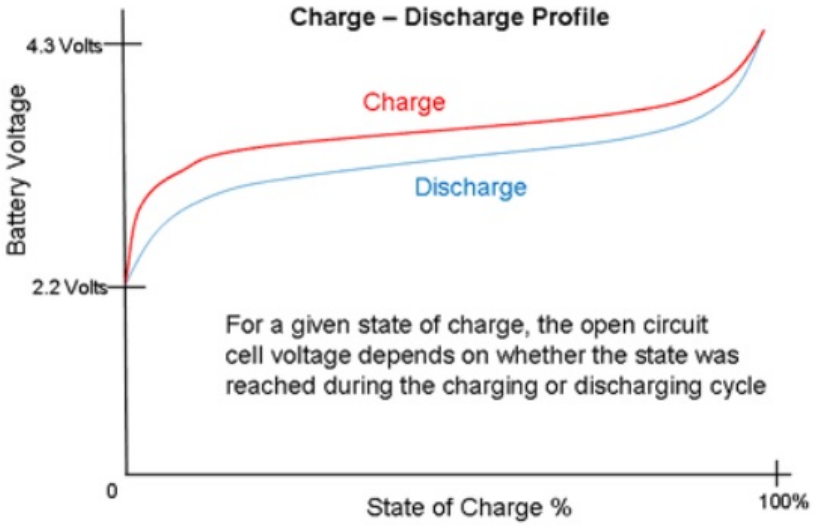


Figure 2.1: Cell hysteresis in Open Circuit Voltage condition

2.2 SoC Evaluation methods

In the following pages will be described some State of Charge (SoC) evaluation methods, showing main processes, advantages and drawbacks. Evaluation methods are usually classified into **Direct methods** and **Indirect methods**. As their name suggests, Direct SoC estimation methods use and measure physical battery features such as voltage, current, and temperature, and then manage them using mathematical equations. Indirect methods are based on finding a battery model, using battery physical properties, that can help capturing cell characteristic and predict its behaviour [13].

2.2.1 Coulomb Counting

The energy contained in an electric charge is measured in Coulombs and is equal to the integral over time of the current which delivered the charge. The remaining capacity in a cell can be calculated by measuring the current entering (charging) or leaving (discharging) the cells and integrating (accumulating) this over time . In other words the charge transferred in or out of the cell is obtained by accumulating the current drain over time. The calibration reference point is a fully charged cell, not an empty cell, and the SOC is obtained by subtracting

the net charge flow from the charge in a fully charged cell. This method, known as Counting Coulomb (CC), provides higher accuracy than most other SoC measurements since it measures the charge flow directly. CC depends on the current flowing from the battery into external circuits since in some applications such as automotive batteries the "continuous" battery current is not monitored. Instead the current is sampled and the continuous current is reconstructed from the samples. In such cases the sampling rate must be fast enough to capture the current peaks and troughs associated with the acceleration and regenerative braking corresponding to the user's driving style [10]. At present, the Counting Coulomb method is the most used for SoC estimation since it is the most accurate technique for short-term calculations. The CC method defines SoC as

$$SoC(t) = SoC(t_0) + \frac{100\%}{C_{rated}} \int_{t_0}^{t_0+\tau} I_{bat} \delta\tau$$

where SoC(t0) is the initial SoC, C_{rated} the nominal capacity, and I_{bat} is the charging/discharging current. This method cannot estimate the initial value, so the initial SoC must be known, and has an accumulative error. As it is an open loop estimation system, small errors will accumulate with time due to the integration term, being a source of significant inaccuracy. Several factors affect its accuracy, such as battery age, discharge rate, and sensor precision [13].

2.2.2 Open Circuit Voltage method

Open Circuit Voltage (OCV) can be defined as the voltage drop on the terminals of a battery when no load is connected. SoC estimation methods commonly impose a characterization of the OCV curve derived through a polynomial or a look-up table, as they use either a direct OCV curve inversion method, or a cell model-based methods. SoC is therefore determined as function of the OCV which relation, shown by the curve in **Figure 2.2**, is defined as:

$$SoC = f^{-1}(OCV)$$

In the OCV method, the cell's voltage is continuously measured, and the corresponding SoC is obtained from a table. The method is not practical due to the requirement of measuring OCV which means SoC estimation is not available

while the battery is charging or discharging thus it cannot be used in real time. Also, the OCV–SoC relationship differs among cells and, therefore results in unacceptable error [13]. Problems can occur with some cell chemistry, particularly Lithium which exhibits only a very small change in voltage over most of the charge/discharge cycle. The following graph shows the discharge curve for a Lithium-ion cell.

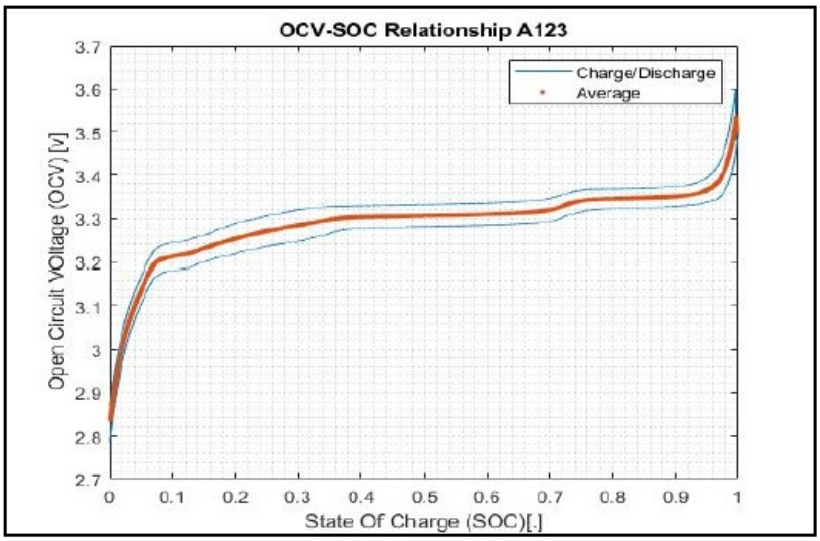


Figure 2.2: OCV SoC curve [12]

As you can see the cell voltage does not fall rapidly as the cell is discharged, but for the same reason, the actual cell voltage is not a good measure of the SoC of the cell, since the smallest error in the OCV obtained from a battery model can lead to divergence of SoC from the actual value. The rapid fall in cell voltage at the end of the cycle could be used as an indication of imminent, complete discharge of the battery, but for many applications an earlier warning is required [10].

2.2.3 Model-Based methods

Model-based methods deploy a battery model with advanced algorithms to estimate the states of a battery from its measured parameters such as voltage, current, and temperature. There are two main model used for SoC estimation **Electrical Cicruit Model (ECM)** and **Electrochemical Model (EChM)**.

Electrical Cicruit Model-based methods are widely used for real-time battery monitoring, using a circuit model built linking together parametric resistances and capacitors. The idea behind is that the dynamic behaviour of the battery needs to be mathematically modelled in a way that the SoC, derived from its relation with OCV (§ 2.2), can be evaluated by just using voltage and current measured at the terminals of the cell, in order to do this the circuit parameters need to be found considering the operating condition of the cell, making it the very big challenge of this technique. A very simple ECM is shown in **Figure 2.3** that describes the ideal battery as an OCV source in series with a resistance representing the cell's internal impedance.

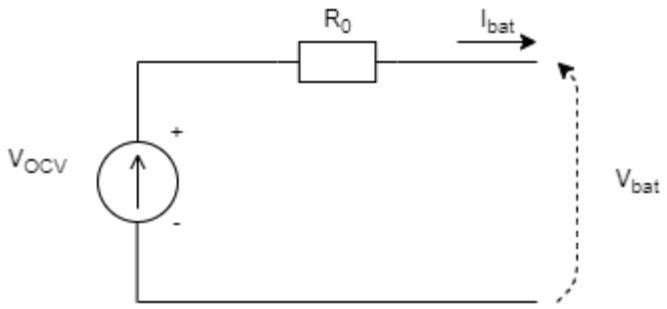


Figure 2.3: Ideal battery model

Starting from the ideal model, the main improvement is given by adding two parallel RC network that describe better the transient response of the battery voltage.

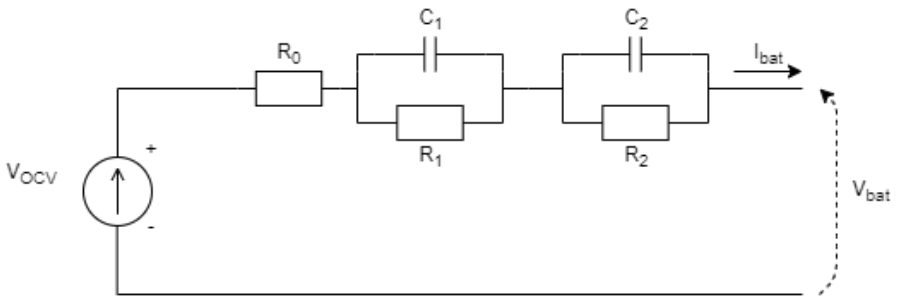


Figure 2.4: Caption

The primary RC network describes the charge transfer procedures while the secondary should be modeled to describe electrochemical polarizations. Of course the number of circuit element implies the increasing accuracy modelling together with the computational complexity. Moreover the procedure for finding the correct parameters is costly and high time consuming, not being able anyway to fully describe the electrochemical reactions occurring inside the battery and the inaccuracies related to those. Nevertheless, once the parameters are found, the equations for SoC are simple enough to be implemented by low cost micro-controllers.

Electrochemical Models use the equations that describe physicochemical phenomena like diffusion, intercalation, and electrochemical kinetics occurring in a battery. The definition of these equations involves a specialized knowledge of electrochemistry, hence their exploitation in the field of electrical and electronics engineering is not common [13]. The main limitation is the mathematical complexity to describe those models being so difficult a BMS implementation-tion, as such equations would have to be solved in real time. To apply those models for online battery monitoring is still something to fully explore, but reduced models have been developed to facilitate employment [8]. The biggest advantage of EChMs is that they inherently include the dependence of the battery behavior on SoC and temperature, while electrical models must store their parameters as look-up tables for various SoC and temperature combinations and they need to be continuously updated for accurate estimation.

2.2.4 Kalman filter-based methods

Kalman filters belong to those techniques defined as adaptive, combining direct and model-based methods, adjusting the systems to changes. SoC is affected by numerous factors and is continuously changing, also according to the user, especially in EVs. The Kalman filter is able to minimize the uncertainty by predicting the cell status and correcting that estimation [18].

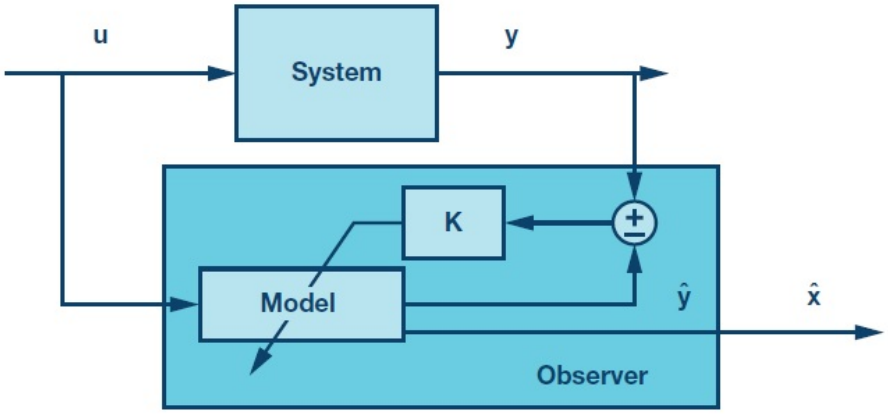


Figure 2.5: Kalman filter schematic

The key element in this method is the feedback network performing the correction. As shown in **Figure 2.5**, the output of the actual system y is compared with the prediction of the model $\hat{y}$. The result is used for correcting the model, if the error between the prediction and the actual estimation is too wide, applying the gain K to the model. After a certain transitory period the estimation converges to the state of the real system, meaning that the model is able to perform an accurate prediction of the system behaviour. Kalman filter implementation resulted as an accurate SoC and SoH estimation method for Li-ion Battery Pack, having nevertheless some drawbacks. Its complexity and computational cost are still very high, making its implementation difficult for ordinary micro-controllers. Of course this kind of solutions is also highly dependant on the battery model design, any inaccuracy in the system modelling may degrade the performance of the filter, causing slow adaptation.

2.2.5 Neural networks-based methods

A Neural Network is a computer architecture modelled upon the human brain's interconnected system of neurons which mimics its information processing, memory and learning processes. They resemble human brain for their ability to acquire knowledge through learning and for storing knowledge within neural interconnections, known as weights. A Neural Network is structured in layers as shown in **Figure 2.6**. Input layer receives the input parameters distributing toward the output layer, through the hidden layers, and each node multiplies the inputs, that come from previous layers, with the weight coefficient [2].

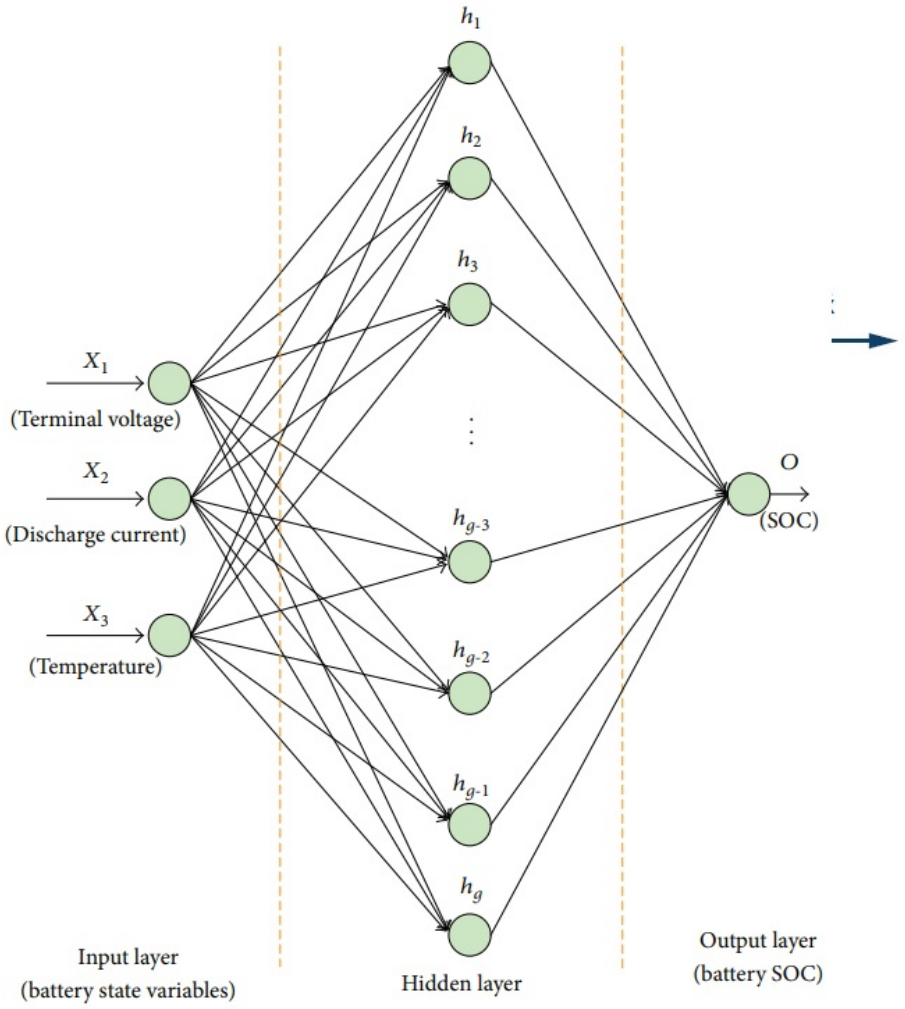


Figure 2.6: Neural network for SoC estimation

Neural Network techniques are useful in estimating battery performance which depends on quantifying the effect of numerous parameters most of which cannot be defined with mathematical precision. Also in this case the calculation complexity is very high, especially knowing that a big number of nodes are needed in order to increase accuracy. Moreover it needs some time for learning about the battery and then adapt to it considering also that in the meantime battery is aging. The high computational effort makes the method not quite fitting for online battery monitoring for automotive applications [13].

2.3 Algorithm implemented

For this thesis work the solution implemented for SoC estimation is based on look-up tables constructed from cell's measured data. To describe the cell behaviour from empirical values is a very good method since it automatically takes in account many of the factors affecting the State of Charge. These look-up tables are step-wise approximations derived from cell curves describing their performance related to discharge rate and other parameters. For this study the cells chosen are Li-ion Molicel-INR-21700-P42B [17] which characteristics are shown in **Figure 2.7**. What is represented in those figures is a general description of how the cell behaves in different conditions. The first two images above represent the discharge rate as a function of the cell voltage. The difference between these two images is that the first shows the characteristic of the cell at constant temperature, discharged at different amounts of current. The second image, on the other hand, has the same purpose, which shows the behavior of the cell which discharges at a constant amount of current, but at different temperatures. The last image, below, is the one that shows the charging characteristics of the cell, representing Capacity, Voltage and Current over time. The key aspect is that the reload time changes when run at different **C-Rates** (C). C-Rate is a way of measuring how fast a battery is charged or discharged, which means that if a battery has a capacity of 4200 mA/h, 4.2A corresponds to 1C. This kind of method let us to employ the model we are going to derive on cell belonging to the same chain of production. This means that the cell characterization from look-up tables should be carried out once with the

assumption that the accumulator is always constructed using the same kind of cell. If cell is changed during design process, the model must be redefined according to the empirical data describing that cell behaviour. The biggest drawback is that if the number of data available is not enough the model derived is not able to characterize the cell behaviour in different conditions. Of course there's still the possibility to implement advanced methods, like neural network based once, that are able to extrapolate a model even if the amount of data does not cover the whole cell performance, but the consequence is that the implementation for on-line monitoring using a micro-controller is not feasible. The best solutions is to use a simpler model assuming the availability of a bigger data set, which requires a big workload, since it needs to test the cell many times, and very time consuming to be completed, in order to cover as much as possible the whole cell performance. In any case using this method those kind of errors on SoC evaluation, coming from cell internal chemical reaction or cell external operating conditions, should be kept at minimum. For example in this case, having at disposal both charge and discharge characteristics let us minimize the error derived from battery hysteresis.

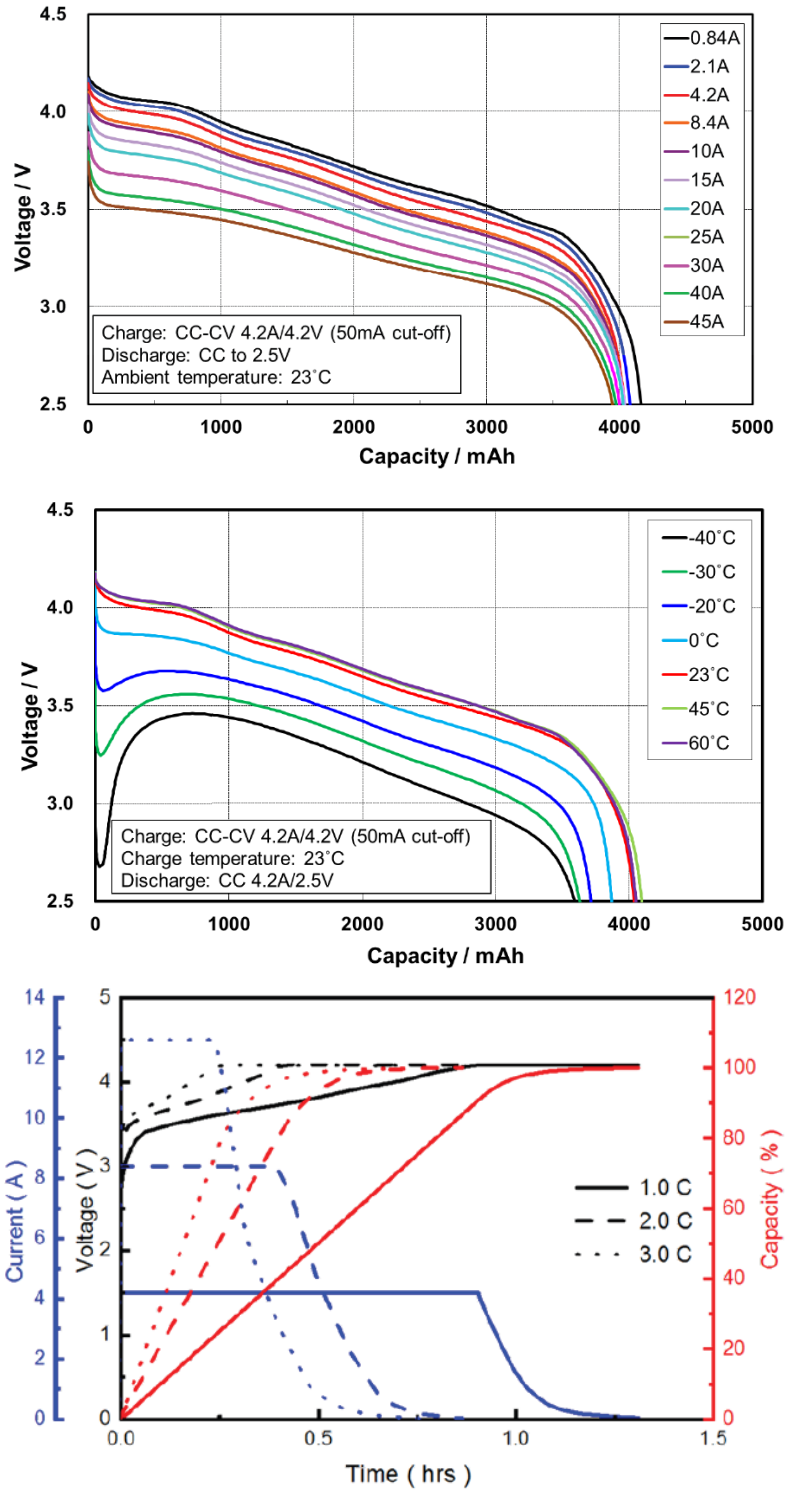


Figure 2.7: Molicel Lithium-ion Cell characteristics

The mathematical model is built using polynomial regression analysis, in which the non-linear relation between multiple input variables is modeled as an nth degree polynomial, so that the model can be used to predict the general trend of the system. In order to better explain the model, let's start from the basic concepts.
The linear regression algorithm is the simplest way to predict the value of a *dependent* variable Y, starting from an *independent* variable X. In case X is a set of variables, the model implemented is called Multiple Linear Regression.

Simple Linear Regression model equation: $Y = \theta_0 + \theta_1 X$

Multiple Linear Regression model equation: $Y = \theta_0 + \theta_1 X + ... + \theta_n X_n$

As it is possible to extrapolate from the the mathematical equations, the relationship is modeled as a straight line that is able to characterize the input dataset and predict future values assuming that the dataset remains linear. As soon as the relationship between variables is no more linear, this kind of model is not able to fit, having a high prediction error. **Figure 2.8** shows a non linear dataset and its Linear Regression model which is not able to estimate the best fit line.

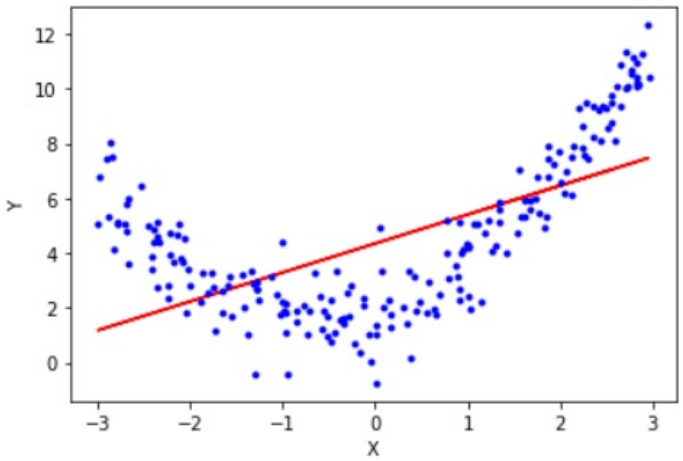


Figure 2.8: Example of non linear dataset

Polynomial regression is similar to multiple linear regression but the relation-

ship between the dependent and the independent variables is not linear. This type of regression is used when data points are present in a non-linear fashion. The polynomial regression algorithm transforms the data points into polynomial features of specified degrees and models them using the linear technique [24].

Polynomial Regression model equation: $Y = \theta_0 + \theta_1 X + ... + \theta_n X^n$

As shown in **Figure 2.9** the model is able to better follow the data-set trend.

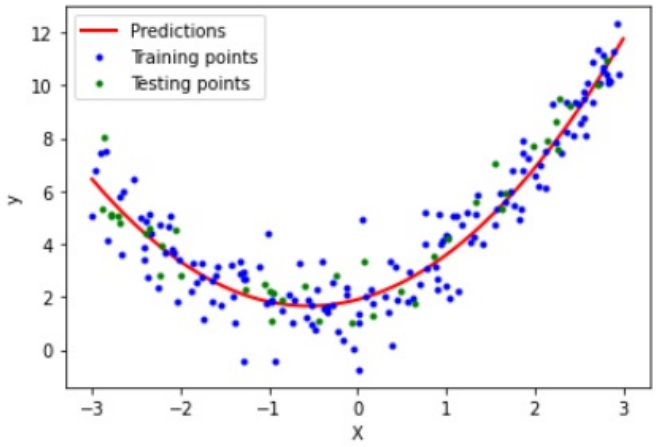


Figure 2.9: Polynomial Regression model

With this kind of regression there is the possibility to better follow the data trend, but risking to $overfit$. This happens when the dataset has missing or imbalanced data or contains noisy data, becoming more evident with higher degree polynomials. The effect is that the model adapts very well to training data points, like the blue line in **Figure 2.10**, and not so good to other points, reducing the model accuracy. A linear regression in this example gives better results.

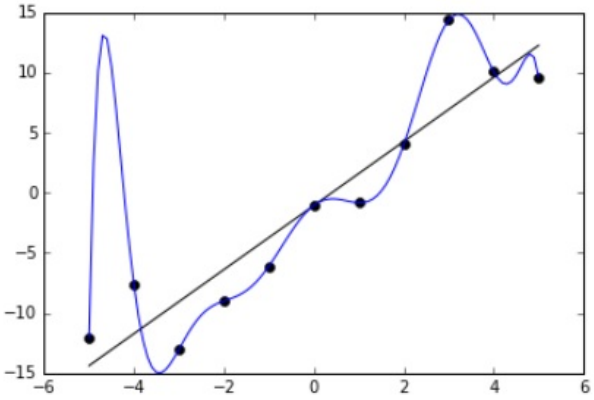


Figure 2.10: Overfitting

There are still ways to minimize the error derived from overfitted models, by eliminating those weights(θ) with higher value, since they may destabilise the model trend, but still keeping the polynomial degree [6]. What has been shown so far was the situation in which it is necessary to predict the performance of a system starting from a single input. In real situations the models are based on look-up tables that contain multiple columns. The easiest situation to present is that the dependent variable Z, has two types of inputs, X and Y, making it possible to have a data-set that can be represented in three dimensions.

Polynomial Regression model equation with 2 inputs:

$$Z = \theta_0 + \theta_1 X + \theta_2 Y + \theta_3 XY + ... + \theta_n X^n + \theta_n Y^n$$

In order to make sure that the model is able to predict the cell trend, the first important step is to split the dataset in two or three subsets:

- **Train set**: it is the actual set of data used by our model to learn about the relation between Y and X. From here the model is training to develop and predict the output trend trying to minimize the difference between prediction and real data. This is done by updating the weights θ.
- **Validation set**: it is the set of data able to validate the predictive capabilities of the model minimizing overfitting. After that the model

evaluated the weight, so after it made a prediction, it is given a portion of X that till now the model did not know about and it makes a prediction. Now there's a Y predicted by the model and a real Y belonging to the input dataset, that are compared to see how much distance there is between those two variables. If the prediction error is too high the model parameters must be changed, by increasing the grade of the equation and restart from training.

- **test set**: this set is omitted most of the time. This is used to perform further observation on the model prediction, after that its performed was considered suitable with train and validation.

Due to its great importance, the train set is usually the one containing the majority of the variables belonging to our input look-up table. The key is to define the percentage of variables to be given at the validation set. A too large validation set would mean stealing learning data, while a set too small is not able to fully validate the model trend [19]. For this work the test set has been omitted and the dataset division has been randomly performed between train set and validation set as shown in **Figure 2.11**.

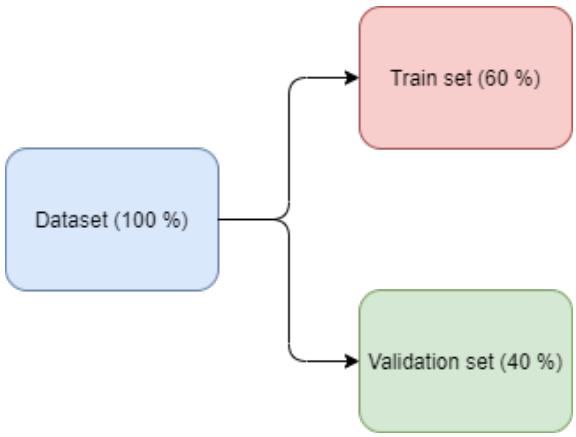


Figure 2.11: Dataset division

```
% Dataset randomly divided: Train 60%, Validation 40%
[index_training,index_validation] = dividerand(length(X),0.6,0.4,0);

X_training = X(index_training);
Y_training = Y(index_training);
Z_training = Z(index_training);

X_validation = X(index_validation);
Y_validation = Y(index_validation);
Z_validation = Z(index_validation);
```

What is shown above is a portion of the code written on $MATLAB$, regarding the dataset variables initial manipulation. The model that is going to be presented, is the one predicting the State of Charge of a cell, starting from a look-up table containing the relation between SoC, voltage and current. This dataset has been completed from cell characteristics shown in **Figure 2.7** and then reported in **Figure 2.12**. The output independent variable to be predicted, the SoC, is labeled as Z, while voltage and current are the dependent variables labeled as X and Y respectively. The validation is evaluated calculating the Root Mean Square Error (RMSE), when it's too high the training is repeated with a new set of parameters, increasing grade model equation.

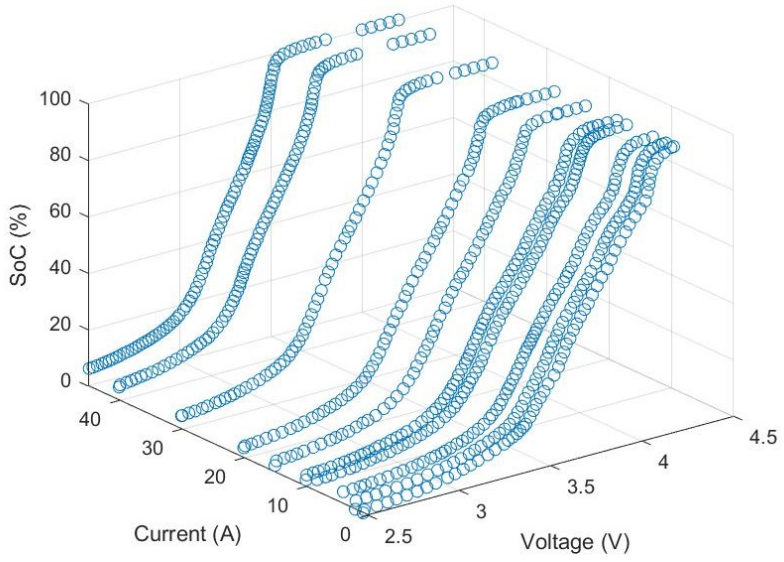


Figure 2.12: Input dataset

2.3.1 1^{st} grade model equation

This equation represents also the multi-linear regression model, since it describes a linear relationship between dependent variable and a set of independent variables. As we already know this model is not fitted due to the non linearity of the dataset. This can be noticed looking at **Figure 2.13**.

$$Z = \theta_0 + \theta_1 X + \theta_2 Y$$

$$\theta_0 = -214.95,\ \theta_1 = 73.586,\ \theta_2 = 0.6411$$

$$RMSE = 11.6085$$

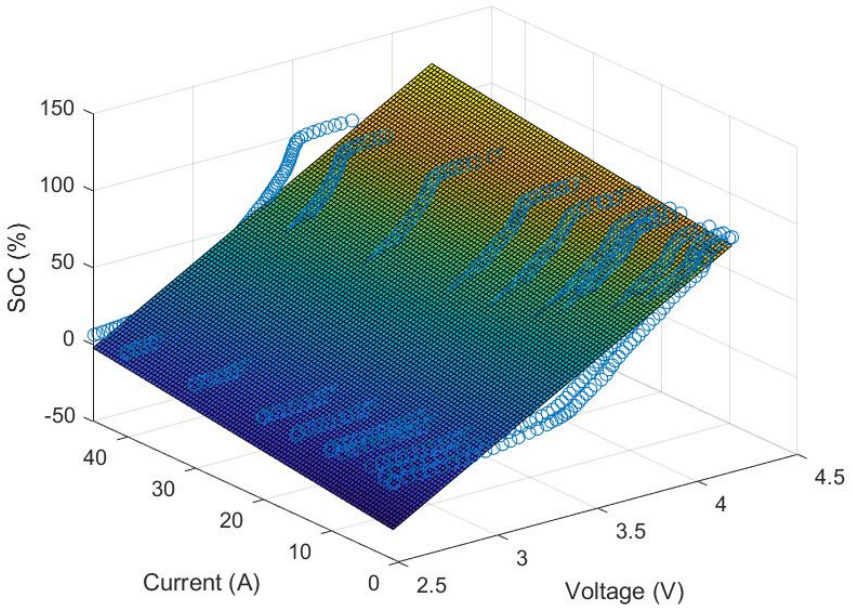


Figure 2.13: Grade 1 model equation

2.3.2 2^{nd} grade model equation

Form now on the model starts to have non linear behaviour, as shown in **Figure 2.14**. Nevertheless this equation is still not suitable with respect to the input dataset. Moreover the RMSE value is still too high to consider the model as good.

$$Z = \theta_0 + \theta_1 X + \theta_2 Y + \theta_3 XY + \theta_4 X^2 + \theta_5 Y^2$$

$$\theta_0 = 165.8021,\ \theta_1 = -150.8119,\ \theta_2 = -0.7859$$

$$\theta_3 = 0.5280,\ \theta_4 = 32.2993,\ \theta_5 = -0.0063$$

$$RMSE = 9.0881$$

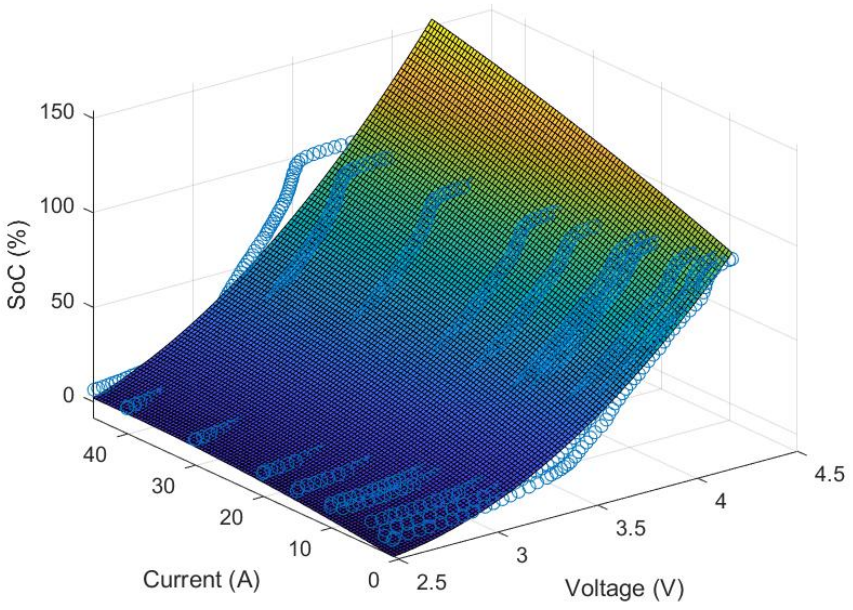


Figure 2.14: Grade 2 model equation

2.3.3 3^{rd} grade model equation

At this point equation complexity is starting to grow, together with accuracy, having higher grade with 10 parameters and the RMSE value is starting to decrease. It is still not enough, not fitting correctly at both ends of the dataset, **Figure 2.15**.

$$Z = \theta_0 + \theta_1 X + \theta_2 Y + \theta_3 XY + \theta_4 X^2 + \theta_5 Y^2 + \theta_6 X^2 Y + \theta_7 XY^2 + \theta_8 X^3 + \theta_9 Y^3$$

$$\theta_0 = 2660.1,\ \theta_1 = -2351.7,\ \theta_2 = -20.265$$

$$\theta_3 = 12.223,\ \theta_4 = 670.07,\ \theta_5 = 0.0060$$

$$\theta_6 = -1.6780,\ \theta_7 = -0.0129,\ \theta_8 = -60.835$$

$$\theta_9 = 0.0004$$

$$RMSE = 5.4602$$

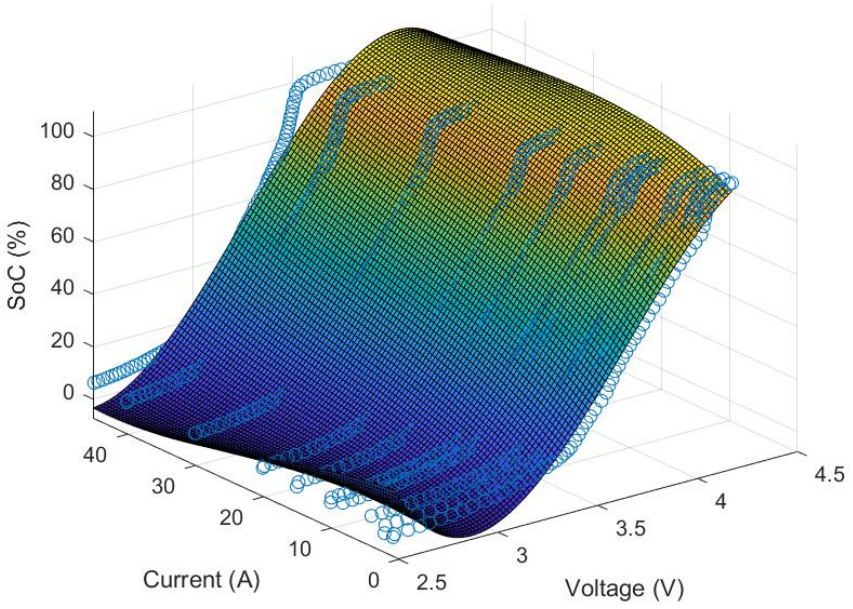


Figure 2.15: Grade 3 model equation

2.3.4 4^{th} grade model equation

Having increased the degree, the number of parameters that make up the equation has increased accordingly. RMSE value still not valid and visually the model in not enough accurate.

$$Z = \theta_0 + \theta_1 X + \theta_2 Y + \theta_3 XY + \theta_4 X^2 + \theta_5 Y^2 + \theta_6 X^2 Y + \theta_7 XY^2 + \theta_8 X^3 + \theta_9 Y^3 + \theta_{10} X^3 Y + \theta_{11} X^2 Y^2 + \theta_{12} XY^3 + \theta_{13} X^4 + \theta_{14} Y^4$$

$$\theta_0 = -7039.1,\ \theta_1 = 8945.4,\ \theta_2 = 107.31$$

$$\theta_3 = -103.05, \theta_4 = -4202.1,\ \theta_5 = -0.1854$$

$$\theta_6 = 32.54,\ \theta_7 = 0.089,\ \theta_8 = 862.22$$

$$\theta_9 = 0.0019,\ \theta_{10} = 4.5 \cdot 10^{-5},\ \theta_{11} = -0.017$$

$$\theta_{12} = -3.335,\ \theta_{13} = -64.89,\ \theta_{14} = -1.8 \cdot 10^{-5}$$

$$RMSE = 3.5718$$

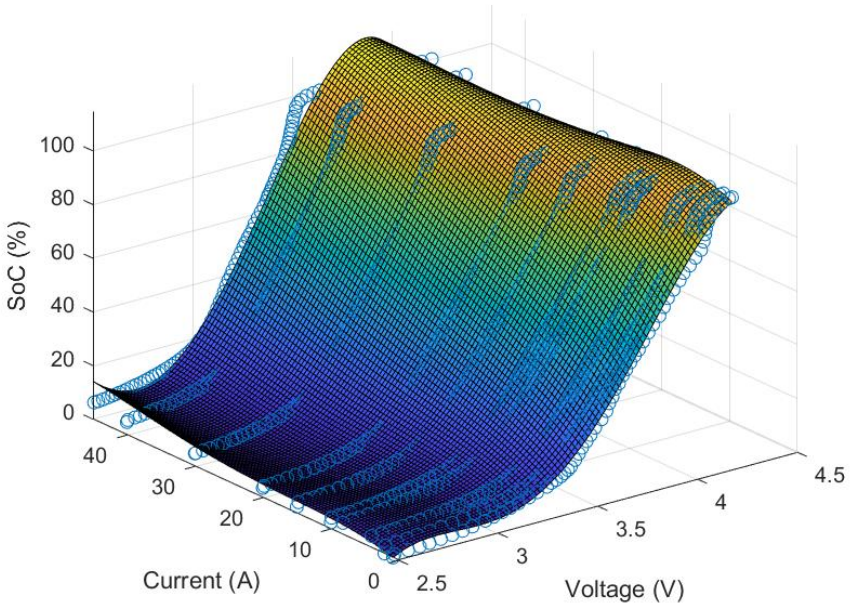


Figure 2.16: Grade 4 model equation

2.3.5 5^{th} grade model equation

$$Z = \theta_0 + \theta_1 X + \theta_2 Y + \theta_3 XY + \theta_4 X^2 + \theta_5 Y^2 + \theta_6 X^2 Y + \theta_7 XY^2 + \theta_8 X^3 + \theta_9 Y^3 + \theta_{10} X^3 Y + \theta_{11} X^2 Y^2 + \theta_{12} XY^3 + \theta_{13} X^4 + \theta_{14} Y^4 + \theta_{15} X^4 Y + \theta_{16} X^3 Y^2 + \theta_{17} X^2 Y^3 + \theta_{18} XY^4 + \theta_{19} X^5 + \theta_{20} Y^5$$

$$\theta_0 = -29852,\ \theta_1 = 43442,\ \theta_2 = 241.26$$

$$\theta_3 = -266.14,\ \theta_4 = -24896,\ \theta_5 = -0.4283$$

$$\theta_6 = 105.46,\ \theta_7 = 0.4451,\ \theta_8 = 7020.7$$

$$\theta_9 = -0.0037,\ \theta_{10} = 0.0035,\ \theta_{11} = -0.1617$$

$$\theta_{12} = -17.541,\ \theta_{13} = -974.48,\ \theta_{14} = -2.9 \cdot 10^{-5}$$

$$\theta_{15} = 1.0196,\ \theta_{16} = 0.0172,\ \theta_{17} = -0.0004$$

$$\theta_{18} = -1.1 \cdot 10^{-5},\ \theta_{19} = 53.363,\ \theta_{20} = 4.2 \cdot 10^{-7}$$

$$RMSE = 2.9890$$

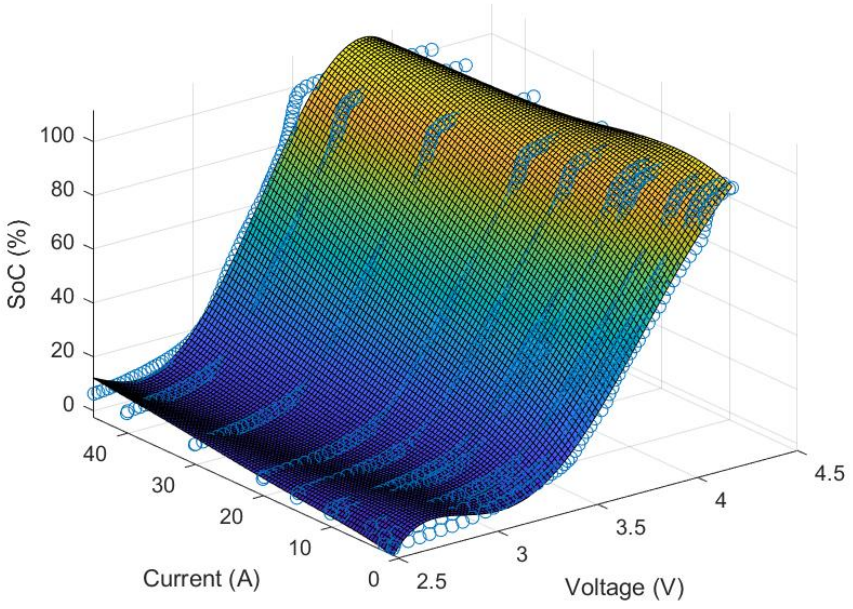


Figure 2.17: Grade 5 model equation

2.3.6 6^{th} grade model equation

Most accurate model obtained having $RMSE < 2$. Equation complexity is very high.

$$Z = \theta_0 + \theta_1 X + \theta_2 Y + \theta_3 XY + \theta_4 X^2 + \theta_5 Y^2 + \theta_6 X^2 Y + \theta_7 XY^2 + \theta_8 X^3 + \theta_9 Y^3$$
$$+ \theta_{10} X^3 Y + \theta_{11} X^2 Y^2 + \theta_{12} XY^3 + \theta_{13} X^4 + \theta_{14} Y^4 + \theta_{15} X^4 Y + \theta_{16} X^3 Y^2 +$$
$$\theta_{17} X^2 Y^3 + \theta_{18} XY^4 + \theta_{19} X^5 + \theta_{20} Y^5 + \theta_{21} X^5 Y + \theta_{22} X^4 Y^2 + \theta_{23} X^3 Y^3 +$$
$$\theta_{24} X^2 Y^4 + \theta_{25} XY^5 + \theta_{26} X^6 + \theta_{27} Y^6$$

$$\theta_0 = -469.54,\ \theta_1 = 0,\ \theta_2 = -2394.8,\ \theta_3 = 3652$$

$$\theta_4 = 246.79,\ \theta_5 = 11.596,\ \theta_6 = -2198,\ \theta_7 = -14.991$$

$$\theta_8 = 0,\ \theta_9 = 0.0093,\ \theta_{10} = 0.0051,\ \theta_{11} = 7.0575$$

$$\theta_{12} = 652.35,\ \theta_{13} = -74.284,\ \theta_{14} = -0.0005,\ \theta_{15} = -95.414$$

$$\theta_{16} = -1.4556,\ \theta_{17} = -0.0023,\ \theta_{18} = 5.7125 \cdot 10^{-5},\ \theta_{19} = 24.504$$

$$\theta_{20} = 7.0458 \cdot 10^{-6},\ \theta_{21} = 5.5003,\ \theta_{22} = 0.1107,\ \theta_{23} = 0.00035$$

$$\theta_{24} = -1.087 \cdot 10^{-5},\ \theta_{25} = 7.816 \cdot 10^{-8},\ \theta_{26} = -2.3131,\ \theta_{27} = -4.7735 \cdot 10^{-8}$$

RMSE = 1.9796

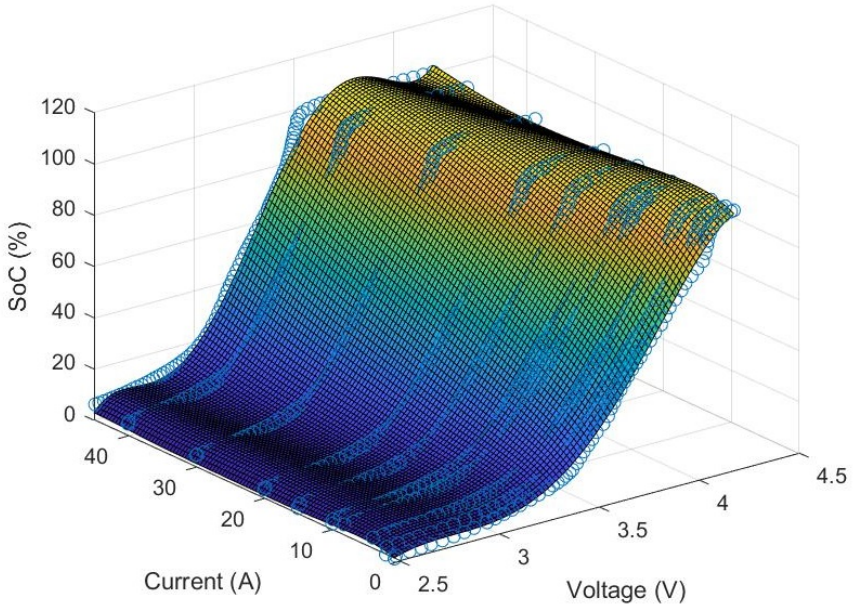


Figure 2.18: Grade 6 model equation

2.3.7 Conclusions

The last model presented is the one with the lowest value of RMSE, so the most accurate. The overfitting is minimal in regions where train data is not present meaning that is not required any additional work and the equation can be still employed, despite its complexity, for a real time online calculation by the Battery Management System Master ECU. The main problem is that the model presented is the one trying to predict the battery trend according to Voltage and Current with constant temperature at 23 °C. This is not enough since temperature is constantly changing around a Battery Pack, making the SoC to change accordingly. For this purpose, there was the attempt to develop a model using as independent variables voltage and temperature derived from the cell characteristic from **Figure 2.7**. The problem was that the little amount of data and the difficult behaviour of the voltage at low temperature, make the model too imprecise with many regions where overfitting occurred, especially where the train data is less dense. **Figure 2.19** shows the model obtained using polynomial regression with 6^{th} grade equation. This one was the most precise obtained, but the RMSE value is still too high.

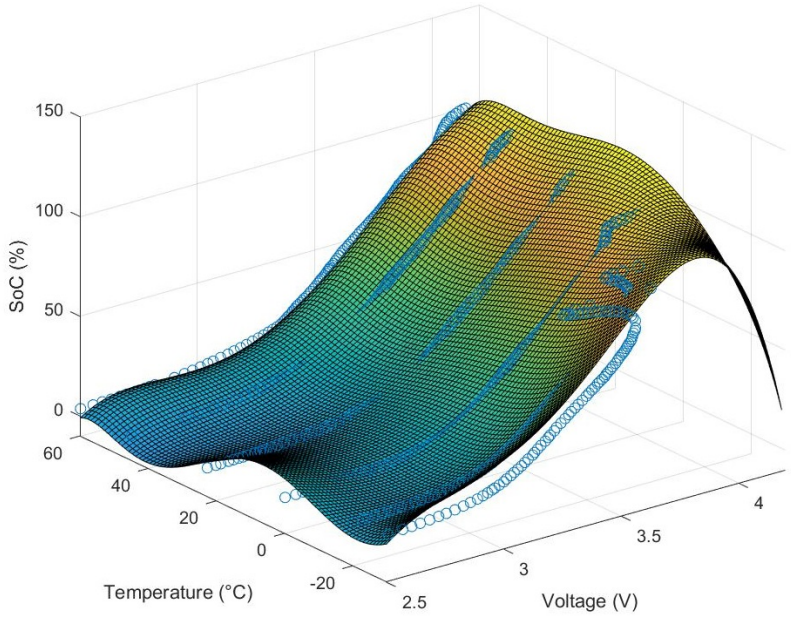


Figure 2.19

Between these two models presented, the V-I model is the one easier to implement. There exist in fact solutions to let the temperature around the Battery Pack to be constant, letting cells to operate at optimal conditions.

CHAPTER 3

Firmware

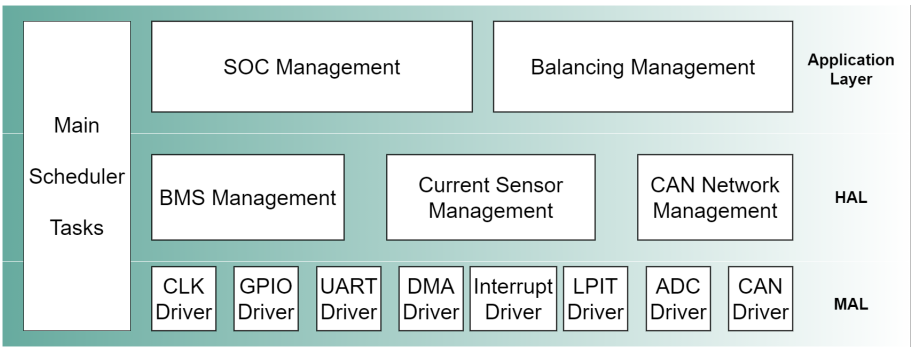


Figure 3.1: Firmware structure

The firmware has been structured to have three main layers. Further visual details are shown in the **Figure 3.1**

- **Microcontroller Abstraction Layer**: modules, or drivers, belonging to this layer have direct access to all the MCU peripheral modules and external devices, which are mapped to memory. In this way upper layers are able to work independently from the micro.
- **Hardware Abstraction Layer**: this layer allows the application to interact with the hardware device at a general or abstract level rather than at a detailed.

Functions belonging to this module are programmed independently to the physical interface of the device or the actual Hardware features.
At this stage is where data are managed and sent to upper layer.

- **Application Layer**: this is the upper level of the structure, where all interactions with an external user take place. This is the upper level of the structure, where all the interactions with an external user take place, in this case the battery pack itself. Here the data coming from the lower levels are acquired and consequently the management of the battery system takes place according to the processing of this data.

The execution of the functions belonging to the three layers is carried out by the **scheduler** inside the main. The idea is to entrust the execution of certain portions of code to three different **tasks** that work at three different frequencies. In order to assure that each code portion is executed at the same frequencies, the constraints are:

- The frequency at which each task is executed is such that the period of execution of each task must be an integer multiple of the fastest task

$$\begin{cases} t_1 = a \cdot t_3 \\ t_2 = b \cdot t_3 \end{cases} \quad \text{with } a, b \in \mathbb{N},\ 1 < a < b.$$

- The execution period of each piece of code belonging to a task must be lower then the time lapse between calls of the fastest task in order to avoid overlapping, which means $T_{ex} < min\{t_1, t_2, t_3\}$

The three tasks are executed with frequency of 1 ms, 2 ms, 5 ms each. Before the execution of the main, during an initialization phase, a timer is set that counts the number of clock cycles that correspond to the execution frequency of the tasks. **Figure 3.2** shows the flowchart of the pieces of codes belonging to the three tasks.

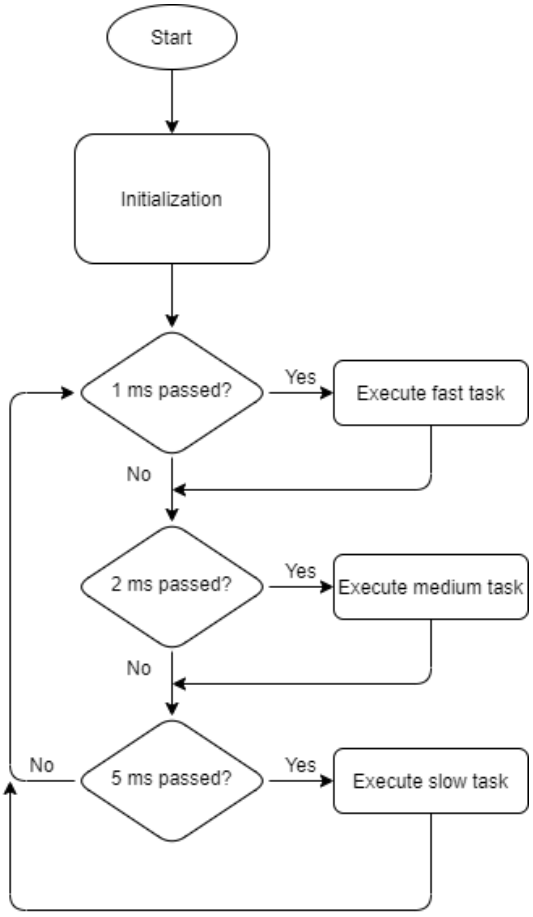


Figure 3.2: Firmware flow chart

3.1 Master-Slave data exchange

The acquisition of voltage and temperature values to monitor the status of a battery is managed directly by the slave and then sends everything in a message via UART. The exchange of information through the UART protocol is instead managed by the micro through the specific driver that works together with the DMA in such a way as to speed up the data processing procedure.

3.1.1 UART message structure

The Universal Asynchronous Receiver Transmitter (UART) protocol is among the most used in the embedded world, it is a type of P2P communication in which the two communicating nodes have two different clock sources, so it is defined asynchronous. Nevertheless, the data transmission requires periodic synchronization between the two nodes so that the message is transmitted and received correctly. The transmission of a message requires a subdivision into frames. What is shown in **Figure 3.3** is the general structure of a frame transmitted by the master and by the slave for this thesis work.

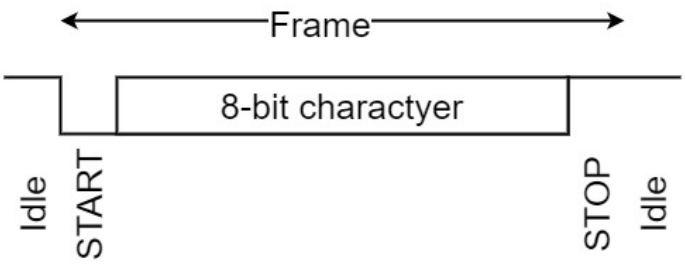


Figure 3.3: UART frame structure

- **START bit**: it is the bit used for node synchronization and to indicate the start of the frame. To favor a correct coding of the protocol, the UART line is kept at a high logic level when it is not used, therefore in idle. The START bit is the first bit to force a transition from high to low level, for this reason it is used for synchronization.
- **STOP bit**: the stop bit indicates the end of the frame. In some cases there may also be two STOP bits and is always kept at a logical high level, before going into idle state. The two states are differentiated by the fact that the position, and number of bits that make up the stop of the frame is known, once the stop bit is identified.
- **8-bit data**: this portion of the frame is nothing more than the actual information to be sent. In this case its length is 8 bits, but it can vary. The most common cases concern 6 to 9 bit data transmission

The UART protocol also provides for the use of an optional bit for parity checking.

All these are parameters set in the UART driver that manages its interface. The transmission rate of the message is also set at this level. This parameter is defined as Baud Rate, which would be the amount of symbols transmitted in one second, in this case the transmission speed is set to 1 MBaud, meaning that the transmission of one bit requires 100ns. If the master transmits the message, this means that the transmission of a bit requires to set a timer that counts 8 clock strokes, the latter having a frequency of 80MHz. The structure of the UART messages that the master must send to the slave ECU is predefined by the microcontroller manufacturer, as specified in the datasheet [23]. The communication via UART by the master, at the beginning consists in having to program and set the slave microcontroller, following the procedures already defined by the manufacturer, in order to carry out the actual acquisition of the battery monitoring parameters. This occurs by interacting directly with the microcontroller registers, setting how to manage certain peripherals. Some examples are the setting of the voltage reading thresholds, the sampling rate and the BAUD rate. For this reason the procedure for communicating with the slave involves the transmission of five different types of frames:

- **Frame initialization**: This is the first frame to be sent in a message and is used to define the type of message, whether it is a response or a command, to whom it is intended, since the slave system is made up of several ECUs, which register to set, based on functionality, and the amount of data to be sent. More details are shown in **Figure 3.5**.

	7	6	5	4	3	2	1	0
Command Frame Init	FRM_TYPE = 1	REQ_TYPE			ADDR_SIZE	DATA_SIZE		
Response Frame Init	FRM_TYPE = 0	RESP_BYTES-1						

Figure 3.4: Frame initialization byte

- **Device address**: It identifies the device or group of devices targeted by the command. If the message is defined by the previous frame as a Broadcast message, this frame can also be omitted.

- **Register address**: This frame defines the address of the register to be overwritten or the one from which we want to read data.
- **Data**: The number of data Bytes is defined in the first frame. In case the message is a command this frame contains the amount of data to write in the destination register. Otherwise it contains the information contained in the target register.0
- **CRC**: This portion is 2 bytes long, contained in two frames, useful for checking the correctness of the transmitted message

	VALUE (BINARY)	DESCRIPTION
FRM_TYPE	0	Response Frame
	1	Command Frame
REQ_TYPE	000	Single Device Write with Response
	001	Single Device Write without Response
	010	Group Write with Response
	011	Group Write without Response
	100	Reserved
	101	Reserved
	110	Broadcast Write with Response
	111	Broadcast Write without Response
ADDR_SIZE(1)	0	8-bit Register Address
	1	16-bit Register Address
DATA_SIZE(2)	000	0 bytes
	001	1 byte
	010	2 bytes
	011	3 bytes
	100	4 bytes
	101	5 bytes
	110	6 bytes
	111	8 bytes
RESP_BYTES-1		Number of data bytes contained in response frame minus 1

Figure 3.5: Frame initialization byte fields table

An example of command frame is the one used for setting the Baud rate:

- **0xF2**: Setting a Broadcast command without response. Data quantity 2 Bytes. No device address frame is required.
- **0x10**: Communication configuration register address.
- **0x30E0**: Data to be written inside the register. In this case in order to set the baud rate, bit 13 and 12 needed to be written as 1.
- **0x26F5**: 16-bit CRC

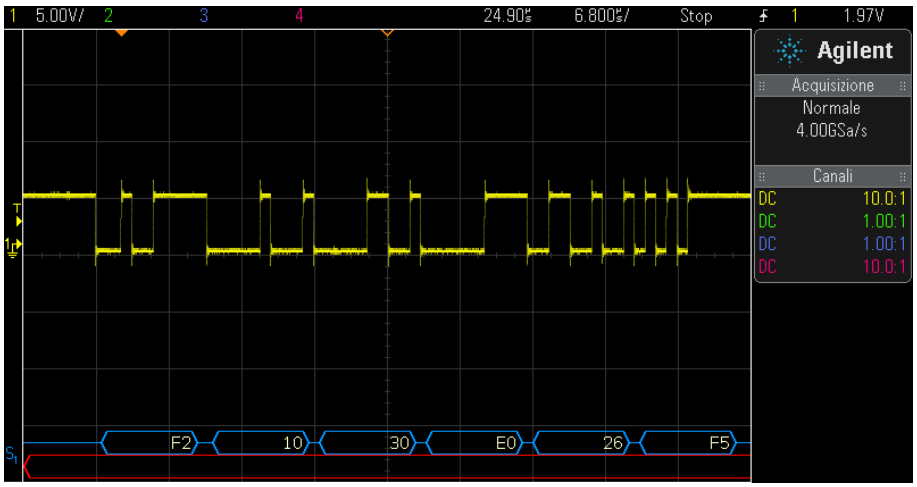


Figure 3.6: UART command message acquisition

As already said previously, the management of the UART device is entrusted to its driver which sets the dedicated registers to have a transmission of messages at 1 MBaud rate and 10-bit frames. The UART driver works concurrently with DMA for both incoming and outgoing messages. The Direct Memory Access (DMA) is the mechanism that allows other subsystems, such as peripherals, to directly access the internal memory to exchange data, read and write, avoiding overloading the CPU with a series of interrupts that could slow it down.

When a message is to be sent, the data is stored in a buffer in Direct Memory Access (DMA), allowing the UART driver to read the data from the memory, construct the frames and send the message. The reverse process occurs in reception. The UART driver saves the data in Direct Memory Access (DMA), ready to be read by the functions dedicated to data processing at the HAL level. This whole procedure is synchronized by an interrupt system that signals the drivers when to act. The main reason why this solution has been implemented is because, once the DMA and UART drives have been set, the message transmission and reception procedure takes place automatically, including saving the data received in the memory in a static variable, allowing the portions of code belonging to the HAL to continue processing data in parallel. Once the transmission or reception is finished, an interrupt is sent in order to avoid that two transmission operations can overlap or that a received data is overwritten before being read and, furthermore, in this way the exchange of information takes place at such a high speed.

```
void DMA1_IRQHandler(void)
{
        DMA -> CINT = 1; /*Clear channel 1 interrupt request*/
        /*UART Tx management*/
        BMSTxDMACallback();
}
```

```
void DMA0_IRQHandler(void)
{
        /*Interrupt request for channel 0 - UART RX*/
        DMA -> CINT = 0; /*Clear channel 0 interrupt request*/
        /*Disable Timer*/
        LPIT0Ch1Stop();
        /*Callback function for managing received values in UART*/
        BMSRxDMACallback();
}
```

The two functions shown above handle the interrupt request when the transmission and reception procedures are finished. The first calls a function that simply allows the system to be able to transmit again. The second, on the other hand, starts the reading of a new data received and its subsequent processing.

3.1.2 BMS data management function

Before reaching the phase in which the voltage and temperature values are cyclically acquired from the slave system, the Master ECU must first perform a series of steps, concerning the transmission of messages via UART, predefined according to the datasheet, in order to correctly set the functionality of the slave system. There are two first stages, the first for initialization and a second one for setting up the analog front-end. In both these states, the messages are only transmitted by the master, with the possibility, however, of requesting to reply to some messages to check the correctness of the previous actions. To speed up the procedure, already knowing the structure of the messages, they have been built in advance and saved in a matrix, in the order in which they must be sent. So once the system is in one of the two states, it constructs the message and saves it in DMA ready to be transmitted.

```
/*{datasize, respsize, cmdtype, devaddr, regdest, TxData}*/
static uint08t *MsgConfigInit[] = {
/*Set Communication Rate(1MBaud) and no comm fault*/
(uint08t[]){2u, 0u, BroadcastNoResp, 0u, COMCONFIGreg, 0x30u, 0xE0u},
/*Enable Auto-Addressing Mode*/
(uint08t[]){1u, 0u, BroadcastNoResp, 0u, DEVCONFIGreg, 0x10u},
/*Start Auto-Addressing*/
(uint08t[]){1u, 0u, BroadcastNoResp, 0u, DEVCTRLreg, 0x08u},
/*Set Address for each slave*/
(uint08t[]){1u, 0u, BroadcastNoResp, 0u, DEVADDRreg, 0x00u},
(uint08t[]){1u, 0u, BroadcastNoResp, 0u, DEVADDRreg, 0x01u},
/*check that the addresses are correct*/
(uint08t[]){1u, 1u, SingleResp, 0u, DEVADDRreg, 0x00u},
/*disable COMM_LOW high-side*/
(uint08t[]){2u, 0u, SingleNoResp, 0u, COMCONFIGreg, 0x30u, 0x80u},
/*Clear All faults*/
(uint08t[]){2u, 0u, SingleNoResp, 0u, FAULTSUMMARYreg, 0xFFu, 0xC0u}
};
```

```
/*{datasize, respsize, cmdtype, devaddr, regdest, TxData*/
static uint08t *MsgConfigAFE[] = {
/*Set sample delay 0*/
(uint08t[]){1u, 0u, SingleNoResp, 0u, SAMPLEDELAYreg, 0x00u},
/*Clear all faults*/
(uint08t[]){1u, 0u, SingleNoResp, 0u, STATUSreg, 0x38u},
/*Set sample period for cells to 60us*/
(uint08t[]){1u, 0u, SingleNoResp, 0u, CELLSPERreg, 0xBCu},
/*Set sample period for temp to 60us*/
(uint08t[]){4u, 0u, SingleNoResp, 0u, AUXreg, 0xBBu, 0xBBu, 0xBBu, 0xBBu},
/*Set no oversampling period*/
(uint08t[]){1u, 0u, SingleNoResp, 0u, OVERSAMPLEreg, 0x00u},
/*Set cells used by the device*/
(uint08t[]){1u, 0u, SingleNoResp, 0u, NCHANreg, 0x00u},
/*Set cells/aux included in the measurement*/
(uint08t[]){4u, 0u, SingleNoResp, 0u, CHANreg, 0x00u, 0x00u, 0x00u, 0x00u}
};
```

The portion of code shown above is the matrix containing the structure of the messages to be sent to the slave system for the initialization phase. Unlike the next phase, that is the setting of the acquisition parameters, this requires to be performed only once, while the second must be performed for each ECU belonging to the slave chain. Once the slave system has been correctly set up, the data acquisition phase begins by decoding the UART messages received from the slave chain. The structure of the received message is quite complex. In fact it is a concatenation of messages deriving from the single ECU, starting from the one with the highest address.

The response structure of a single device in the slave chain consists of:

- **Header byte**: this value represents the number of bytes given in the response, subtracted by one, considering that for each reading the value is saved in 2 bytes. This means that if an ECU has read 6 voltage values, the response contains 12 data bytes representing those valued and the value of the header byte would be 11.
- **Data bytes**: as already specified, each voltage and temperature acquisition channel saves the read values in 16-bit registers. These are then sent in a specific order, always starting from the channels with the highest cardinality, sending first the voltage values and then the temperature.

The complexity of the message therefore requires a phase of elaboration. To speed up this, once the decoding has been carried out, the single values are saved in static variables and while the data is being processed a new request is made. In the end, the voltage values of the cell and temperature groups are saved in data structure containing information to be sent to the application level. Those shown below are in fact structures containing information on the generic state of the system and on the conditions of the cells, in terms of voltage and temperature, the latter useful for calculating the State of Charge.

```
typedef struct {
        uint16t TotalPackV;
        e_BMSStatus_t status;
}s_SysInfo_t;

s_SysInfo_t BMSSysInfo;
```

```
typedef struct {
        sint16t High;
        sint16t Low;
        sint16t Avg;
        uint08t HighId;
        uint08t LowId;
}s_cell_t;

s_cell_t BMS_TCELL;
s_cell_t BMS_VCELL;
```

3.2 SoC value calculation and validation

Once all the parameters have been acquired and processed, the voltage, current and temperature values of the accumulator are given as input to the model for the calculation of the SoC, which we talked about in the previous chapters. The model was validated in a low voltage prototype electrical system developed by the team "Squadra Corse" at Politecnico di Torino. Each year the team develops a fully electric racing car to compete in the student formula in which the best engineering project is awarded. The vehicle's electrical system is powered by two accumulators, one at high voltage, for supplying the engines, and one at low voltage, to power the set of ECUs to monitor using multiple sensors and to manage some vehicle functions. **Figure 3.7** shows the whole electrical system.

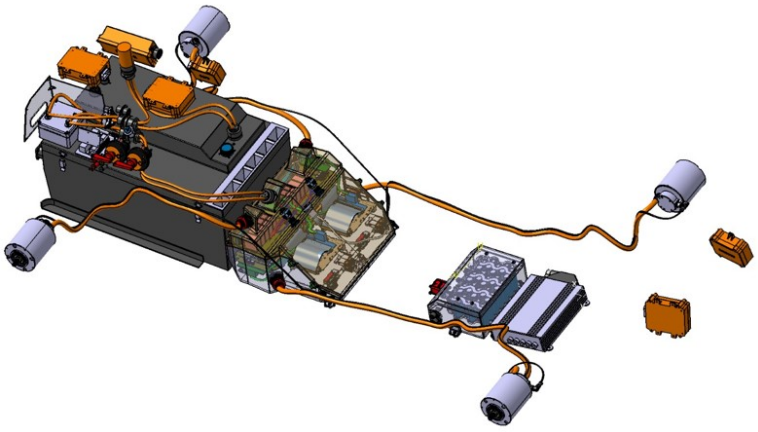


Figure 3.7: Prototype Electrical system

As for the validation of the SoC model, the parameters were acquired by writing a log file while charging the battery pack. A device for CAN Analyzing has been used, connected to CAN Network, reading the target values and constructing the log file. The acquired parameters are the same ones that are supplied in input to the model and sent to the rest of the vehicle via CAN. In the case of this thesis work, the temperature value is not considered because the model to be validated is based on the calculation of the State of Charge having voltage and current as independent parameters. The log file containing voltage, current

and SoC was then reworked on MATLAB in order to graphically represent these data.**Figure 3.8** represents the graphs obtained while **Figure 3.9** shows the path for processing the values up to the final graph.

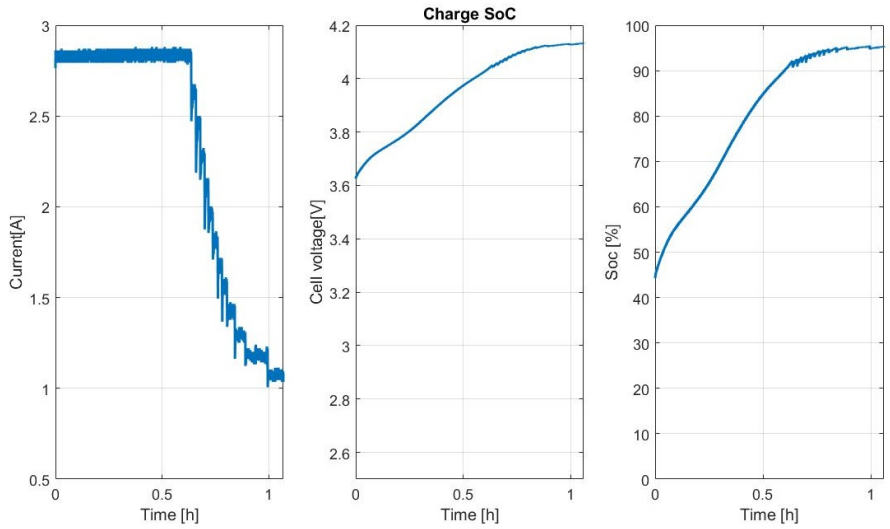


Figure 3.8: Battery charging curves

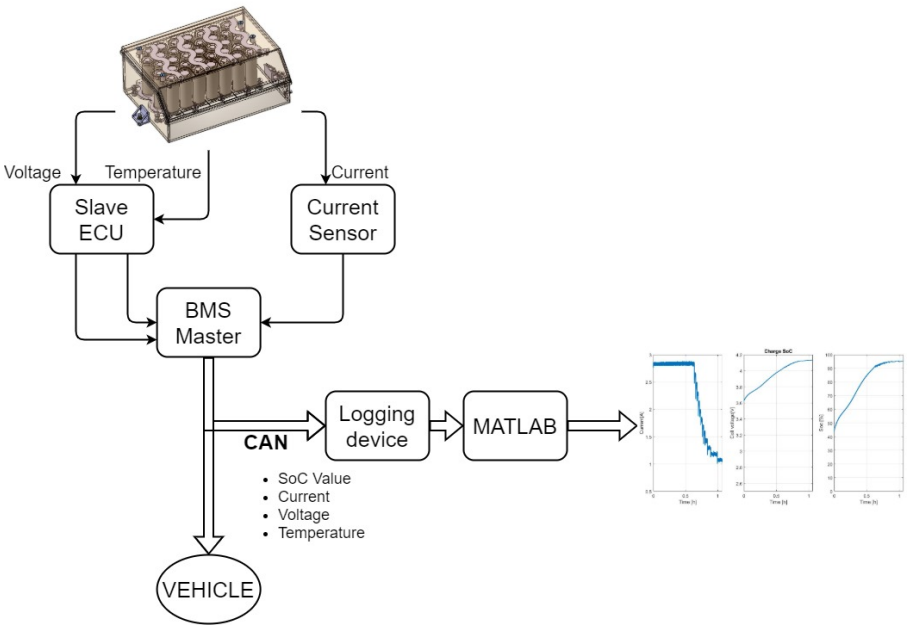


Figure 3.9: Data flowchart

CHAPTER 4

Conclusions

This thesis work stems from the idea of developing a management system for high voltage accumulators in vehicles, hybrid or fully electric, in such a way as to optimize the use of the cells, prolonging their life times and reducing the risks deriving from a bad handling, especially if the cells are made up of lithium batteries. One of the key parameters is what defines the state of charge of a cell. The main objective is to design and implement a mathematical model that can calculate the SoC parameter starting from physical characteristics that describe the battery, such as voltage, current and temperature. This model must be implementable with a microcontroller for an evaluation of the SoC parameter in real time, without blocking the execution of other functions that the MCU manages. The model was built starting from experimental data provided by the manufacturer of the lithium cells, with which it was possible to develop a function that describes the performance of the battery according to these tables and predict its behavior in different situations.The final result is a fairly precise model, able to adapt to situations other than those used for the realization of the input data set. The main limitation of the model is that the accuracy reached with the model is valid when the temperature is around 23 ° C. As noted in previous chapters, temperature is a vital parameter for monitoring sudden changes and predicting cell instability. One possible solution is to be able to have input datasets that describe as many situations as possible. However, this is a procedure that requires a considerable expense in terms of time, since it

would involve filling tables with experimentally measured values. However, this can lead to a mathematical model capable of calculating the soc more precisely in non-standard conditions. This would favor the implementation of techniques for good battery management such as cell balancing with control criterion based on the state of charge of the battery. The feasibility of solutions of this type increases if the hardware design of the ECU is carried out according to the needs of the designer. Currently the system has been implemented using demo boards that mount the micro controllers that were needed. These modules offer the ability to create a system without the need for hardware development, reducing costs and saving time, provided you have a system not tailored to implement any specific application. A further possible improvement is the possibility of evaluating the State of Health of a cell, which together with the SoC, are very useful parameters to understand how much the battery is able to deliver in terms of capacity and how many number of cycles charge and discharge is able to withstand.

Bibliography

[1] Battery management systems (bms). URL `https://www.mpoweruk.com/bms.htm`.

[2] Giuseppe Luca Amoroso. Realizzazione di algoritmo per la stima dello stato di carica di batterie al litio con identificazione online dei parametri. Master's thesis, Università di Pisa, 2015.

[3] Ines Baccouche, Sabeur Jemmali, Asma Mlayah, Bilal Manai, Najoua Essoukri, and Ben Amara. Implementation of an improved coulomb-counting algorithm based on a piecewise soc-ocv relationship for soc estimation of li-ion battery. *International journal of renewable energy research*, June 2017.

[4] *BSI ISO-11898-2 Road vehicles - Controller Area Network*. British Standards Institution, December 2016.

[5] *BSI ISO-6469-3 Electrically propelled road vehicles - Safety specifications*. British Standards Institution, November 2021.

[6] Cristiano Casadei. Apprendimento supervisionato-la regressione polinomiale e la regolarizzazione. *Maggioli developers*, Maggio 2019.

[7] K.W.E. Cheng, B.P. Divakar, Hongjie Wu, Kai Ding, and Ho Fai Ho. Battery-management system (bms) and soc development for electrical vehicles. *IEEE transactions on vehicular technology*, Vol. 60, January 2011.

[8] Matteo Corno, Nimitt Bhatt, Sergio M.Savaresi, and Michel Verhaegen. Electrochemical model-based state of charge estimation for li-ion cells. December 2014.

[9] Eletropedia. Lithium battery failures, . URL https://www.mpoweruk.com/lithium_failures.htm#soc.

[10] Eletropedia. State of charge (soc) determination, . URL https://www.mpoweruk.com/soc.htm.

[11] Eletropedia. State of health (soh) determination, . URL https://www.mpoweruk.com/soh.htm#top.

[12] Fadlaoui Elmahdi. Fitting the ocv-soc relationship of a battery lithium-ion using genetic algorithm method. *EDP Sciences*, 2020.

[13] Henry Omar Sarmiento-Maldonado Juan Pablo Rivera-Barrera, Nicolás Muñoz-Galeano. Soc estimation for lithium-ion batteries: Review and future challenges. *Electronics MDPI*, November 2017.

[14] Barrie Lawson. A software configurable battery. URL https://www.mpoweruk.com/Software_Configurable_Battery.htm#balancing.

[15] *Automotive current transducer open loop technology.* LEM, 2018 March.

[16] Leandro Malara. Analysis and modeling of an energy management system in automotive environment. Master's thesis, Politecnico di Torino, Marzo 2020.

[17] *Lithium-ion rechargeable battery.* MOLICEL.

[18] Martin Murnane and Adel Ghazel. A closer look at state of charge (soc) and state of health (soh) estimation techniques for batteries. *Analog devices.*

[19] Davide Nardini. Train, validation, test: cosa sono e come si usano nel machine learning. *Pulp Learning*, Novembre 2020.

[20] Java T Point. Can (controller area network) protocol. URL https://www.javatpoint.com/can-protocol.

[21] Aswinth Raj. Cell balancing techniques and how to use them. URL https://circuitdigest.com/article/cell-balancing-techniques-and-how-to-use-them.

[22] Ajit Sharma. Enabling the electric future of mobility: Robotic automation for electric vehicle battery assembly. *IEEE Access*, page 32, September 2019.

[23] *bq76PL455A-Q1 16-Cell EV/HEV Integrated Battery Monitor and Protector*. Texas Instruments, 2015 April.

[24] Analytics Vidhya. An introduction to simple linear regression. URL `https://www.analyticsvidhya.com/blog/2021/07/an-introduction-to-linear-regression/`.