Battery Management System applied to Projecto FST's EV prototype

Bruno Manuel Baraças dos Santos

Thesis to obtain the Master of Science Degree in

Aerospace Engineering

Supervisors: Prof. Dr. Rui Manuel Rodrigues Rocha
Prof. Dr. Moisés Simões Piedade

Examination Committee

Chairperson:	Prof. Dr. João Manuel Lage de Miranda Lemos
Supervisor:	Prof. Dr. Rui Manuel Rodrigues Rocha
Member of the Committee:	Prof. Dr. António Joaquim dos Santos Romão Serralheiro

October 2014

Agradecimentos / Acknowledgments

Devo uma palavra de agradecimento, em primeiro lugar, aos professores que me orientaram. O professor Moisés Piedade cuja orientação começou bem antes deste projecto e ao professor Rui Rocha que tantas vezes deu uma nova perspectiva e me obrigou a pensar em novos problemas. Aos dois um obrigado.

Ao Daniel Pinho, Pedro Oliveira, Miguel Silva, Rui Andrade e Miguel Guedes, tenho a agradecer o seu envolvimento directo nos BMSs originais que precederam e tanto acrescentaram ao aqui proposto. Agradeço também ao Pedro Costa e à Mariana Cunha, em particular, pela enorme ajuda na recuperação da bateria e pelo empenho em trabalhar nas soluções futuras aqui apresentadas. E, com receio de excluir alguém, agradeço a todos os restantes membros do Projecto FST pelo trabalho em torno de um objectivo comum sem o qual não poderia existir esta tese.

Finalmente, agradeço a todos os que através da sua constante amizade e amor me motivaram e suportaram ao longo de mais esta jornada e que tantas vezes suportaram a minha ausência.

Last but not least, a thank you note in English to fellow Formula Student teams *StarkStrom Augsburg e.V.*, *AMZ Racing* and *ETSEIB Motorsport*. Specifically, Robert Dollinger, Fabio Widmer and Guifré Vendrell respectively, for the contributions that enriched this work.

Resumo

Esta tese tem como objectivo projectar, construir e testar um novo Sistema de Gestão de Baterias (BMS) para os protótipos mais recentes do Projecto Formula Student Técnico (Projecto FST), em particular o actual *FST-05e* e o futuro *FST-06e*. O novo sistema deve responder a alterações chave na tecnologia empregue pela equipa Projecto FST em sintonia com os regulamentos aplicáveis da competição de Formula Student (FS).

Para cada um dos protótipos mencionados, a equipa desenvolveu / está a desenvolver um Veículo Eléctrico (EV) com uma bateria de 600 V composta por células de lítio tipo pouch, mas com configurações potencialmente diferentes — dependente da capacidade das células escolhidas e outras especificações do carro como a geometria da bateria. Com um sistema tão delicado como este, o sistema BMS requer fiabilidade e segurança em condições de operação bastante ruidosas e com bastante vibração como é o caso de um veículo, e mais ainda devido à natureza de competição do projecto. Ao mesmo tempo, o sistema deve ainda obedecer a um orçamento limitado, satisfazer os constrangimentos de espaço (e peso) e disponibilizar todas as interfaces apropriadas.

A solução proposta consiste num BMS híbrido com características de sistema distribuído e modular composto por um módulo mestre e vários módulos escravos que comunicam através de um barramento Controller Area Network (CAN). O sistema possibilita ainda a existência de vários contentores de baterias, podendo por isso haver mais do que um módulo mestre no sistema completo.

Keywords: Sistema de Gestão de Baterias, Vehículo Eléctrico, Segurança de Baterias, Baterias de Lítio, Formula Student

Abstract

This thesis has the purpose of designing, building and testing a new Battery Management System (BMS) for the recent electric prototypes designed by Projecto Formula Student Técnico (Projecto FST), in particular the current *FST-05e* and the future *FST-06e*. The new design should accommodate core changes in the technology used by the Projecto FST team in accordance with the applicable rules from the Formula Student (FS) competition.

In both mentioned prototypes, the team developed / is developing a battery powered Electric Vehicle (EV) with a 600 V battery of pouch lithium cells, but potentially different configurations — dependent on cell capacity and other car specifications like battery geometry. With such a sensitive system, the BMS needs to be very reliable and safe, even in a noisy environment and subject to vibrations as is the case of a vehicle, even more due to the competition nature of the project. At the same time, the system should still fit the budget of the team as well as comply with space (and weight) limits and provide all the appropriate interfaces.

The solution presented here is a hybrid BMS with modular and distributed features composed by one master module and several slave modules communicating through a Controller Area Network (CAN) bus. This system also allows for more than one battery container, therefore more than one master module may be present in the complete system.

Keywords: Battery Management System, Electrical Vehicle, Battery Safety, Lithium Battery, Formula Student

Contents

List of Figures

List of Tables

Abbreviations

ADC	Analogue-Digital Converter
AIR	Accumulator Insulator Relay
ASCII	American Standard Code for Information Interchange
BFRP	Basalt Fiber Reinforced Plastic
BMS	Battery Management System
CAD	Computer Assisted Design
CAN	Controller Area Network
CCV	Closed Circuit Voltage
CFRP	Carbon Fiber Reinforced Plastic
CRC	Cyclic Redundancy Check
DC	Direct current
DC/DC	Direct current (DC) to DC converter
DLC	Data Length Code
DoD	Depth of Discharge
EEPROM	Electrically Erasable Programmable Read Only Memory
EMI	Electromagnetic Interference
EV	Electric Vehicle
FS	Formula Student
FSAE	Formula Society of Automotive Engineers (SAE)
GLVS	Grounded Low Voltage System
GNU	GNU is Not Unix
GUI	Graphical User Interface
I/O	Input/Output
I^2C	Inter-Integrated Circuit

IC	Integrated Circuit
ID	Identifier
IMechE	Institution of Mechanical Engineers
IST	Instituto Superior Técnico
LDO	Low-dropout regulator
LED	Light Emitting Device
$LiCoO_2$	Lithium Cobalt Oxide
$LiFePO_4$	Lithium Iron Phosphate
LiPO	Lithium-Ion Polymer
MPU	Microprocessor Unit
NTC	Negative Temperature Coefficient thermistor
OCV	Open Circuit Voltage
PCB	Printed Circuit Board
Projecto FST	Projecto Formula Student Técnico
PWM	Pulse-Width Modulation
RTCC	Real Time Clock & Calendar
RTOS	Real Time Operating System
SAE	Society of Automotive Engineers
SD card	Secure Digital Card
SMD	Surface Mounted Device
SoC	State of Charge
SoH	State of Health
SPI	Serial Peripheral Interface
TS	Tractive System
TSAL	Tractive System Active Light
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
VIP	Vacuum Infusion Process

Symbols

C	Capacity
I_b	Balancing current
I_L	Load current
I_m	Measurement current
R_b	Busbar resistance
R_c	contact resistance
R_i	Internal resistance of cell / battery
R_L	Load resistance
SDC	Self-discharge current
SDR	Self-discharge rate
V_{max}	Maximum cell voltage
V_{min}	Minimum cell voltage
V_c	Voltage drop through a contact
V_m	Measured voltage
V_{pol}	Voltage polarization of a cell through (dis)charging
V_{cell}	Cell real voltage (Open Circuit Voltage (OCV))

Chapter 1

Introduction

Electric propulsion is arguably the foremost green propulsion technology and despite its current drawbacks it's a fast becoming alternative against conventional powertrains. The battery is still one of the most limiting systems in any Electric Vehicle (EV), but there has been huge improvements in this area, especially regarding the advances in lithium battery technology which is why there have been increasing success cases of battery powered vehicles. These batteries yield a large increment in energy density mitigating two of the greatest shortcomings of EVs: the short range and poor performance. This advance comes with a drawback given the higher maintenance and safety concerns with this type of cells.

In fact this technology is dangerous enough that, even after more than two decades in the market and lots of progress and innovations in products from cellphones and laptops to EVs and planes, there are still reports of dangerous accidents and difficult to extinguish fires in such systems. Three *Tesla Model S* cars burning last year in less than two months [Green Car Reports, 2013] and two fires aboard the relatively new *Boeing 787* [Wikipedia, 2014], are a testament to the general danger of this cell chemistry. Still, it's no surprise that especially for competition purposes, this is the chemistry of choice for Formula Student (FS) competitions. Here EVs compete side by side with internal combustion prototypes and already claim the best positions all around in the latter years.

This thesis is made in a partnership with one of these teams — Projecto Formula Student Técnico (Projecto FST), from Instituto Superior Técnico (IST) — with the goal of rehabilitating its last prototype's battery and building the technology for future cars developing the required systems in-house. These prototypes are destined to compete in events organized by several engineers societies such as the Society of Automotive Engineers (SAE) or the Institution of Mechanical Engineers (IMechE), targeted at student teams from universities from the whole world. These competitions put the teams to the test in designing and building a formula car prototype against a set of rules, actual on-track performance, and all the management and business side of developing such project.

The dynamic events these prototypes run range from a short straight line (acceleration event) to a narrow and demanding track in which the cars have to run a total of 22 km (endurance event). For such electric car, the battery has to have a high capacity and voltage, increasing the dangers associated with any battery. One of the most important aspects in these batteries is therefore the monitoring system which assures the correct operation of the batteries, keeping them within their normal operation limits.

In particular, a Battery Management System (BMS) has the objective of monitoring voltage and temperature for a cell or group of cells. Usually, it also enables the safe recharging and discharging of the monitored cells within safe limits. Such systems are able to take preventive actions and avoid permanent damage to the battery or any surrounding system. For the purpose of this thesis, a BMS system comprises all of these functions since they are all needed for the intended application.

Figure 1.1: *FST-05e* and team in Germany 2013. [Courtesy of Projecto FST]

1.1 Motivation

In 2013 Projecto FST went to three competitions but faced too many difficulties in several fronts that prevented the fifth prototype from running. One such difficulty was with the then new BMS system which failed to monitor the battery well enough. For that reason, the team decided to redesign the BMS for the next car, but having *FST-05e* rehabilitation as a milestone in the process. This would allow the prototype to have its official on-track debut in Silverstone, United Kingdom, in July 2014, and have the new battery system well positioned for the 2015 season with *FST-06e*.

The main shortcomings of this previous BMS design were mainly due to delays and consequential lack of testing, but also from design options that were proven to be flawed or not fully developed. Specifically, the voltage measurements in this system were dependent on individual calibration for each of the 144 channels in the whole battery and was still subject to big drifts in measurements. For that reason the team was always unsure of the real state of the batteries and the usable capacity of the battery was severely harmed from the increased restrictions we had to set in the BMS software.

The late delivery was also a problem that ultimately led to the software component never being fully developed. By the end of the 2013 competitions the system was merely a short lived "tech demo", and one that was proven unfit for the application.

From here arose the necessity of a new system that traded some of the previous features and design options for reliability, ease and speed of implementation.

While there are commercially available solutions, adopting those would restrict the ownership and fine control the team demands from such system, for both packaging and feature reasons. There is however an open hardware BMS: BMSafe [Poly eRacing, 2011]. Unfortunately, to date, it makes use of tools and hardware with which the team is unfamiliar, it's poorly documented and it has a significant lack of features. More importantly though, it's unmaintained (publicly) for two years and there is no performance data available.

It was then necessary to develop a custom BMS that was able to satisfy the team's specifications. Given the architecture always relying on a master module and the previous master module being sufficient to comply with the rules, a plan was worked out with the team to invest in the slave modules until the 2014 competition and limit the master functionality to what was already present in previous hardware.

Figure 1.2: *FST-06e* render for Class 2 competition in Silverstone 2014. [Courtesy of Projecto FST]

The master module could still make use of new features, that being the reason it was still redesigned in a later process even though most of the software was already functional with the previous one, minimizing the risk of developing such system.

1.2 Project scope and objective

The objective of this work is the development of an all new BMS to replace the previous generation in the current and second electric prototype, *FST-05e*, and set a clear road towards implementation in future ones, specifically the *FST-06e*.

This thesis started from identifying any fault or shortcoming with the previous hardware, software and container combination in *FST-05e* to pin-point where the focus of the new work should be. This encompasses the new BMS (subject of this thesis) and the battery container as well.

The main objectives until July (Silverstone competition with *FST-05e*, *FSUK2014*) were therefore to investigate the appropriate specifications and design a slave module that would fit them. At the same time, the software should integrate these new slaves with the previous master module. This way, rather than designing a full system upfront, the new master module hardware could be delayed until after the competition with greater care put into the slaves and the software integration.

Also, all the control and safety features predicted in Formula SAE (FSAE) 2014 rules [SAE, 2014] and / or necessary for the safety of the system and to comply with the team's goals should be well implemented and tested before the competition. This includes some peripherals such as an interface that exposes all voltages and temperatures in a safe manner.

From this point, new features that weren't possible to develop before July — because of time constraints but also because of inherent limitations with *FST-05e* — should then be developed targeting integration in *FST-06e*. The sixth prototype is scheduled for completion early 2015, and by the end of 2014 this work should be ready to be relayed on the electronics and propulsion team of Projecto FST. Therefore, good and throughout documentation should also be provided. Given the calendar of this thesis, this work is only partially covered here.

Time on track being as valuable as it is, design should focus primarily on critical features. Phasing in

the system for early integration and testing is a main focus of this work.

Finally, the system should not significantly increase the cost of previous ones. The best appropriate reference being roughly 1100 € for the complete BMS system[1] with spare parts but without labor and cable harness from the *FST-04e* prototype [Guedes, 2011].

With this approach, and the tools developed, the new BMS should provide a future proof solution for a broad range of battery configurations. Additionally, a unified code base and hardware development should yield a greater maintainability and longevity of the prototypes with great benefits to design iterations.

All the developed hardware and software referenced throughout this document are hosted at https://bitbucket.org/projectofst with the prefixes *FST BMS* and *FST CAN tools*. All sources are distributed under a GNU GPLv2 license.

1.3 System definition

This work consists on the development and assembly of all the required modules to manage the battery pack(s). These include a module responsible for taking measurements and ensuring the correct balancing strategy to keep the cells working within their limits, but also to maximize the available capacity of the whole pack — these will be called *slave* modules from this point onwards. In *FST-05e* there are 6 stacks of 24 cells in series and 2 in parallel, requiring a total of 144 voltage measuring points, and in *FST-06e* there will be 12 stacks of 12 cells in series, a compromise should then be made in regard to the chosen architecture and how many measuring points each slave should have.

To manage the slave modules, a *master* module is required. Also, the master has the ability and duty of managing a set of relays that control the availability of high power to the Tractive System (TS), and a soft charge and discharge of the capacitors within the motor controllers. Finally, this module is responsible for all the communications with the exterior of the battery, from which everything else needs to be galvanically isolated.

In *FST-05e* there's only one battery container, but the team requires this container to be divided in two for the next prototype. This means that the system should support more than one master module, each with its own responsibilities within the respective battery container and the car. This increases the complexity of the system and care should be taken to keep one master from overruling the other or taking action without mutual agreement, depending on the task.

Each module, both slaves and masters, is a programmable unit that communicates in a dedicated Controller Area Network (CAN) network, with the master(s) being the only module(s) with access to an external CAN bus providing an isolated interface to the battery. This second CAN bus allows communications with a Graphical User Interface (GUI) running on a computer and with a *charger* module, responsible for the control of a stock racket 3 kW power supply to charge the battery.

1.4 Structure of the thesis

This thesis is divided in eight chapters, the introduction being the first one. The remaining document follows the structure:

- *Chapter 2:*

[1] Actual complete system with cells and container is necessarily more expensive given the technology being targeted. Throughout this work, cost will refer only to the BMS system.

Technical aspects of a BMS in general. Summary of the available technologies and their purposes.

There's also a brief overview of solutions from other formula student teams.

◇ *Chapter 3:*

An in-depth analysis of the the problems requiring a solution and specifications for the proposed BMS. The desired architecture of the BMS system, its peripheral modules and interfaces are discussed as well.

◇ *Chapters 4 and 5:*

On-board system (slave and master) is detailed, both the hardware and the software as well as details on some compromises being made.

◇ *Chapter 6:*

A set of tools required to debug and otherwise interface with the battery from a computer is introduced.

◇ *Chapter 7:*

Details on the actual usage of the system in laboratory and on-track.

◇ *Chapter 8:*

Final considerations regarding the chosen design, evaluation of attained objectives and description of possible / necessary future work.

Chapter 2

State of the art

BMS systems have existed for a long time with different purposes depending on the cell chemistry in use and the configuration of the monitored pack. Many batteries don't even need a monitoring system on-board for their inherent safety such as lead-acid batteries, even though chargers and "fuel gauges" or meters are often needed to fully maximize their potential.

The currently optimal chemistry for this project is a variant of Lithium-Ion Polymer (LiPO), given the higher energy densities and C rates[1] when compared with other allowed battery technologies by FSAE. All of the following sections will assume this chemistry when referring to batteries unless noted otherwise.

Therefore, the next sections provide an overview of this particular type of batteries and insight on the available BMS technology. Still it's worth noting some or all of these systems are also relevant to some extent to other battery chemistries.

A more comprehensive overview of the state of the art in regard to batteries and BMS systems is presented for instance in Andrea [2010].

2.1 Battery technology

Lithium-ion batteries have been in extensive use in small devices like cellphones and laptops for a long time, typical systems having less than ten cells. Bigger applications are also increasingly common such as recent consumer EVs — e.g. *Tesla Model S*, *Opel Ampera* or *Chevrolet Volt* [CHEVROLET, 2014; Opel, 2014; Tesla Motors, 2014]. In fact, lithium-ion batteries are rapidly becoming standard in all sorts of consumer applications following as well as driving the reduced costs of fabrication.

These cells provide great energy densities, especially LiPO cells which replace the hard shell by a soft pouch with a polymer separator between the electrodes, thus reducing the weight[2]. Moreover, lithium based batteries have some of the lowest self discharge rates, i.e. the rate at which a cell looses charge in storage conditions.

Projecto FST wanted a competitive car with a demanding target weight of 200 kg for *FST-05e* and even stricter for *FST-06e*, therefore all considered cells are LiPO variants (see figure 2.1b). These come with a cost premium attached though, but as stated before lessened by the generalized demand and improved production

[1] The charge and discharge rate at which the cell may be used safely. For instance, for a C=5 A h, a cell rated at 20C nominal discharge can withstand a discharge of 20×5=100 A continuously.

[2] Another kind of lithium polymer cells, not to be confused with LiPO, actually have a polymer (solid) electrolyte, but those have currently issues that prevent their introduction in the market. In this document, LiPO denotes the first kind of cells, not the second.

(a) *Headway 38120S* cylindrical lithium-ion cell, as used in *FST-04e* (120×38mm).

(b) *EPS4500XP* LiPO (pouch) cell, as used in *FST-05e* (152×44mm).

Figure 2.1: Two types of lithium-ion cells. Not to scale.

processes.

These cells are structurally prone to damage with the delamination of the electrodes from the polymer sheet, increasing internal resistance and reducing capacity. Besides, physical damage is also associated with some cases of swelling and spillage of the electrolyte. For this reason, LiPO cells have greater packaging issues, but the saved weight and volume is nonetheless important for small applications like cellphones and high performance applications in general.

These are currently some of the most challenging batteries to manage for their sensitivity to overcharges, deep discharges and temperature issues, making them a primary target of BMS manufacturers. Actually these cells are famous for their thermal runaways usually resulting in explosions and / or hard to extinguish fires. These malfunctions are often times caused by physical abuse, but also from overcurrents and overcharging.

Additionally, cells overly discharged rarely accept energy again, effectively becoming unusable — note however that the residual energy required at all times to avoid this is not part of the stated capacity, not counting towards the energy density calculations.

Depending on the composition, LiPO cells have slight variations of nominal voltage, capacity and C rates, but large variations of degradation rates and inherent safety. The best performing chemistries sacrifice lifespan and safety for larger C rates and capacity, but new fabrication processes are constantly pushing these limits. Still, all of them pose the same problems regarding the required BMS features to manage such batteries and the differences are usually small enough that a single BMS model is able to accommodate all chemistries.

One of the highest performing chemistry is Lithium Cobalt Oxide ($LiCoO_2$), being the chosen technology for *FST-05e* and *FST-06e* over the safer yet more limited Lithium Iron Phosphate ($LiFePO_4$) used in *FST-04e*. These offer the best energy densities and C rates of the lithium-ion family, and are the main target of the proposed BMS system.

2.2 BMS features

BMS systems have a huge range of functionalities and may implement those in different ways. The first distinction to be made is between analogue and digital BMSs.

The superior and most common option is digital, which allows easier (re)configuration and integration, besides the collection of invaluable data to be recorded and / or transmitted to other interfaces. Additionally, if there is a fault within the battery, a well designed digital BMS is capable of identifying where the fault was detected (e.g. which cell is above some threshold) and by how much (e.g. by how many volts the cell is above its limit). Another important feature of digital BMSs is the ability to report the level of charge still available from the battery, usually an important feature of any battery powered device.

In contrast, an analogue system may identify every fault but is typically very limited in informing the user or a peripheral module of which fault occurred, where and by how much. On top of that, changing a configuration (e.g. adjusting the overvoltage threshold) is a hardware adjustment that has to be made individually for each channel.

Analogue chargers / balancers (a form of BMS) are still of great importance in standalone chargers for cylindric cells for instance. These devices are often targeted at a specific chemistry or use a selectable preset for each one, being generally cheaper than equivalent digital chargers. Still, LiPO cells demand that a BMS is present at all times and not only for charging. It's no surprise then that analogue BMSs are not that common for LiPO cells.

Finally, among both categories there are a few kinds of BMS systems beside cell balancers, each with different features as seen in table 2.1.

Table 2.1: Distinction between BMS types; red column only applies to digital BMSs.

BMS type	***Balances cells***	***Requests that battery be switched off***	***Capable of shutting down battery***	***Reports individual cell voltages, etc.***
Protector	✓	✓	✓	✓
Balancer	✓	✓		✓
Monitor		✓		✓
Meter / Fuel gauge				✓

A complete LiPO BMS system offers all of the summarized features, as it is usually the case with cell packs that are sold as a final assembled product. However, there's a huge need for custom specifications, for which reason there are BMSs that only offer certain features leaving others to be supplemented by the buyer through their own systems.

An example where such BMS would be required is when the user needs to comply with certain unconventional regulations that wouldn't be supported by a complete commercial solution — e.g. FS competition rules.

2.2.1 Cell balancing

Batteries are often a set of individual cells connected in series and parallel to achieve the required voltage and capacity. Whenever there are more than one cell in series, the battery may become unbalanced, i.e. different cells may have different charge levels. Assuming cells with V_{min}=3.0 V and V_{max}=4.2 V, figure 2.2 illustrates a possible scenario of a battery charged to its fullest, yet well below its full capacity.

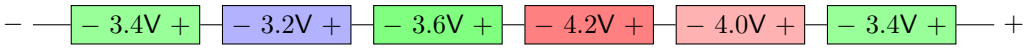


Figure 2.2: Example of an unbalanced array of cells.

This illustrates how a single cell in series hampers the ability to extract or store the nominal energy of the pack. The whole battery is 100 % discharged or charged whenever a single cell in series reaches its lower or upper voltage limit. With severe unbalancing of a battery, the actual usable capacity may be reduced to very low levels even if the cells themselves are in good conditions and performing within their specified limits. An extreme case would be a battery 100 % charged *and* discharged at the same time, not allowing any current in or out without risking permanent damage.

The reasons why a battery becomes unbalanced are differences in capacity and self discharge rate. These are usually very abrupt between different chemistries and manufacturers obviously, being one of the reasons why different cells shouldn't be used within the same system. Still, that's an avoidable problem, but even cells of the same model and from the same manufacturer are bound to have slight differences. Given enough time, these differences pose a balancing problem regardless of being actually common for manufacturers to match cells before shipping them.

Cell wear is another mechanism to unbalance a battery even further with the associated increased internal resistance and lower capacity. For a large battery it's not cost effective to throw all the cells away whenever a single one shows signs of wear. However, mixing older cells with new ones is equally bad for the battery even if it isn't damaging in any way to the cells as it would be in connecting them in parallel without being closely matched.

Furthermore, lithium-ion cells in particular age faster at certain voltage levels than others. Most suppliers recommend a minimum charge of 40 % for storage. Severe unbalancing may then contribute for a faster aging and degradation of the whole pack and the least charged cells in particular.

The balancing issue is therefore the reason why a BMS should have, or 'must' in the case of LiPO cells, ways of assessing each cell voltage individually. Leaving a single cell unmonitored means it could be completely unbalanced against the others and could lead to permanent damage to that cell or even the whole accumulator with potentially dangerous consequences.

To cope with cell wear and balancing problems in general, several mechanisms exist, some more sophisticated than others. An example of an advanced system would be a grid capable of taking one cell or group of cells offline, effectively emulating the action of disassembling a battery and removing the problematic cells. While these systems are the only option to deal with damaged cells without interrupting the power supply, they're not the ideal solution to balance cells, even more because of the infrastructure these systems require, making them only usable in stationary batteries where size and weight is not a concern.

To properly handle cell balancing though, there are two approaches: wasting excessive energy in cells that are more charged than the others and transferring energy from the most charged ones to the less charged. These techniques are often called *passive* and *active* balancing respectively, or *dissipative* and *non-dissipative*. The differences between the two are analyzed in the following subsections.

Common to both solutions though is the balancing effort required by a system: how much energy should be moved around or dissipated to keep a battery balanced? The answer to this question is part of the BMS system specifications and will be discussed in chapter 3.

2.2.1.1 Passive balancing

Passive balancing is based on the dissipation of energy, usually in the form of heat through resistors. This requires that the governing BMS identifies cells with more charge and connects them individually to a resistor or group of resistors until their charge is back within a certain difference to the least charged one. The result is a battery with lower charge yet capable of subsequently storing more charge than if only a single cell reached the upper limit instead of all of them.

While it's arguably wrong to waste energy from an environmental stand point, this solution is still better than producing and recycling batteries. Therefore, the greatest disadvantage is not the environmental impact but the heat dissipation, or lack thereof. In fact, the heat generated by such system may be harmful to cells, BMSs and even the resistors responsible for this dissipation.

However, this balancing strategy is easier to develop and implement, and cheaper since it uses only passive components like resistors. Also, the BMS may not have any extra connection to the cells since the voltage sensing connections may be used for balancing purposes. Finally, the added volume and weight required for the refrigeration systems (when needed) aren't meaningful compared to the volume and weight required by active solutions.

2.2.1.2 Active balancing

This solution is by far the most complex and there are several ways to achieve it. The short term advantage is that it moves energy around through Direct current (DC) to DC converters (DC/DCs) instead of dissipating all that energy.

This characteristic is very important in certain EV applications allowing, for instance, a car to be parked for a few hours and have a boost in available energy once the owner returns, without any actual charging of the vehicle. This is due to the BMS usually shutting down the battery once the first cell is depleted or nearly so. Since the least charged cells receive some energy from the most charged ones, this results in an increased available capacity despite actual stored energy being obviously lower by the end of the balancing process.

Also obvious, this process takes time, so for instance if the EV in question is not a convenience vehicle and has no down times between cycles, this system may not yield any advantage from a usability point of view since it won't balance the batteries quickly enough, as discussed later. That wouldn't take away the advantages in the charging process and thermal management though.

The main disadvantages of these solutions are the price, the increased development difficulty, as well as the complex cable harness and extra peripheral devices that most of these systems require. Also, these systems aren't usually very efficient as a whole as their efficiency is highly conditional on the battery state, with actual balancing capabilities being very moderate. This may be circumvented with larger systems, but again at the cost of extra volume, weight and price.

There are three common configurations for active balancing systems: cell-to-battery, battery-to-cell and bidirectional. As the name implies, the difference is in the way energy is transferred from some cells to others, each allowing for several topologies with varying numbers of converters and switches.

Another alternative is a cell-to-cell mechanism, the usual implementations having cells feeding adjacent ones. This is easily the cheapest and smallest solution, but unfortunately it's only capable of very small balancing currents and has low efficiency. An overview of this and other mentioned configurations is shown in table 2.2.

However, despite passive and active balancing being concurrent features, the compromise in a battery design is not between the chosen balancing technique — unless the judgment is strictly environmental of course. It's rather a compromise between active balancing and increased battery capacity, since that's the

Table 2.2: Comparison of active balancing strategies in usual topologies. Actual implementations introduce some variations tough.

	Cell-to-battery	***Battery-to-cell***	***Bidirectional***	***Cell-to-cell***
Converter	Low to high voltage.	High to low voltage	Bidirectional.	Low to low voltage.
Operation	Feeds the battery when a cell is excessively charged.	Feeds a low capacity cell when battery is sufficiently charged.	Feeds the battery or the cells depending on ratio of excessively charged cells.	A cell feeds adjacent ones if more charged, else is fed by adjacent cells.
Advantages	◇ Very efficient. ◇ Switches are on the low voltage side. ◇ Most effective when only a minority of the cells have low charge (most converters active).	◇ Most effective when most cells have a low charge (most converters active).	◇ Effective regardless of bias towards most cells charged or uncharged.	◇ Small low voltage components. ◇ Easy integration within the BMS.
Disadvantages	◇ Bad efficiency / cost ratio.	◇ Bad efficiency / cost ratio. ◇ Requires high voltage switches and isolated control circuitry.	◇ Bad efficiency / cost ratio. ◇ Most complex solution: requires switches on both the low and high voltage side.	◇ Low efficiency. ◇ Very low balancing currents (slow balancing). ◇ Complex, if not integrated in the BMS.

advantage of this system.

In certain applications the available capacity is greatly increased with active balancing as in the car example mentioned earlier, but other applications may present a situation where adding extra cells to the battery is more cost, weight and/or volume effective, with different applications prioritizing different performance parameters.

Such an example is given in Andrea [2010] for an application where the only performance measure is price. The conclusion being that it's not cost effective to invest in active balancing if the battery is depleted in less than 20 minutes. This evaluation is derived from the average dollar per Watt cost of converters and dollar per Watt-hour cost of lithium batteries from 2010.

2.2.2 Performance parameters

There are three parameters usually determined for batteries: State of Charge (SoC), Depth of Discharge (DoD) and State of Health (SoH). These have different aims at providing information of how much 'fuel' is left and general health of a battery.

These parameters are determined in several ways, but some are not possible to use depending on chemistries and some even being destructive to the cell. For common purposes, on a live system, these are important performance indicators nevertheless, and several indirect ways of determining these parameters are possible. In this work, the methods of interest are those capable of working in live systems, particularly taking advantage of a digital BMS.

For lithium-ion cells, the indirect ways of determining them rely on measurements of voltage, current and temperature, but SoC, DoD and SoH have a very non-linear and interdependent relation with these and many other factors. In order to correlate these with the intended parameters, voltage measurements, current integration (Coulomb counting), self-learning algorithms, modeling and others or a combination of them are used. Extensive examples and analysis of most common methods are found in Pop et al. [2008] and Bergveld [2001] to name a few.

2.2.2.1 SoC and DoD

SoC is usually intended as a percentage of available charge in a cell or cell pack from 0 % to 100 %. For certain battery types, this poses no challenge as charge varies somewhat linearly with the voltage, but lithium-ion have a very non-linear relation between the two — see figure 2.3. The problem is actually worse since temperature, usage history and several other factors may have great influences in perceived voltage for any charge level.

> "Theoretical [SoC] calculations are based on Coulomb counting modified by the cell voltage and temperature, the rate at which cells have been charged and discharged, the chemical composition of the various active chemicals and any doping which has been used, the possibility and effects of contamination, the shape and length of the physical current paths within the cell, the volume of electrolyte, the thickness of the electrolyte and the separator, the resistivity of the components, the rate of mass transfer of the ions through the electrolyte, the rate of chemical action at the surface of the electrodes or the rate of absorption of the ions into the intercalation layers, the actual surface area of the electrodes, the effective surface area of the electrodes taking into account the particle sizes of the chemicals, the effect of passivation on the electrode surface, the ambient temperature, the Joule heating effect, the self discharge rate of the cells, the time between charges plus possibly several other factors."
>
> — Electropaedia [2014]

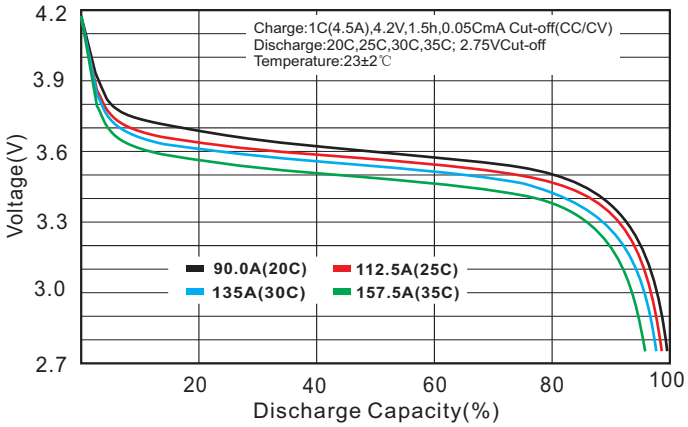


Figure 2.3: Discharge rate characteristics for LiPO cells EPS4500XP, used in *FST-05e*. [E Propulsion Systems, 2012]

As the quote suggests, even with a lot of parameters and measurements known for any given cell, a purely theoretical and universal approach is nearly impossible. However, for a lithium-ion cell, there are two SoC values possible to identify with 100 % accuracy in a very easy way: when its voltage reaches the lower or upper bound.

These bounds do not mean the cell is as charged as it can be — i.e. maximum capacity —, it only means that the cell cannot be discharged or charged further without cycling it or risking damage to the cell, which fits the definition of SoC. However, even this presumes the cell is in open circuit (Open Circuit Voltage (OCV) measurement) and that it had enough resting time to eliminate influences from the polarization of the cell.

Fortunately, for $LiCoO_2$ cells, Closed Circuit Voltage (CCV) differs only slightly from OCV, even for relatively high currents. This is due to the usually small internal resistance, but as a cell wears down this resistance may increase significantly. In such conditions, even a LiPO cell may be charged to its maximum of 4.2 V and some minutes or hours later, in open circuit, measure only 3.4 V, for instance.

The voltage difference may and should be modeled and accounted for, but a better method of avoiding such problem is to use Coulomb counting. This is a very common technique to cope with the problems of measuring accurate voltages and non-linearities between voltage and charge.

Finally, DoD is somewhat the opposite of the SoC giving a percentage of how much charge was already drained. Depending on the intention of the developer, DoD may go beyond 100 % — e.g. if a cell has higher capacity than nominal rating and Coulomb counting is used to determine the value. However, regardless of the implementation, these are usually similar parameters that are calculated with similar methods.

Table 2.3 summarizes available methods for SoC and DoD determination of lithium-ion cells although they may apply to other types of batteries.

2.2.2.2 SoH

A good SoH implementation should detect differences in cells, degradation relative to nominal characteristics and cells that have degraded unevenly since installation. The first applies to a battery whose performance is affected by using cells with dissimilar characteristics. An example of the second and third situations is a battery that has already made several full cycles and has consequently aged, evenly or not.

SoH determination may then need to be tuned to reject production margins. On the other hand, these

Table 2.3: SoC and DoD estimation methods for lithium-ion cells. [Piller et al., 2001; Pop et al., 2008]

Method	***Advantages***	***Disadvantages***
Discharge test	◇ Accurate ◇ Easy ◇ Independent of SoH	◇ Offline ◇ Time intensive ◇ Change of battery state ◇ Loss of energy
OCV	◇ Online ◇ Easy ◇ Cheap (it has to be provided for safety purposes already)	◇ Non-linearity ◇ May require some rest time after charging or discharging ◇ Demands high precision voltage measurements
Coulomb counting	◇ Online ◇ Accurate	◇ Requires several re-calibration points ◇ Depends on initial state (integration problem) ◇ Demands high precision current measurements ◇ Very sensitive to parasite reactions
Artificial neural network	◇ Online	◇ Requires training data from similar battery ◇ Expensive development
Fuzzy logic	◇ Online	◇ Requires a lot of memory in real-world application
Extended Kalman filter (state of the art)	◇ Online ◇ Dynamically adapts to errors and SoH	◇ Difficult to implement ◇ Requires suitable battery model

production margins may be significant enough that should impact the "health measure" of a battery. This is only one of the arbitrary aspects in the definition of this parameter.

Actually, SoH is by far the most arbitrary of the three parameters mentioned. For a single cell, its health is tied to the actual capacity in relation to the nominal value and variations in internal resistance[3]. Each has different implications, yet these are still interconnected. Therefore, for a single cell, it's difficult to define a single expressive performance value of health.

Moreover, while one can easily argue how the SoC of a cell influences the SoC of the battery, it's not as easy to define how the health of one cell influences the health of the pack. For this reason, any SoH parameter may refer to the cells or the battery, but it's not possible for a single value evaluation of the health of both.

To evaluate the health of a battery, the available methods are somewhat similar to those of table 2.3. The difference is in the parameter and meaning extracted from the measurements, models and filters.

Given the complexity of the problem, this work won't suggest any method to determine SoH. Still, all the logging features should allow collecting data for studying the problem and an eventual implementation.

2.3 BMS topology

A battery may have a large number of cells to monitor, as it is the case with Projecto FST's prototypes. There are several possible architectures for such system, each with advantages and disadvantages. The main ones are *Centralized*, *Distributed* and *Modular*. None of them is inherently better than the other and the right choice is a function of the application and other factors like development budget.

These architectures aren't unique. In fact there are options that combine features from each concept. The end result is naturally a compromised mixture of their qualities, but also of their shortcomings.

In the following sections, a summary of each architecture's main advantages and disadvantages is given.

2.3.1 Centralized

The most straightforward topology for any system is the centralized — see figure 2.4. Its simplicity often reduces cost and it's particularly suited for small batteries. Still, while the manufacturing cost is typically reduced, the same doesn't necessarily apply to the development cost and time. The BMSs represent the majority of analogue BMSs.

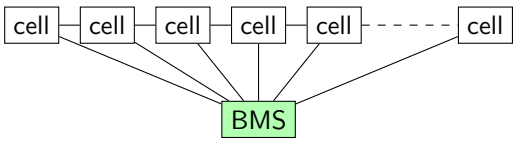


Figure 2.4: Centralized BMS.

Since every cell must connect both poles to a single Printed Circuit Board (PCB), large number of cells impose difficulties in the cable harness production, thus making them less common in such systems. To avoid wiring problems with the harness, the PCB may instead connect directly to the cells, but, again, it's hard to adapt to larger batteries.

Finally, the typical centralized BMS cannot be used in batteries of different sizes, or doing so would at least imply unused circuitry in the smaller ones. This is in fact the main disadvantage since these systems

[3] Physical damage also translates in a difference in capacity and/or internal resistance.

tend to require an extensive redesign for each battery.

From a control point of view, this solution is the easiest and most common for analogue BMSs, but it's also easier for digital ones: having only one Microprocessor Unit (MPU) simplifies the control loop and requires at most one communication channel (to the outside).

2.3.2 Distributed

The most radical departure from a centralized system is the distributed topology (figure 2.5). Generally these BMSs try to isolate cell monitoring from high level circuitry that coordinates simple modules — usually denominated *slaves* — and controls peripheral devices such as relays or fans through another (single) module — referred to as *master*.

Each slave module is autonomous as well as the master and require some sort of communication protocol between the master and each slave. Common interfaces are CAN, Serial Peripheral Interface (SPI), Inter-Integrated Circuit (I^2C), among several others.

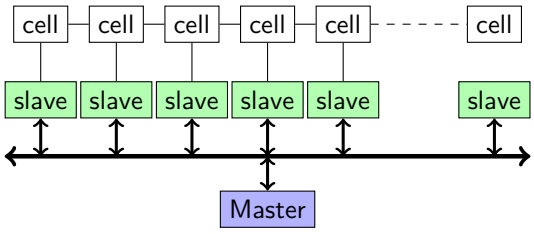


Figure 2.5: Distributed BMS.

Since each slave connects to a single cell, it's easier to ensure the correct assembly to prevent harness faults, but more importantly makes it easier to integrate the slave in the mechanical packaging without wires at all. In fact, wire faults in these systems often come from the bus connecting them to the master module and although it is possible to reduce the complexity of the cable harness with serializable communications, this makes it possible for more slaves becoming offline from a single wire fault.

This topology offers the greatest versatility in regards to the battery size provided the number of cells (and thus slaves) is within the maximum number a master can support. The trade-off is usually the initial development time, but, in case of a redesign, only one module may change at a time as long as the communication protocol remains compatible, which is not a hard task.

Despite this modularity being appropriate in batteries with larger cell counts, high voltage systems may easily require hundreds of slave modules. This leads to an increasing relative cost of the BMS and total volume of the system on top of assembly and packaging problems.

A successful example of a distributed BMS is the one present in the first Projecto FST's electrical prototype, *FST-04e*, and documented in Guedes [2011].

2.3.3 Modular

The modular topology is a combination of centralized and distributed systems. It consists on modules that monitor an arbitrary number of cells with varying levels of hierarchy.

The architecture obviously shares some versatility with distributed systems but usually only supports multiples of a certain number of cells. This constraint may be surpassed though with heterogeneous modules, i.e. all modules are able to coordinate themselves regardless of having the same number of cells to monitor or

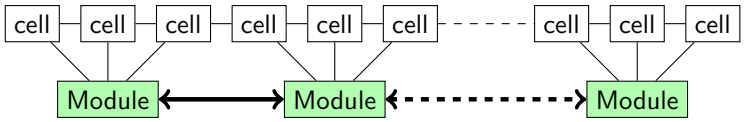


Figure 2.6: Modular BMS.

not. Unfortunately this effectively increases the development and production costs even more.

In cases of completely horizontal hierarchies, it also introduces the problem of responsibility: which module should control the relays; how should the exterior communications be handled and by which module; how should performance parameters be calculated and stored; etc. Still this is a problem that is dependent on specific needs and overall architecture.

In its simplest form, these are the most common BMS systems, only behind centralized BMSs — consequence of most battery applications being small, often with one to three cells. This is due to being generally cheaper than distributed BMSs but still modular enough to support batteries with different sizes and ease assembly and maintenance.

2.4 BMS in Formula Student

In 2014, the German FS competition alone saw 40 teams registered with electric cars, but there are plenty of others. It's natural that several solutions arise for the same (or at least similar) problems.

Table 2.4 shows a few examples of electric FS cars with corresponding specifications. While it represents only a small fraction of the FS universe, it correctly shows a trend towards distributed BMSs given the usually high cell count within these EVs.

Among the two commercial solutions represented in this table, one is in halted production and not available, the Li-BMS v3 [LION E-Mobility AG, 2014] — in blue in the table. It has a very similar architecture to the one being proposed.

The other one, in red in the table, is used by several teams. One of the main factors that make this system so attractive to FS teams is the excellent documentation provided online [elithion, 2014] and stability of the system which exists since 2008. Also for this reason, many of the specifications from elithion's Lithiumate™ pro were taken as a target reference for the proposed system.

As seen in chapter 3, the specifications from *FST-05e* and *FST-06e* are closer to these other systems than *FST-04e*. However, *FST-05e* is not presented in this table since the designed system was largely ineffective and most specifications had little or no correlation to its actual performance.

Table 2.4: Sample of BMSs used in electric FS cars. [Courtesy of AMZ Racing, Starktrom Augsburg e.V. and ETSEIB Motorsport]

	University	**TU Lisbon**	**ETH Zürich**	**UAS Augsburg**	**ETSEI Barcelona**
Model	*Team*	Projecto FST	AMZ Racing	StarkStrom Augsburg e.V.	ETSEIB Motorsport
	Car	*FST-04e*	Grimsel	Elinor	CAT07e
	Year	2011	2014	2014	2014
Battery	*Battery configuration*[a]	48s5p	112s4p	96s3p	140s2p (2×70s2p)
	Max. voltage	175.2 V	470 V	403.2 V	588 V
	Capacity	50 A h	15.6 A h	18.75 A h	12.7 A h
	Chemistry	$LiFePO_4$	Undisclosed	$LiCoO_2$	$LiCoO_2$
	Cell type	Cylindrical	Pouch	Undisclosed	Pouch
BMS	*Model*	Self-developed (see Guedes [2011])	Joint-developed with Elektromotus	Li-BMS v3 (prototype; limited production)	Elithion Lithiumate™ pro
	Topology	Distributed (1 slave per cell in series)	Distributed (1 slave per cell in series)	Central, modular or distributed (distributed used; 1 slave per 12 cells in series)	Distributed (1 slave per cell in series)
	Balancing	Passive; 2 A	Passive; up to 2 A (0.2 A used)	Passive; 0.25 A	Passive; up to 3 A (0.2 A used)
	Current sensor	Hall effect	Shunt resistor	Shunt resistor	Hall effect
	Internal bus(es)	CAN	4×CAN	2×CAN	CAN
	External bus(es)	CAN	CAN	CAN, USB, Ethernet, RS-232, I^2C (CAN used)	CAN, RS-232
	Approximated cost	1100 € (components only)	1500 € (components only)	Unknown	4200 €

[a] XsYp refers to X cells in series and Y in parallel.

Chapter 3

System architecture

The overall battery specifications are set by the *chassis*, *vehicle dynamics* and *electronics and powertrain* teams of Projecto FST. Namely the chassis team defines the geometry and ensures mechanical specifications, the vehicle dynamics team uses self-developed simulation packages to estimate the energy and power requirements, and finally the electronics and powertrain team is responsible for choosing the right cells and configuration to fulfill aforementioned specifications.

Relevant parameters are presented where needed, specifically in section 3.1, but this is obviously an iterative process which is beyond the scope of this thesis and such details like capacity determination and cell choice will be omitted. Instead, this thesis will focus on the task of providing the electronics and software that ensure the correct operation of the battery.

However, in a system that needs to be installed in more than one prototype with different cells, capacity, geometry and voltage, this process is tightly connected to the remaining design of the car. To cope with some of these problems, especially when *FST-05e* was already built at the start of this project, compromises were necessary to accommodate both designs — *FST-05e* and *FST-06e*.

The only upfront requirement for the proposed system is that the BMS must be digital. This is not only desirable for the motives presented in chapter 2, but also a silent imposition of FSAE rules — the rules require a set of features impossible to satisfy with an analogue BMS, like reporting each cell voltage and making such values available for technical inspection.

This chapter will detail the remaining requirements for the system. However, longer term objectives will only be discussed in chapter 8. Before any specifications though, an in depth overview of the vehicles targeted by this technology is presented in section 3.1.

3.1 System definition

Both *FST-05e* and *FST-06e* are designed to have a maximum of 600 V — also the maximum voltage allowed in FS competitions. Given the low voltage of single lithium-ion cells, this results in a very large array of cells: 144s2p for *FST-05e* and 144s1p for *FST-06e*.

Still it was an important requirement not to not limit the new BMS to these 600 V batteries, allowing instead to install it on smaller (or bigger) containers. This is due to the fact that *FST-06e* is designed to have the cell array split into two containers of 300 V maximum, but also to provide a more scalable solution for any future prototype.

Another important concept related to the battery internal architecture is the *stack* concept. FSAE rules

state that stacks of cells must be built to a maximum of 120 V or 6 MJ, whichever results in a smaller stack. These stacks are then to be separated by fireproof walls with some other structural constraints and connected by maintenance plugs that should be capable of a tool-less operation.

For maintenance reasons as well as for safety, the team quickly opted for relatively small stacks on *FST-05e* and even smaller in *FST-06e*. The configurations are 24s2p for the fifth prototype and 12s1p for the future one. This makes the battery a group of small replaceable stacks improving maintenance as well as reducing the risk of hazards while handling each stack or even the whole battery.

Finally, the container(s) is(are) to be installed in a vehicle in which all the electronics have a somewhat stable protocol for communications and data logging through CAN.

The following sections give an overview of parameters from each of these vehicles that are relevant in the energy accumulator integration. In *FST-05e* case, an in-depth analysis of the problems and limitations found with the design is also presented.

3.1.1 *FST-05e*

FST-05e's battery is a U-shaped container that is mounted below the car, outside of the monocoque chassis. The characteristics of the energy accumulator are resumed in table 3.1.

Table 3.1: *FST-05e* battery specifications.

Parameter	***Value***
Maximum voltage	600 V
Nominal voltage	532.8 V
Minimum voltage	432 V
Maximum peak current	630 A
Main fuse current	100 A
Maximum charging current	18 A
Capacity	9.5 A h
Total energy	5.7 kW h
Total number of cells	288
Cell configuration	144s2p
Number of stacks	6

The problems found with the battery 'ecosystem' in *FST-05e* were very broad, beyond the BMS itself. Some of these problems are bound to the design of the system, others to the actual implementation and all of them have important implications in the design of an alternative solution.

The following subsections focus on the analysis of: battery design, BMS design, implementation and deployment. These problems cover all aspects of the BMS system and peripherals and are relevant either in the design of the BMS electronics, its layout and fixation or the peripherals discussed in this thesis.

3.1.1.1 Battery design

The battery developed for the fifth prototype is shown in figure 3.1 as a Computer Assisted Design (CAD) render. In the picture, it's shown the positioning of the original BMS along with the CAN bus that connects each slave to the master module.

The container is entirely made from composite materials except for fixation points. It uses a combination

of Carbon Fiber Reinforced Plastic (CFRP) for structural rigidity and Basalt Fiber Reinforced Plastic (BFRP) for fire protection and electrical insulation, produced through Vacuum Infusion Process (VIP). Other materials used are high performance foam and copper mesh for structural and shielding/grounding purposes respectively.

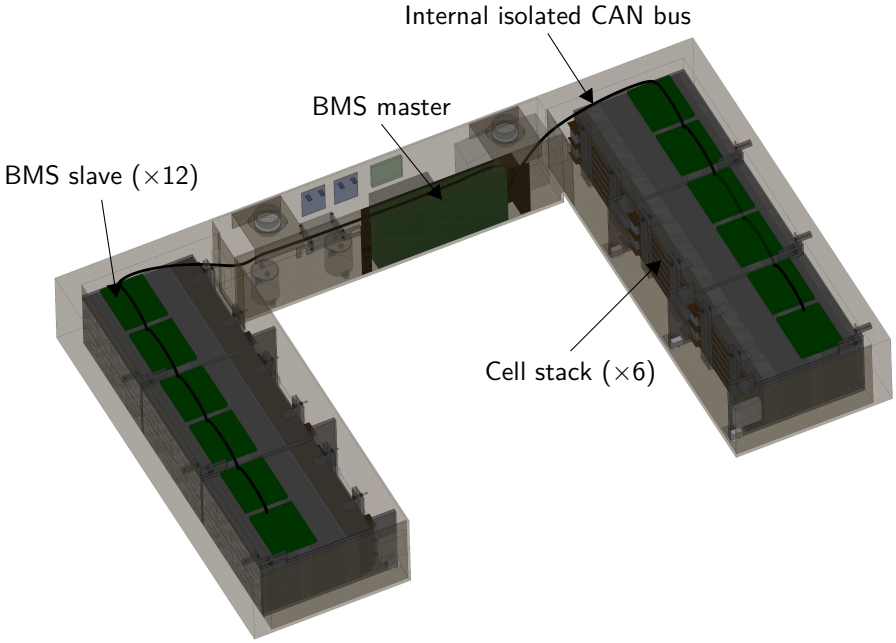


Figure 3.1: Render from the design phase of *FST-05e*'s battery. [Adapted from Projecto FST [2013*a*]]

These structural details have no impact on the BMS system. However, the cell stack geometry and assembly do. Figure 3.2 shows a complete cell stack, where the slave modules connect through the yellow wires to the cell poles and through the red ones to the Negative Temperature Coefficient thermistors (NTCs) that monitor the cells. Structural integrity of the stack is assured only when assembled within the container.

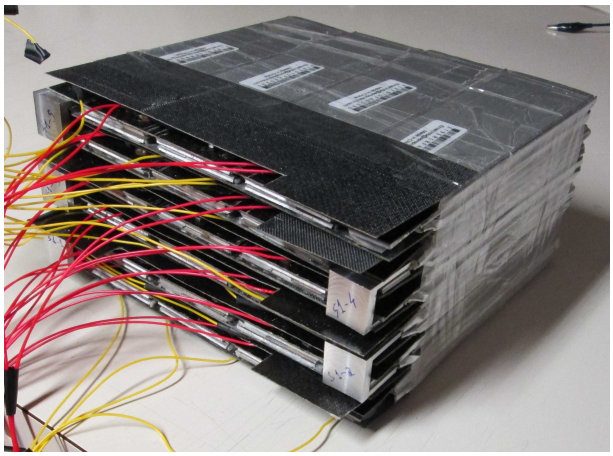
This stack has no refrigeration considerations since adiabatic tests with expected discharge and charge profiles for the chosen cells yielded temperature increases below 10 °C [Projecto FST, 2013*b*]. This is well within accepted working temperatures for these cells even in hot days. The team then decided not to install any active cooling in the battery.

While the cells themselves can withstand the projected conditions with no overheating, the same can't be said for the electronics. Namely, the original BMS was designed to use dissipative balancing, thus generating heat to balance the cells. With no way to dissipate the generated heat, the balancing was severely crippled. Therefore, opting against active cooling proved to be one of the flaws of this design.

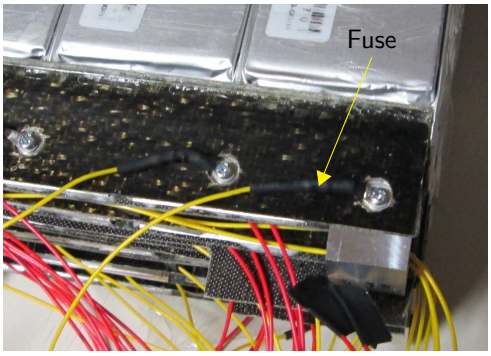
Another aspect of this design is the connections between each slave and corresponding cells being made through wires. This results in a large number of wires which makes assembly difficult and prone to errors. Also, the system is less reliable for that and maintenance procedure is very sensitive.

Last but not least, the BMS connections to monitor cell voltages are sub-optimal. Given the mechanical design of the stack, the voltage sensor (slave module) is unable to connect directly to the pads of the cells. Instead, they're connected through the high current busbars, which introduce an extra contact resistivity between the pad and the busbar itself. Since the measuring current is negligible ($I_m \approx 0$), this resistivity is never high enough to influence the measured voltage — i.e. the voltage drop across the contact resistivity R_c is approximately 0 as well (equation 3.1). However, the same does not apply to the load current, I_L.

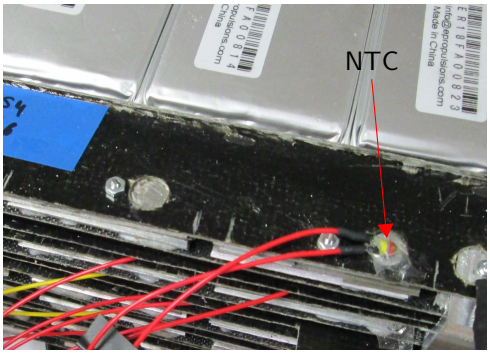
$$V = I \cdot R \tag{3.1}$$



(a) Cell stack from *FST-05e*.Cell connections are assured by small aluminium busbars and insulation by BFRP.



(b) Cell connections to BMS and protection fuse placement.



(c) Temperature sensors placement.

Figure 3.2: Cell stack from *FST-05e*. [Projecto FST, 2013*a*]

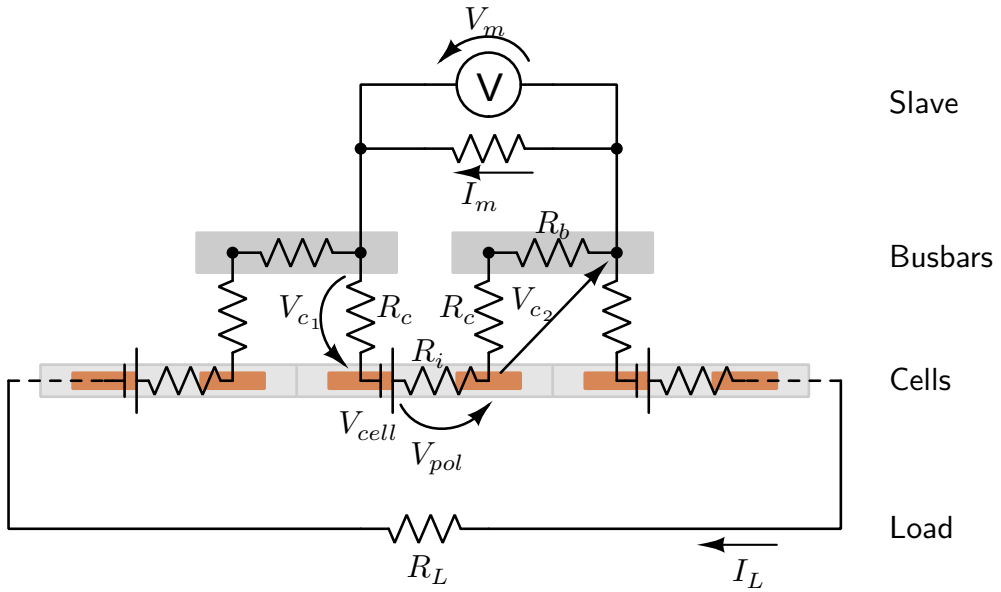


Figure 3.3: Analysis of connections between cells and BMS for *FST-05e*. For simplicity, the schematic illustrates a differential measurement.

Figure 3.3 shows an equivalent schematic of these connections. From Ohm's law (equation 3.1), if I_L is sufficiently large, even a small resistance will yield a significant voltage drop in the contact. The slave module then measures the cell voltage with an error dependent on the load current, I_L. The following equations show the relationship between cell voltage and measured voltage:

$$V_m = V_{cell} - V_{pol} - V_{c_i} \tag{3.2}$$
$$V_{c_1} = I_L \cdot R_c \tag{3.3}$$
$$V_{c_2} = I_L(R_c + R_b) \tag{3.4}$$
$$V_{pol} = I_L \cdot R_i \tag{3.5}$$

More so, the error would be present even if the cell was an ideal source, i.e. $V_{pol} = 0$ or $R_i = 0$. Assuming an optimistic situation with R_b=0, R_i=0, this yields an error of:

$$V_{cell_{\text{cell } i}} - V_{m_{\text{cell } i}} = V_{c_1} + V_{c_2} = I_L \cdot 2 \cdot R_c \tag{3.6}$$

If the slave doesn't read voltages differentially, the situation is even worse. Indeed, for the previous BMS, voltages were measured against the bottom cell of the group of 12. In figure 3.3, this would mean placing the multimeter across several cells for any cell but the first one.

For the largest voltage being measured, this would result in stacking the voltage drop in every connection and from the polarization of all 12 cells. With $V_{c_{\text{cell } i}} = V_{c_{1_{\text{cell } i}}} + V_{c_{2_{\text{cell } i}}}$ as sum of the voltage drops between each pole of the cell i to each probe of the voltage sensor and assuming $R_b = 0$, $R_i = 0$ and uniform contact resistance across all connections , the influence of I_L is now given by equation 3.7.

$$V_{c_{\text{cell } i}} = I_L(2 \cdot i \cdot R_c + i \cdot R_b + i \cdot R_i) \tag{3.7}$$

In this case, two consecutive measurements are required for any cell except the first one, i.e. the voltage of the 2$^{\text{nd}}$ cell is given by the difference between the 2$^{\text{nd}}$ and 1$^{\text{st}}$ cell measurements. With the same assumptions, this yields:

$$V_{cell_{\text{cell } i}} - V_{m_{\text{cell } i}} = V_{c_{\text{cell } i}} - V_{c_{\text{cell } i-1}} = I_L \cdot 2 \cdot R_c \tag{3.8}$$

The obtained error is equal relative to differential measurement for any cell, but that's assuming a constant current for two measurements. Assuming I_{L_i} as the current during each measurement, the error is given by equation 3.9.

$$\begin{aligned} V_{cell_{\text{cell } i}} - V_{m_{\text{cell } i}} &= V_{c_{1_{\text{cell } i}}} + V_{c_{2_{\text{cell } i}}} - V_{c_{1_{\text{cell } i-1}}} - V_{c_{2_{\text{cell } i-1}}} \\ &= 2 \cdot R_c \left[I_{L_i} \cdot i - I_{L_{i-1}}(i-1) \right] \\ &= 2 \cdot R_c \cdot I_{L_i} \left[i - \frac{I_{L_{i-1}}}{I_{L_i}}(i-1) \right] \end{aligned} \tag{3.9}$$

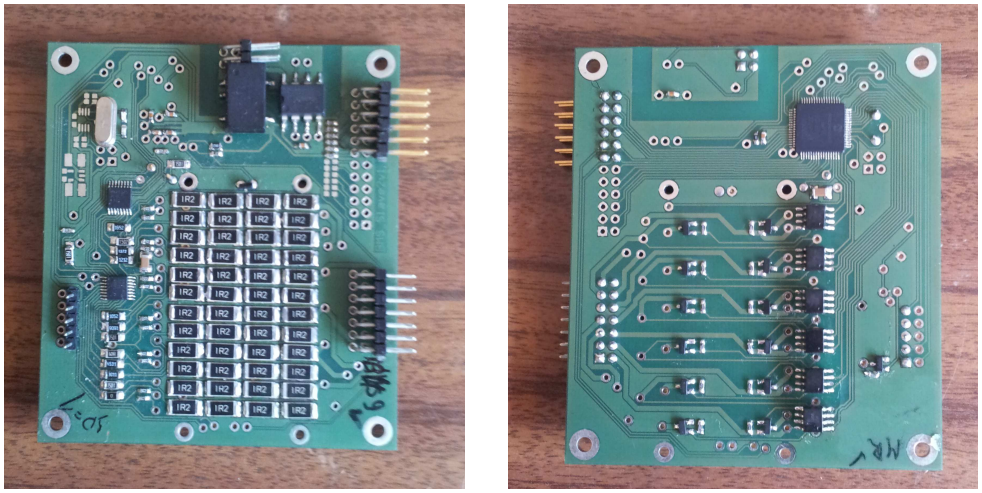
While this problem is still avoidable for differential measurements, it would only result in a uniform 'small' error across all cells. However, in the second case, it's not avoidable and any small variation in current between each measurement makes the error significantly larger and less predictable. Moreover, the second kind of error is solely dependent on BMS design.

3.1.1.2 BMS design

For the original BMS system, the chosen architecture consisted of a distributed BMS with each slave module monitoring a group of 12 cells.

In the design phase for the slave module, it was adopted a simplistic solution which used a single Analogue-Digital Converter (ADC) from the MPU to measure all voltages. For the voltage level conversion, all cells had to be multiplexed through an analogue multiplexer with a different voltage divider for each channel.

This is a consequence of measurements being made relative to lowest monitored potential, i.e. absolute measurements instead of differential. In itself, the choice for absolute measurements is automatically negative, for the reasons presented in section 3.1.1.1. But the method used to achieve that is equally troublesome.



(a) Top layer. (b) Bottom layer.

Figure 3.4: Original slave module for *FST-05e* (v2.1).

This approach was chosen for its simplicity and moderate compactness, but requires a lot more calibration and characterization work relative to integrated solutions that dominate the BMS market. With a total of 144 channels plus spare slave modules of 12 channels each, this is a complex and time consuming task.

Finally, the voltage dividers shared one resistor, meaning that the current drain from the cells under measurement was highest at the bottom of the stack and lowest at the top. The actual difference between the lowest and highest drain was $\cong$1 mA, which meant a permanent albeit slow unbalancing of the cells atop all other unbalancing factors — see section 3.7.2 for an analysis of expected unbalancing rate and their implications.

Regarding temperature management, the slave modules also had design issues since they were unable to properly read their own temperature[1] With only one NTC to monitor the PCB, it's impossible to monitor the temperature of the balancing load, MPU and other Integrated Circuits (ICs) or PCB zones independently.

The master module however, was composed mostly of CAN and Input/Output (I/O) hardware and code, having no major flaws in the implemented features. However, it lacked several features that made it unsafe or otherwise less capable. The lack of certain safety features resulted in most common installation faults being undetected by the system. They were only detectable after a catastrophic fault (burnt components) and/or tedious visual inspection giving room for human errors.

[1] It was still able to read a larger number of cell temperatures than required, and some of the sensor channels could be used to monitor the module. A recent rule change requiring more temperature sensors for the cells [SAE, 2014] invalidates this approach for this battery however.

3.1.1.3 Implementation and deployment

With the mentioned design issues, high measurement errors and temperature management problems were expected. Given the nature of the voltage dividers, it's no surprise that even with high precision resistors, this solution is still very sensitive to temperature fluctuations, humidity and requires a significant power to be drained from each cell compared to other solutions. All of these factors are detrimental to a good voltage measurement accuracy, further debilitated by the absolute measurements instead of differential and independent of the connections problem.

Indeed, even under optimal operation conditions — negligible load and laboratory conditions —, the slave modules shown sways in successive measurements of $\cong$200 mV. This value is the maximum error recommended for lithium-ion safety, but well above the maximum recommended for advanced features like SoC determination [Andrea, 2010]. However, there were also linearity problems, i.e. the 200 mV error was only obtainable after calibration for a given voltage level.

Also, since balancing was designed to generate a lot of heat in the slave PCB, the lack of temperature monitoring and dissipation led to disabling the balancing completely. Indeed, with both the errors and the lack of balancing, the battery was never properly charged. This prevented the prototype from racing and may have aged the cells from bad maintenance, requiring a lot of cell replacements since initial assemblage despite their 'lack of' use.

Another major problem in the implementation phase was in the difficulty of calibration and characterization of each module. This proved to be time consuming and as a result the software development was held back. Indeed the software was barely functional by the end of 2013 season.

The major problems with the software were:

- ◇ Communication protocol between modules inexistent.
- ◇ Almost no error handling; system was oversensitive.
- ◇ No estimation of SoC of any kind.
- ◇ No abstraction between slave and master modules leading to no scalability and waste of slave processing capabilities.
- ◇ No appropriate debug tools or interfaces.

3.1.2 *FST-06e*

For *FST-06e*, the battery is divided in two containers similar to the one present in *FST-05e*. The differences are in the absence of a 600 V/12 V DC/DC and pre-charge circuit inside the container. Also, the number of stacks in each container is the same since in *FST-06e* each stack is half the size of a *FST-05e* stack.

In contrast to *FST-05e*, these containers and specifically these stacks were developed alongside this work. Most of these problems regarding the battery design were therefore focus of attention by the team. Chapter 4 should cover design changes that hope to solve the identified problems.

The characteristics of the energy accumulators are resumed in table 3.2.

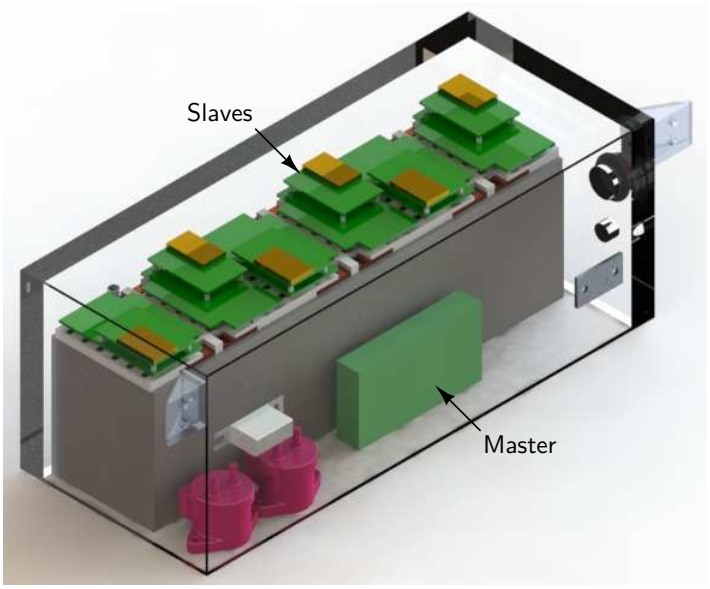


Figure 3.5: Preliminary render of right hand side battery of *FST-06e*. [Courtesy of Projecto FST]

Table 3.2: *FST-06e* battery specifications.

Parameter	***Value***
Maximum voltage	600 V
Nominal voltage	532.8 V
Minimum voltage	432 V
Maximum peak current	250 A
Main fuse current	160 A
Maximum charging current	10 A
Capacity	10 A h
Total energy	6 kW h
Total number of cells	144
Cell configuration	144s1p
Number of stacks	12 (2 containers)

3.2 BMS architecture

Of all the possible topologies presented in section 2.3, the only one that is completely inadequate for the proposed system is the centralized one. This is mainly due to the lack of scalability, meaning that a major redesign would be required to adapt the system to different cars, contrary to the idea of similar hardware and code bases across prototypes and could also result in slower or even stale progress from one prototype to another as was in part the case with *FST-05e* relative to *FST-04e*.

Also, with the manufacturing cost being so high given the prototype quality of the vehicle, a centralized solution is also a lot more expensive and risky. Specifically in maintenance, a single fault in one channel could require a full replacement of a 144 channel system for instance.

Finally, considering the mentioned stacks subdivision of the battery, it was desirable to minimize the number of connections attaching a stack to any other system. A centralized solution would then be the worst option possible since every stack would require a whole harness to connect to the same BMS, which couldn't

be removed easily by itself either.

From the remaining topologies, a fully modular solution has several disadvantages over a simpler distributed approach as covered in chapter 2, and more importantly *FST-04e* had a functioning distributed BMS [Guedes, 2011]. Therefore the first proposed solution two years ago for *FST-05e* was to reuse the same system, unfortunately it couldn't adapt to the new prototype very easily.

To name the most limiting restrictions, the PCB area required was unreasonably large, i.e. the required PCB area for the slaves of a single stack was significantly larger than the dimensions of the stack itself. For the chosen cells and configuration, the slaves would take more than double the reserved space in the first design iterations.

Moreover, the higher voltage of the new systems would require 144 slave modules instead of 48 — *FST-04e* had a 48s5p configuration —, which was already a significant number of modules. And finally, it was designed for lithium iron phosphate cells with different voltage ranges, and indeed it may be able to handle other chemistries, but was not designed nor tested under such circumstances and is documented to work only within a smaller than required range.

Reusing such system would then pose the following problems:

- *Cost:*

  FST-05e would be easily more than twice as expensive as *FST-04e*, not counting redesign, based on the huge increase of components.

- *Reliability:*

  Too many modules and CAN bus with 144 nodes.

- *Assemblage:*

  Too many modules of difficult installation as they cannot touch each other and would have to fit a really small space to keep the container competitive.

- *Suitability:*

  The slave module would require adaptation to be at least half the size which would be impossible without some fundamental changes in the design.

- *Validation:*

  Perhaps one of the most time consuming and important aspect of the design of such a critical system was not accelerated in any way by reusing this tested design. This would be a great advantage otherwise, and one that was expected when recycling a proven solution.

Therefore, the only gains were cheaper maintenance, easier software design and the existing code base that would fit this architecture, not enough to outweigh the mentioned disadvantages though. The team decided then to invest in another solution that could address these issues in a safe manner.

A way of solving the cost, assemblage and suitability problems is to have a single slave monitor more than one cell, like in a modular solution, but keeping the master-slave architecture. Also, from the BMS perspective it makes sense that such slave would then fulfill the requirements of individual stacks.

This maximizes the density of the slave modules and still allows a relatively fine grained control over the system's voltage if the team required so, given the small stacks. Each slave module is then required to monitor an array of 12 cells in series, amounting to 12 slave modules in the complete car, both in *FST-05e* and *FST-06e* — $12 \times 12 = 144$.

The proposed system was developed within the team in 2013 specifically for *FST-05e*, but ultimately failed in other aspects as covered in section 3.1.1. The topology was kept for the new design though, since the

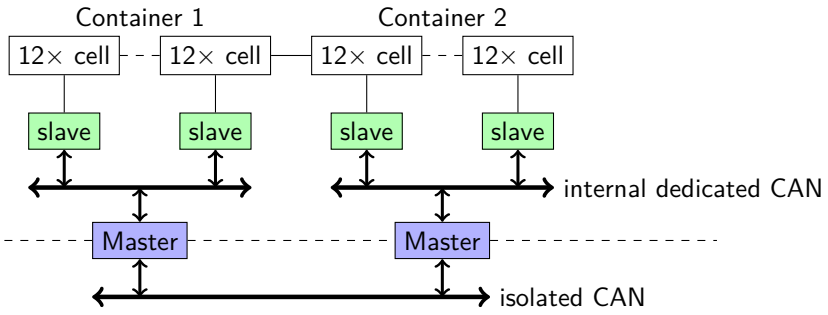


Figure 3.6: Proposed topology for new BMS in a two container configuration.

same considerations apply, with one addition: since *FST-06e* would have two batteries, the system would have to support at least two master modules, something mostly dependent on software however. This topology is illustrated in figure 3.6.

3.3 Communications

The chosen communications protocol for both internal and external networks was the CAN. This protocol is particularly suited for noisy systems subject to Electromagnetic Interference (EMI) with high requirements of reliability. Given that the vehicle has electric propulsion, thus generating a lot of electromagnetic noise, this is even more important.

CAN is also a standard in the vehicle industry, known for its robustness, so much that it was adopted by the aerospace industry as well (e.g. ARINC [2014]). But it's also desirable as the team has extensive experience with CAN and all of the modules within the car communicate through CAN with a strict message format. So, using the same format also promotes better integration.

Given the chosen architecture tough, an independent network must be established to connect master and slave modules. This bus should also be implemented as a CAN network following the same message format for consistency and for being easy to use and develop.

CAN was first developed at Robert Bosch GmbH, but later superseded and further developed in the ISO 11898 standard. The standard won't be detailed here except for the format of the message to better understand the message protocol the team has agreed on and maintained. This frame is illustrated in figure 3.7 without extended Identifiers (IDs) as they're not currently in use by the team since there is no need for them in the present.

The standard frame is, from an end user point of view, composed of three main parameters: the ID, the Data Length Code (DLC) and the data field composed of DLC×8 bits from 0 and up to 64 bits (8 bytes). The team then defined a simple implementation to be used within the car, where the only actual constraint is the presence and format of a timestamp for logging purposes mostly:

- *ID:*

  Unique identifier for network node or specific message of a node. In no circumstance there should be two nodes capable of using the same ID for a message — this is a requirement from the arbitration method of the CAN protocol.

- *DLC:*

  No specific rules imposed by the team, except for having a minimum value of 2.

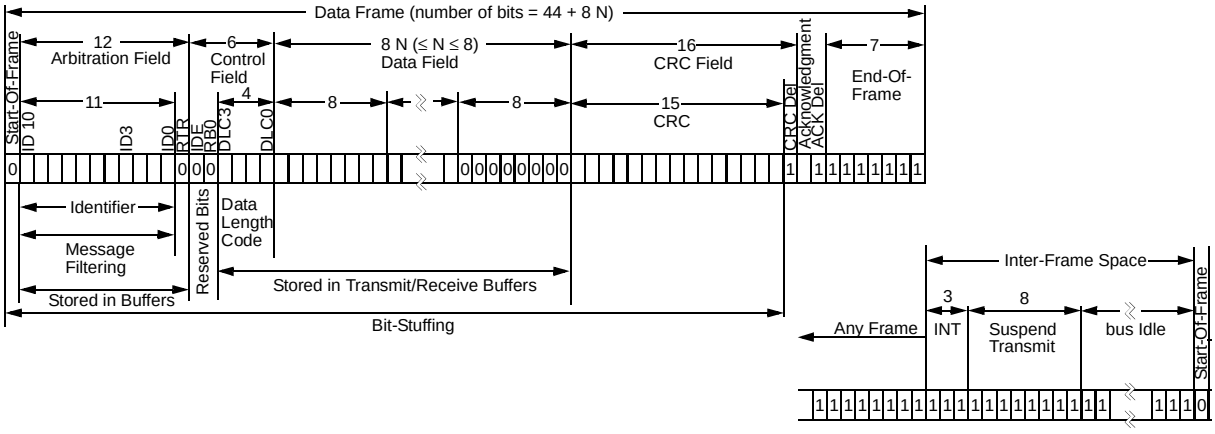


Figure 3.7: Standard CAN frame. [Microchip, 2006]

◇ *Data:*

First two bytes are always a timestamp of 16 bits in millisecond resolution, remaining bytes are to be used as needed.

Finally, the frequency for both internal and external CAN buses is fixed at $1\,\mathrm{Mbit\,s^{-1}}$. These unifying requirements are meant for compatibility with the remaining nodes of the network and better integration of debugging or logging tools.

As with any bus or feature, fail modes must be defined. Both because it's the only safe option and because it's a mandatory rule, any failure within the internal bus should be treated as a critical emergency. Therefore, the inability of the master module to assess a slave's status should prompt the isolation of the battery. However, a failure in the external CAN may be only considered a safety hazard in a 2+ container configuration.

3.4 Integration within the vehicle

In the system's integration there are a set of expected features from both sides, the BMS and battery as a whole, and the vehicle. The vehicle has two electrical systems distinguished within the rules: the Grounded Low Voltage System (GLVS) and the TS. This section details the integration of the proposed system within the GLVS.

For the interface with the master module, it was chosen to have all features controlled through CAN. This minimizes the number of wires and external components like buttons that are required to be connected directly to the BMS from outside of the battery container. This streamlines several other modules in the prototype and eases development and integration, since there are less wires to crimp, etc. This is in contrast with the previous approach which is still supported to minimize the number of changes required for the installation in *FST-05e*.

Any programmable module is then able to locally decide what the battery should do and forward the request to the BMS through the same CAN bus. This includes, but is not limited to, mandatory commands to turn on or off the TS, change between modes of operation — e.g. charging or balancing —, request debug information or battery status.

Regarding the power supply of the system, there are two options for BMSs: external or directly from the monitored cells. *FST-04e* used the second approach for the slave modules and the first for the master

module, which meant that the slaves were always turned on, draining the battery even with every other system turned off (including the master module).

This has disadvantages over the maintenance of the battery given that it's depleted faster in storage conditions, depending on the power requirements of the slave modules. More so, without the master, the slaves can only protect the cells from overvoltages (by enabling the balancing load). Therefore, without a master, a slave should enter a deep sleep state to economize energy.

However, the alternative allows a true zero power standby, reducing the maintenance costs of the system without any safety penalty. For *FST-05e*, the slave modules should then be supplied through the internal CAN bus at 5 V. This bus is in turn supplied through the master module. This way, the whole BMS system is dependent on being supplied by the GLVS and subject to the manual control of the GLVS master switch. However, there are two disadvantages in this approach.

The first is the dependence on the CAN supply and the second is the inability to log events while the GLVS is not powered. In *FST-05e*, the slaves would still be supplied by wires, invalidating the first disadvantage if not for the CAN bus being daisy chained. The system is still equally safe and it has an equal amount of cables, but a power interruption may shutdown several slaves. Since a FS car has to automatically shutdown with a single slave going offline, the availability of power is not compromised in the chosen organization.

The logging of critical events could also be important in a production car, in which case the BMS slave should be adapted with a different DC/DC to draw its power from the cells entirely and keep its isolation barrier in the CAN transceiver. However, in this particular application this has little to no safety usage. More so, not implementing and testing a sleep mode was also important to save development time and focus on online monitoring modes.

Finally, this option is software independent, apart from the eventual implementation of a reduced power mode. Depending on the particular target prototype this should be changed to fit the requirements.

Regarding the master module supply, this module is powered through the GLVS master switch, which needs to be able to interrupt the supply of all low voltage systems. In *FST-05e* the GLVS is a 12 V system, but in *FST-06e* it will be 24 V, a value set by the team for easier integration of other systems.

This makes it impossible to have compatible master boards between both prototypes without extra voltage level conversions that would otherwise increase the cost and possibly the dimension of the master module.

To address this without compromising the design of the PCB, it should be possible to fit two versions of certain components in the same footprint to target one system or the other.

3.5 Cell monitoring

One of the main tasks of any BMS is to monitor the cells to guarantee they're working within the documented and tested limits. As discussed before, this means monitoring voltage, temperature and current for each cell or group of cells. The limits and general requirements for this project are summarized in table 3.3.

For LiPO cells, voltage is so critical that it's mandatory by the FSAE rules that each and every voltage interval is monitored, but temperature not so much. For this reason the number of supported temperature sensors may be lower than the amount of cells installed[2].

Finally, the current should be monitored, but doesn't constitute a critical measurement. The proper

[2] Since the conclusion of the slave module for *FST-05e*, a new rule set was published imposing new limits, but still possible to comply given the choice to install more sensors than required in the first place, as seen in section 4.3.

Table 3.3: Measurement limits.

	Parameter	Value
Voltage	maximum	4.5 V
	minimum	2.5 V
	error	10 mV
	channels per cell	1
	frequency	1 Hz
Temperature	maximum	120 °C
	minimum	−30 °C
	error	5 °C
	channels per cell	0.5
	frequency	0.2 Hz
Current	maximum	200 A
	minimum	−50 A
	error	5 A
	frequency	20 Hz

design of the battery fuse, powertrain and charger ensures a safe operation regardless of the monitoring of this parameter. It is an important feature of an advanced BMS system nonetheless.

3.5.1 Voltage

The voltage limits were chosen based on all investigated lithium-ion cells having voltage intervals between 2.0 V and 4.3 V. Given that the BMS should be able to correctly measure cells that are for some reason out of their bounds, a good precision was required in a greater interval.

As for the measurement error, for most purposes, 100 mV to 200 mV resolution is enough to safely monitor a lithium-ion cell [Andrea, 2010]. Obviously, the error affects how far a cell may be safely charged / discharged, but given the sharp charge-voltage curve near the top and bottom voltages this has little impact on the available capacity — see figure 2.3.

However, this has a big impact on SoC estimation and balancing performance, as will be discussed in section 4.4. For these applications, Andrea [2010] suggests a resolution of 50 mV or better, but for logging purposes mostly, a smaller resolution was desirable. For this reason, and after analyzing available ICs, it was taken as a requirement a maximum error of 10 mV. This is also a common upper bound of voltage measurement error in commercial systems.

For the acquisition frequency, a value of 1 Hz for the voltage acquisition is a usual value found within commercial systems [Andrea, 2010]. However, the sharp ends of the capacity-voltage curve result in very abrupt voltage changes whit a highly irregular current profile, as is the case with an EV and more so with a racing EV. As a cell reaches its voltage boundaries, the frequency should then be increased, not for safety, but logging purposes.

Another requirement for the voltage measuring circuit is that measurements are made differentially, i.e. atomic measurements for each cell. The reason for this is a direct consequence of the analysis of *FST-05e* previous shortcomings as discussed in section 3.1.1.1.

Given the distributed architecture, the voltage measurements are the responsibility of the slave modules.

3.5.2 Temperature

Temperature requirements were again set with the general characteristics of lithium-ion cells in mind, with $-20\,^{\circ}\mathrm{C}$ to $65\,^{\circ}\mathrm{C}$ covering all usual limits for commercially available cells. Still, an extended range was needed for detecting abnormally high temperatures.

The error requirement was set to a modest $5\,^{\circ}\mathrm{C}$ since it's only meant to monitor if cells are within specific temperature windows. Previously, in Guedes [2011], this value was found to be a good setting for lithium-ion batteries.

For the temperature, the acquisition frequency may be more relaxed since it doesn't build up so quickly. Given the thermal capacity of the cells and low propagation under normal operation, a period of 5 s is appropriate. The only reason for higher frequencies would then be recording thermal runaways — e.g. testing the limits of a cell.

These measurements are also responsibility of the slave modules, but the master should control the room temperature of the container independently.

3.5.3 Current

The current measurement requirements were specified for ease of implementation. These are not safety critical, and precise measurements are difficult to obtain in a single iteration. The reason is that, for good measurements, a shunt resistor needs to be used, but that solution is inherently not isolated, demanding extra hardware since the shunt is installed in the high current path. Also, power dissipation becomes an issue, and such resistor needs to have a very small resistance, but that makes it harder to detect small currents and signal conditioning and amplification becomes a challenge. The cost of such high current, high precision resistor is also considerable, and more so the full measuring system.

An easier solution is a Hall effect sensor. This is isolated without requiring extra components and already outputs a strong signal that most microprocessors are able to identify and translate to a current value. The disadvantages though, are an offset current that may be measured — i.e. measured current doesn't tend to 0 if actual current does. Also, sensors for high currents are usually not very accurate for small ones.

An ideal precision was 10 mA or better. This would allow for correct monitoring of the charging process even in very slow charging stages. From an initial market research tough, it was unrealistic to expect a significantly better precision than 1 A or 2 A with a Hall effect sensor. However, more accurate (and presumably expensive) sensors may be available.

Moreover, the major concern with precise current measurements is the possibility of complex SoC estimation algorithms. However, with mix estimation models — i.e. taking both voltage and current as inputs for the estimation —, it's possible to eliminate the current induced error from the SoC with precise voltage measurements, but not the other way around [Codecà et al., 2009].

The frequency of acquisition has to be higher relative to voltage or temperature since current may vary a lot faster than voltage or temperature. 20 Hz was found to be a good value for a previous system [Guedes, 2011], and is therefore kept as a target reference.

The chosen solution was a Hall effect sensor that should be read from the master module. However, the current drawn from the balancing circuit has to be monitored independently through the slave modules.

3.6 Safety systems

The foremost function of any BMS is safety, which means it needs to be able to actuate the battery poles and manage the current being supplied. To avoid overcurrent under normal conditions, the container is internally fused to an appropriate value, independent of the BMS Still there are circumstances where the BMS must intervene.

The following sections describe the safety features controlled by the BMS that allows it to protect the battery and users from misuse. Some other safety systems need to be installed inside the container but in the chosen architecture are not integrated in the BMS system nor interact with it and therefore won't be detailed within this thesis.

3.6.1 AIRs

The Accumulator Insulator Relays (AIRs) are two normally open relays that control the availability of high voltage and high current from each battery container. Each relay controls one pole of the battery as illustrated in figure 3.8.

The master module is one of several safety systems that is connected in series to control these relays and should only allow their activation under safe conditions. It's through these relays that the TS becomes live.

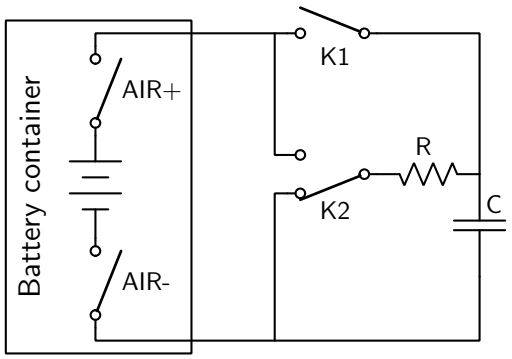


Figure 3.8: Pre-charge and discharge circuit. All switches represent relays controlled by the master module.

3.6.2 Pre-charge and discharge circuits

The pre-charge and discharge circuits are meant to charge and discharge the capacitors in the TS in a controlled manner. The pre-charge is a protection to the TS current path that would otherwise be endangered by the excessive current drawn by those capacitors at startup, i.e. turning the TS on. And the discharge circuit is meant as a protection to the driver or anyone touching the vehicle after shutdown, discharging the same capacitors and thus eliminating the presence of high voltage outside of the battery container(s) if the AIRs are opened.

For the discharge circuit, the rules mandate a normally closed circuit and a maximum of 5 s to reduce the TS voltage to the high voltage threshold defined in the rules as 40 V and 60 V for 2014 and 2015 seasons respectively. These specifications have to be fulfilled by components external to the BMS, but it's the BMS that controls the activation of said circuits through two remote relays and a resistor or bank of resistors with the appropriate value and power rating.

The implementation of the circuit may vary as long as the interface with the BMS is the same, i.e. two

relays. Figure 3.8 shows the chosen implementation for *FST-06e*, being the cheapest configuration. *FST-05e* has a slightly different setup, requiring only different logic (software) to actuate the same relays.

The failure of the pre-charge or discharge circuit may have harsh consequences. The discharge resistor is usually not able to withstand the maximum discharge current (i.e. $600\,\mathrm{V}/R$ in both *FST-05e* and *FST-06e*) for longer than 15 s, the minimum time limit imposed by the rules. This means it may burn if the master is unaware of a *K2* relay malfunction, e.g. broken wires would leave the relay in its normally closed position.

The failure of this circuit would then leave high voltage inside the motor controllers endangering anyone during maintenance tasks. Also the burning of the resistor could be violent enough to damage surrounding components.

A pre-charge failure on the other hand would result, best case scenario, in a broken main fuse from the overcurrent — assuming the discharge circuit was working fine and the bank of capacitors in the TS was indeed discharged. Still it's also likely that the AIRs become damaged and stuck closed, leaving the container poles always energized.

For these reasons, the BMS should be able to test these circuits in order to actuate the AIRs accordingly. More so, this avoids human errors on easy yet unnecessary manual tests every time the car is turned on — tests that would indeed do nothing about a malfunction that happens in the middle of the track.

3.7 Cell balancing

As seen before, balancing is not only desirable, it's a mandatory feature. It's the only way of assuring that the full installed energy can be used and the cells protected from early failures.

The BMS should then be able to balance all the cells in series within a margin of each other. As a target reference, the maximum SoC difference should be around 1 % and 3 %, which roughly translates to a 100 mV difference in cell voltage for a typical lithium-ion cell.

This setting could be more strict depending on the precision obtained in the voltage measurements. Still, this is already a good target considering commercial systems.

3.7.1 Balance method

In section 2.2.1 were presented two distinct approaches to balancing: active and passive. From the analysis, it's concluded that the decision should be made between increasing installed capacity and active balancing.

A formal analysis of which method would yield greater gains to the team is beyond the scope of this thesis and involves vehicle dynamics simulation. Still, it's easily assumed that in fast tracks, active balancing wouldn't improve the vehicle's range and would in fact increase weight and volume of the container, actually decreasing performance.

In contrast, passive balancing is not only easier to implement, but also possible to do so in less volume and with a reduced weight for a greater balancing power. Therefore, the proposed system should use passive balancing. The balancing should then occur only under charging or safety exceptions — e.g. if an overvoltage is detected.

3.7.2 Balance current

If cells unbalance at a certain rate, it's natural that a minimum balancing power must be available to keep them within balance. Both the balancing capability and the rate of unbalancing of a battery may be specified in several ways, the most common being a percentage over the nominal capacity.

For the purpose of a better illustration though, and for better connecting with following chapters, the value being used will be current intensity. However, this value is not adjusted for different capacities and should not be used directly to compare different cells / batteries.

As a first approximation, we'll consider cell unbalancing to occur only through their self-discharge rates. For each cell is then possible to define an equivalent current I_d that would represent the self discharge rate — not actual current since we're treating the problem of a cell in an open circuit. This may be thought of as if the ideal cell was constantly connected to a small load imposing such current. Naturally this equivalent current is different from cell to cell, and with the above considerations, this is the only unbalancing factor.

The maximum difference in this current, ΔI_{max}, is the unbalancing source. The minimum current to keep the battery balanced is therefore $I_b = \Delta I_{max}$, but only if the balancing is *always* active and if the cells are already balanced when they are installed. For other unbalancing reasons, it may also be defined a drainage difference as ΔI_{max}, which is likely very different across operation modes, i.e. different loads.

Depending on the ratio between the two and battery capacity, full balancing may be achieved from seconds to years. But a BMS or the balancing load may have an uptime lower than 24 h a day, which results in even longer balancing time. The installed system should then be able to dissipate an energy equivalent to a current $I \gg \Delta I_{max}$.

Lithium-ion cells have very small self-discharge rates, making the delta even smaller. In unfavorable harsh conditions, these cells may have self-discharge rates around 30 % of capacity per month [Abe et al., 1999], but are usually significantly lower.

In *FST-05e*, the cell pairs have a capacity of 9 A h and *FST-06e*'s cells have 10 A h. Worst scenario, this translates to self-discharge currents of 3.75 mA and 4.17 mA respectively, according to equation 3.10.

$$SDC = \frac{SDR \cdot C}{30\text{d} \cdot 24\text{h}} \tag{3.10}$$

In the proposed design, all cells have a very similar load under all circumstances and there was a careful choice of cells to minimize differences in internal resistance and capacity. For these reasons it was assumed that the unbalancing rate in either prototype would always be $\ll$10 mA, a very conservative value. Any exception would only happen for severely damaged or worn out cells.

Taking *FST-04e* as a reference, the maximum balancing current was 2 A for a 50 A h battery[3]. For an equivalent effect and potentially similar charging time, the ratio should be maintained. For *FST-05e* and *FST-06e*, the equivalent balancing currents would be significantly lower as shown in equations 3.11 and 3.12 respectively.

$$\frac{2}{50} = \frac{I_b}{9.5} \Leftrightarrow I_b \cong 0.38\text{A} \tag{3.11}$$

$$\frac{2}{50} = \frac{I_b}{10} \Leftrightarrow I_b \cong 0.4\text{A} \tag{3.12}$$

Of course, the higher the balancing current, the faster the balancing. In fact, for *FST-05e*, assuming a balancing current of I_b=250 mA results in a 38 h long process for a 100 % unbalanced battery, i.e. at least

[3] Note that this battery unbalanced a lot faster given the different chemistry and higher internal resistance.

one cell fully charged and one fully discharged:

$$\text{Time to balance} = \frac{C}{I_b} = \frac{9500}{250} = 38\text{h} \tag{3.13}$$

On the other hand, greater currents involve greater power being dissipated inside the battery. As stated before, *FST-05e*'s battery was already built and has no refrigeration of any kind. For this reason this prototype proves to be very limited in its balancing power.

Finally, while this is critical for initial balancing — e.g. after replacing a battery stack —, it's not as meaningful for keeping the battery balanced. If the difference in capacity that needs to be balanced is only 10 % of the total capacity — still a lot higher than expected if the battery is properly maintained —, the balancing period is reduced to 3.8 h. A nominal value of 250 mA is therefore appropriate for the battery since the charger being used requires between 3 h and 4 h for complete charge, depending on how pedantic the charging cycle is.

Still, balancing techniques that improve upon *FST-04e* should be suggested, allowing for a competitive charging process even with *FST-05e*'s limitations. For *FST-06e*, proper container refrigeration will be implemented greatly improving balancing performance.

3.8 Battery interface

One of the greatest advantages of a digital BMS is allowing for digital communications with it. This is highly desirable for the development process, for the everyday usage and maintenance of the battery, and also mandatory by the FSAE rules to a certain extent.

Both for helping the development of the BMS system and to complete it with the proper tools to manage it at a higher level, appropriate interfaces should be developed that expose and monitor any parameter within the embedded system.

Since the communications are made through CAN bus with the same format used in *FST-04e*, certain tools should already exist to give response to some of the interface problems. One such example would be a CAN translator to convert those messages to a protocol that a 'normal' computer would understand without installing CAN hardware and software translation layers.

Indeed, these tools exist, but are not capable of delivering the required performance. For the intended 144 cell system, assuming 6 parameters per cell, 3 parameters per CAN message and a 1 Hz throughput, 288 messages per second are required:

$$\text{Messages per second} = \frac{6 \cdot 144 \cdot 1}{3} = 288 \tag{3.14}$$

Before the start of this project, existing tools could only take roughly 50 messages per second among other issues. Although pessimistic, this estimate is almost 6 times greater than that. Furthermore, the CAN bus at 1 Mbit s^{-1} can usually host around 6200 messages per second (80 % full), depending on how uniformly the messages are distributed in time.

For this application, the solution should reliably identify upwards of 1000 messages per second. Additionally, messages should be buffered, if necessary, to handle bursts of messages. The existing tools were far from achieving any of these specifications.

As for the information made available by the interfaces, safety is a critical feature. For this reason, the interface should be stateless, i.e. display values read directly from the bus instead of doing so from an internal representation that could otherwise be compromised. Proper indications of lost communications should be

implemented through watchdog timers, for instance, assuring that the displayed information is up-to-date and exactly replicated from the CAN bus.

These tools should be able to expose, at least, all the individual voltage levels and temperature readings from the BMS until the British competition, to comply with the rules. Still, better functionality should be available and accounted for on the interfaces' design.

Finally, the base of the system should be versatile enough to handle any CAN message on top of the BMS ones. Also, the Qt framework should be used to help future integration on other tools like [Figueiras, 2014].

Chapter 4

Slave module

The slave module was the most critical in the new design. The objective was to use *FST-05e* as a testbed for the electronics that would be included in *FST-06e*. This also had the advantage of repairing the battery of the fifth prototype.

The versioning scheme for this module starts at *v3.0*. This is meant to distinguish from version *v1.x* from *FST-04e* [Guedes, 2011] and from *v2.x*, the original *FST-05e* module. It also gives a sense of continuity from one generation / technology to the next one.

In this chapter, there are extensive references to internal parameters of the module. Although the names are largely self-explanatory, the definitions are presented in appendix A. These are distinguishable as all capitals strings in typewriter font — e.g. `V_CRITICAL_H`, which should be read as critical high voltage.

The schematics for the slave module are presented in appendix B and the production masks for *FST-05e* version of the module are documented in a 1:1 scale in appendix C, both obtained with Altium Designer.

Finally, all software was versioned through the source code management tool *git* and all references to software — including following chapters and appendixes — are relative to `git:tag:BMS_THESIS_DELIVERY` in the branch `git:branch:master` of the respective source trees, unless explicitly mentioned.

Module programming, as for every other module with a *dsPIC*, was achieved through a PICkit™ 3 and the IPE interface [Microchip, 2010, 2013*a*]. The code is written mostly in C with a few assembly calls, compiled through the XC16 C compiler and MPASM™ assembler (free versions) [Microchip, 2013*b*,*c*]. Most embedded software is C90 compliant, but some Microchip extensions and assembly macros are used.

4.1 Architecture

This module uses a *dsPIC30F6012A-30I/PT* [Microchip, 2006] to acquire temperatures and drive a *LTC6803-4* [Linear Technology, 2011] to monitor cell voltages. The first is a Microchip 16 bit MPU referred to as *dsPIC* from now on, and the second is a BMS IC from Linear Technologies referred to as *LTC*.

The MPU is able to communicate with a master module through a CAN bus. This bus includes a 5 V power line and supplies the whole module except for the *LTC*. To avoid the installation of an extra DC/DC, the *LTC* is powered directly by the cells being monitored.

As each module works at different voltage levels, an isolation barrier is needed. This is provided by an isolated DC/DC and CAN transceiver while the remaining electronics share the lowest potential of the monitored stack of cells. Therefore, between any two slave modules there is a maximum of $12 \times V_{max}$ [V], the maximum voltage of each cell.

An overview of the module architecture is shown in figure 4.1. While this architecture is specific to *FST-05e*, the only difference to *FST-06e* should be the inclusion of the temperature sensors in the module's PCB — see subsection 4.1.1.

For the MPU, there are some drop-in replacements, and others may require only small code changes. Some of these options are cheaper, e.g. a PIC24 could be used with no loss of performance or capability. However, most of the MPUs used by Projecto FST are of this particular model, making it an attractive solution given the vast experience with the chip and its versatility. Also, processing power and memory shouldn't be an issue with most 16 bit controllers of these families, providing a good base to expand on the software complexity in future revisions.

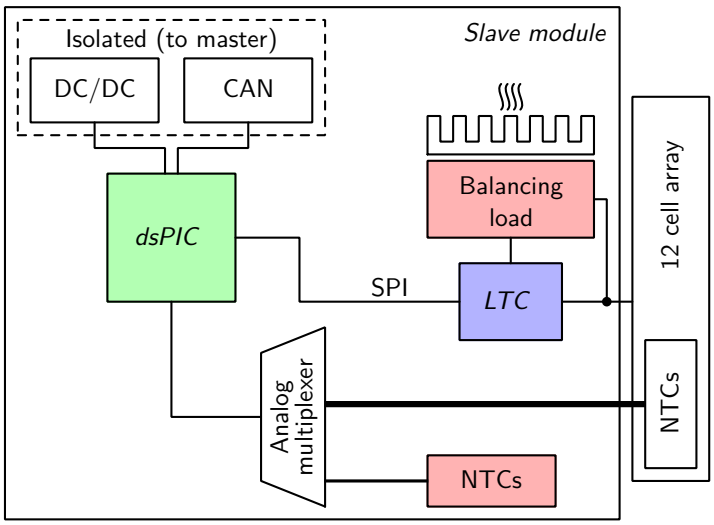


Figure 4.1: Slave module architecture (*FST-05e*).

4.1.1 Layout

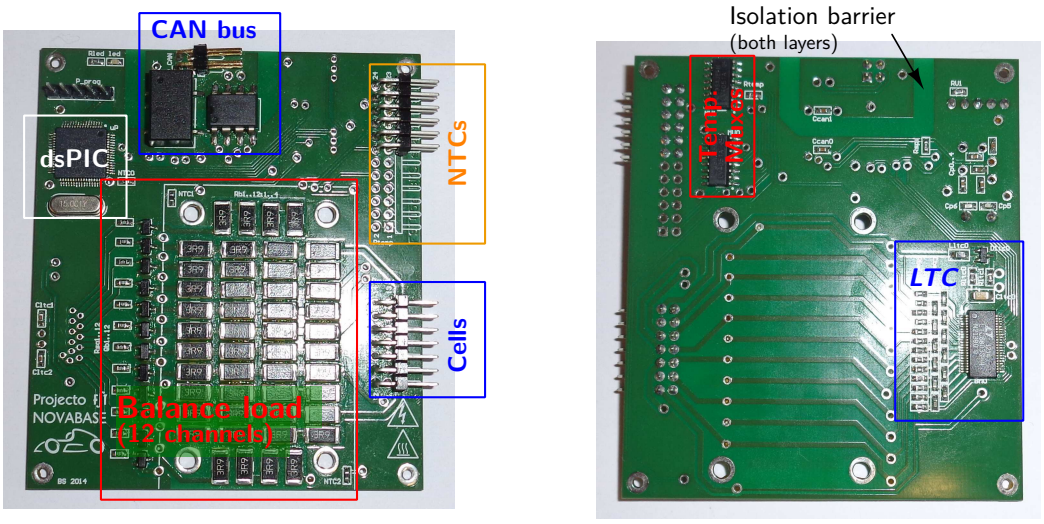
The designed slave module targeted *FST-05e*. For this reason, the current prototype is very similar to the original module (v2.x) in shape, selected connectors and respective position — compare figures 3.4 and 4.2. Indeed, many components are similar or equal in an attempt to recycle stocked components and to reduce the time spent on component identification.

For *FST-06e*, the team needed to solve some of the problems found in *FST-05e*. Namely, and foremost, the team wants a system without wires between slave and cell stacks, also for easier assembly. There are currently two solutions being developed and prototyped together with Projecto FST, both requiring a new layout for the slave module, as expected.

Figure 4.3 shows a CAD render of the first and most promising solution. This significantly improves on the quality of the connections by crimping the pads of the cells in series together rather than through a busbar. Busbars are still used in the illustrated solution, but are meant for slave connections and no high current flows through them. More so, the assemblage is easier, more compact and reliable compared to *FST-05e*.

4.1.2 Routines

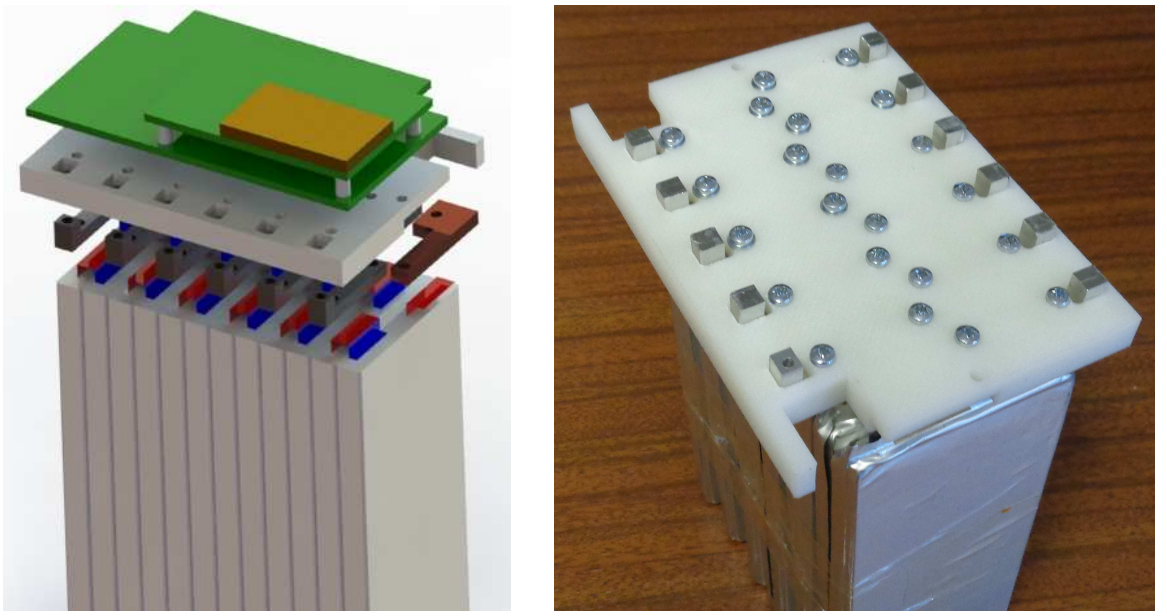
For the software implementation, a Real Time Operating System (RTOS) was considered, particularly FreeRTOS [Real Time Engineers Ltd., 2014] for being supported in Microchip devices and being open source.



(a) Top layer.

(b) Bottom layer.

Figure 4.2: Slave module v3.1 (*FST-05e*).



(a) Partial render of proposed stack solution in exploded view.

(b) Early proof of concept. The slave module should connect to the exposed part of the bus bars.

Figure 4.3: *FST-06e* stack concept. [Courtesy of Projecto FST]

However, a quick evaluation of the complexity required for the code of the slave module showed that only a few and simple interruptions would be needed.

Interruptions allow for tight time constraints, and thus real time execution, as well as a light approach to event handling. More so, nested interruptions allow certain tasks to be put on hold to process a higher priority interruption. Since there's a single monitoring loop and most tasks require a sequential execution, an interruption driven code is able to cope with these tasks. The extra complexity of a RTOS was then largely unnecessary in the slave module and unfamiliarity with RTOS deployment provided a deciding reason not to use it.

Figure 4.4 represents the main routines of the slave module. Here the normal monitoring routine is represented by blue elements and arrows, whereas red is used to denote error handling. Additionally, some code paths still have limited functionality in the current stage — gray elements. These are not fully implemented for lack of support from the master module at the time of deployment / implementation or for design reasons — e.g. the SoC estimation is very crude as seen in section 4.5.

4.2 Voltage monitoring

For the voltage monitoring, any solution that has a MPU per cell in series may easily use the ADC from the MPU itself. However, for larger number of cells, the high voltages require extra components to convert them to values within the MPU range. Indeed, for the chosen *dsPIC*, this is only 5 V relative to the bottom of the stack. This is not enough to accommodate the +50 V of 12 cells in series and, actually, it would only cover the first cell.

Compared to integrated solutions, implementing the measuring circuit in discrete components would require a lot of testing and characterization work. Possibly, it would also incur on the same problems of the original *FST-05e* solution. More so, it would require a larger area in the PCB — in order to have differential measurements — and therefore make it impossible to fit in *FST-05e* and probably hinder the design of other prototypes.

Of all possible integrated solutions, the most capable ones are those dedicated to a low count of cells — between 1 and 3 cells in series. These are used in cellphones and laptops for instance, and integrate advanced algorithms to determine SoC, cell chemistries abstractions, developer tools such as reference interfaces and several other features depending on the manufacturer. Unfortunately, these options become more expensive and hard to integrate in systems with large cell counts.

A good reference for comparing available BMS ICs is Andrea [2014], which contains an up-to-date comparison of available ICs befitting a high voltage battery. The best IC we found to fulfill our specifications was the *LTC6804*, but, being a new model, it was not available in time for manufacturing.

The chosen IC was then the previous generation of the *LTC6804*, the *LTC6803*, which has a very similar pin-out and feature set. In future revisions of the slave module, the newer IC could be used with small adaptations for a greatly improved performance. Regardless of the model, however, this IC offloads the task of measuring voltages of up to 12 cells in series per chip from the *dsPIC*. This allows for a single IC per module interfacing with the *dsPIC* through a 1 MHz SPI bus.

Given this architecture, voltage monitoring is dependent on the initialization and communication with the *LTC*. Failure to do so results in an emergency being broadcasted, followed by a reset of the module, which stays in this loop until proper initialization is achieved.

Under normal operation, the *LTC* is configured to read all 12 voltages every 13 ms. The *dsPIC* queries the *LTC* every 500 ms through an interruption and compares all values against a set of rules comprised of limits and operation modes. If an emergency is detected, a CAN message is generated and the module enters

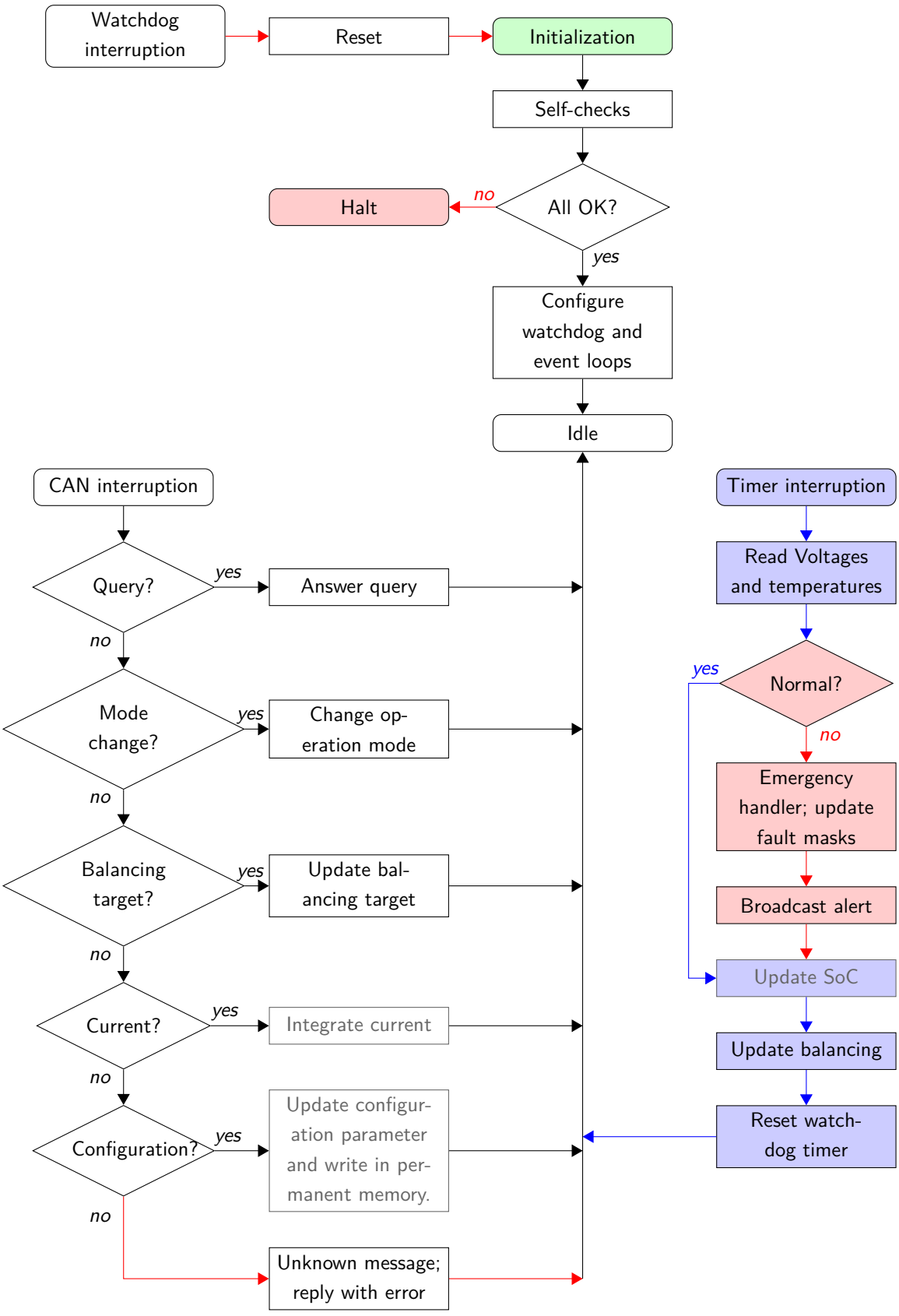


Figure 4.4: Simplified slave module flowchart.

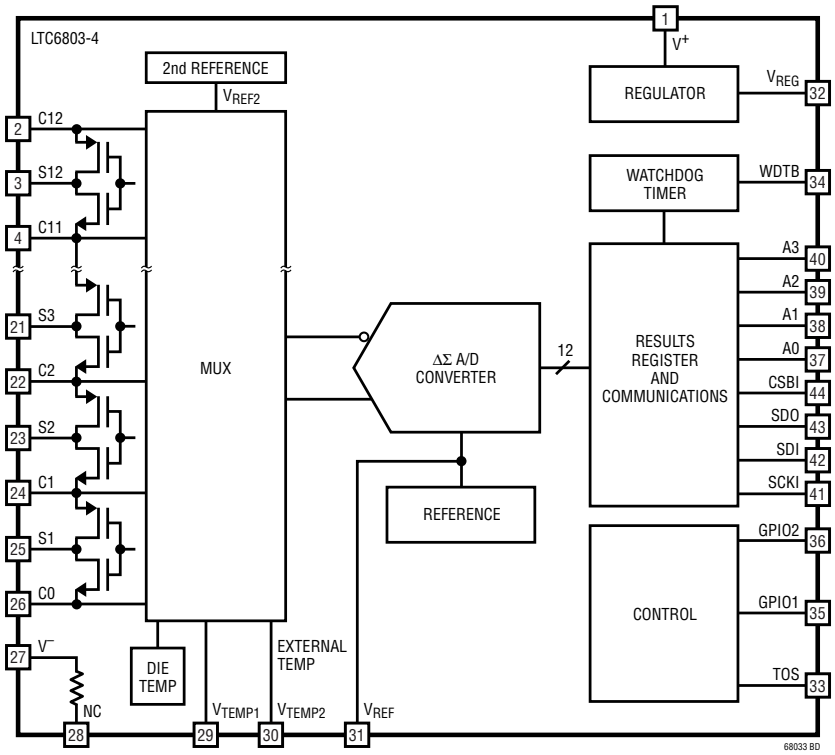


Figure 4.5: *LTC* internal architecture. [Linear Technology, 2011]

the according emergency mode.

Further actions to minimize the risks to damage the cells are taken, if possible. Forcing the activation or deactivation of the balancing load in case of over and undervoltage, respectively, are the only actions the slave module is able to perform. For complete protection, the slave module relies on the master module to react accordingly — see section 5.2.

4.2.1 Measurements

The *LTC* uses a 12 bit ADC to measure accurately voltages between -0.3 V to 5 V. The wide range is intended to allow monitoring of any cell and also supercapacitors, which may go slightly below 0 V during operation.

However, in the chosen architecture, the *LTC* is supplied by the cells. If these were in fact supercapacitors, the *LTC* could have insufficient power to operate. As this system targets batteries, this is not a limitation, still the adaptation requires only the provision of the *LTC* with a different supply source.

The full resolution of the ADC is slightly larger and ranges from -0.768 V to 5.376 V. This translates to the resolution of 1.5 mV (equation 4.1).

$$\textit{LTC} \text{ voltage resolution} = \frac{5.376 + 0.768}{2^{12}} = 1.5 \times 10^{-3}\text{V} \tag{4.1}$$

The values are reported to the SPI master (the *dsPIC*) as the ADC value: between 0 and 2^{12}-1, directly proportional to the cell voltage. However, the CAN messages use multiples of 8 bit data fields and there's enough performance and memory on the chosen MPU. For these reasons, it was chosen to convert the value directly to a human readable format.

In embedded systems with no direct interface with an operator / user, data processing is usually minimal, since the system itself has no need for human readable formats. However, it was an option that helped initial debugging and development of the system. Without stable interfaces and abstraction layers, this was crucial for fast debugging from the raw output of the CAN bus. This way all measurements are stored and reported in $mV \cdot 10^{-1}$. The use of these units extend to every module and interface designed afterwards to avoid further conversions.

Finally, a different representation is needed for stack voltage, i.e. the sum of all 12 cells. This is stored and communicated in mV.

4.2.2 Self-checks

All self-checks are done at startup and in every measurement cycle. These consist of assuring the *LTC* is responding correctly to commands and checking if all cells are online.

In the first case, the sheer lack of response is indicative of malfunction, but every message in the SPI bus is also sum-checked on both ends. Any failure results in complete reset of the module and reinitialization, which also forces the reconfiguration of the *LTC*.

However, the bad connections are more difficult to detect, but specially necessary for *FST-05e* — in this prototype, the slave modules connect to the cells through wires introducing an extra point of failure. The standard procedure of the module is to accept any value read provided it's within normal intervals for a cell. This may result in a few bad readings before the fault is detected / diagnosed. However, the alternative is very costly since a full detection is dependent on several successive readings. For this reason, the normal readings provide the data for detection, but results in some undervoltages and overvoltages being detected before identifying the wire (or PCB trace) fault.

To facilitate and reduce the time required for this diagnosis, further measurements are made with a special purpose mode built into the *LTC*. This mode uses internal current sources to discharge the capacitance of the infeed filters. This should reduce the number of cycles until the detection and provide better consistency.

The detection delay was considered to be acceptable as long as it was within 3 measurement cycles. However, even this solution introduces 'measurable' jitter in the main loop since it duplicates the amount of data on the SPI bus and sum-checking every message is computationally expensive.

Certain optimization flags were able to more than double the performance of this task, but for stability reasons, the optimization level -O0 was be kept until further testing is possible. Increasing the frequency of the *dsPIC* is also an option, but it would increase power consumption and possibly require a heat sink for the MPU. However, current configuration allows enough idle time to answer any query from the master, even in worst case scenarios.

Finally, a particular case in wire faults occurs when all but 2 or 3 cells have been disconnected from the module. In this case the *LTC* may become underpowered depending on which cells are online and the *dsPIC* resets itself reporting a module fault rather than connection faults.

This could be avoided if the *LTC* supply was independent from the cell array. However, this is a rare and largely inconsequential wrong detection — only 2 or 3 out of 12 cells could be monitored anyway. To prevent against this situation, an extra DC/DC would be required to power the *LTC* from the CAN bus. If this DC/DC would be introduced in a new layout, the detection would work normally with no software changes.

Information about the state of all the connections is stored and communicated in a 16 bit (12 bit useful) binary mask.

4.3 Temperature monitoring

Among the features present in the *LTC*, one may not be used without extra electronics: temperature acquisition. Virtually every BMS IC solution on the market has support for reading temperatures, usually through NTCs. However, manufacturers only implement 1 to 3 channels for temperature acquisition per chip — e.g. the *LTC* has 2 despite targeting a total of 12 cells in series.

To comply with FSAE rules, a minimum of ¼ of the cells should be directly monitored for temperature variations. Therefore, additional multiplexing is required. In order to improve performance by not relying on the SPI bus and take advantage of the installed hardware, the temperature acquisition function is implemented with the 12 bit ADC from the MPU.

This strategy was also used in the original BMS apart from a small alteration. The *dsPIC* has enough ADC channels to connect to all of the NTCs at the same time. This has one major advantage: it allows asynchronous acquisition of values in a regular period, reducing the time spent on temperature readings — reduced to the time of accessing an internal array of registers.

However this is a very slight time difference, PCB routing is harder and, above all, it forces all the NTCs to draw power constantly. The reason is the impossibility of synchronizing the ADC module with the multiplexing of the NTCs.

At an average resistance of 5.7kΩ at 25 °C and a supply of 5 V, this would require an excess of 50 mW per module if all 12 NTCs are installed (equation 4.2). Actually, for higher yet still normal temperatures — i.e. lower NTC resistance —, the power consumption could raise above 100 mW.

$$P = \frac{V^2}{R} \tag{4.2}$$

The target power requirement for the module after component identification was a maximum of 100 mW, making this extra power requirement unacceptable. It would also be more expensive since a larger isolated DC/DCs were required to supply all the slave modules on each end of the CAN bus.

For these reasons, the NTCs are multiplexed and only one is powered at a time. The acquisition time is longer, but still negligible — around 10 µs more per measurement. Actually, reading the temperature of the *LTC* die through the SPI takes longer than the whole acquisition after proper configuration of the ADC module. Once more, this is due to sum-checking all the messages in the SPI bus.

A simplified schematic of how the NTCs are connected to the MPU is illustrated in figure 4.6.

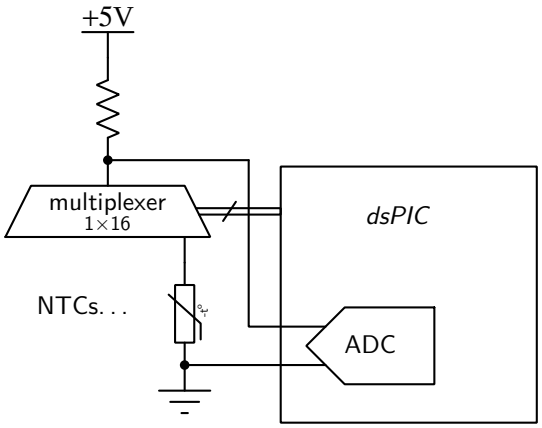


Figure 4.6: NTCs multiplexing schematic.

4.3.1 Measurements

Any voltage measurement is subject to the references used by the ADC. In this case however, we're interested in a resistance value, not the measured voltage. Considering the ADC has a 12 bit resolution and referring to figure 4.7, equations 4.3 and 4.4 show that the resistance value measured is independent of the reference used.

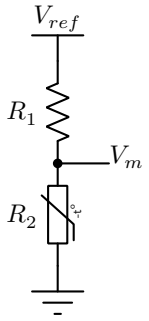


Figure 4.7: NTC resistance (R_2) measuring schematic through a voltage divider.

$$V_m = \text{ADC value} \cdot \frac{V_{ref}}{2^{12}} = \text{ADC value} \cdot \frac{V_{ref}}{2^{12}} \tag{4.3}$$

$$\frac{V_{ref} - V_m}{R_1} = \frac{V_m}{R_2} \Leftrightarrow R_2 = \frac{\text{ADC value} \cdot R_1}{2^{12} - \text{ADC value}} \tag{4.4}$$

For this reason no voltage reference is needed for this slave module[1]. For the ADC reference, the 5 V supply from the CAN bus is used, which is also the supply for the NTCs. Even if this value varies slightly, the final resistance calculated should be the same.

Figure 4.8 illustrates how the ADC value and NTC resistance map temperature values for the chosen NTCs. For low temperatures, the resistance of the NTC varies very slightly with temperature, reducing the accuracy of the measurement. However, sub-zero temperatures are rare if not impossible for the projected life of the prototype.

Similarly to voltage measurements, and for the same reasons, the ADC values for temperatures are also converted to human readable units. All temperatures are therefore converted to °C·10 and stored as 16 bit integers.

The math operations to obtain the temperature are relatively complex given the logarithmic relationship. However, after a general optimization and prototyping, it was verified that these calculations didn't compromise the *dsPIC* performance in a meaningful way and methods like conversion tables were not used.

4.3.2 Self-checks

All cell temperatures are measured through NTCs, as well as most temperatures within the slave module. The one exception is the *LTC*, which has an internal sensor.

For the temperature of the *LTC*, a digital interface is provided. Therefore any thermal malfunction is detected unless the *LTC* is unresponsive, resulting in a device reset. In case of an abnormally high temperature, the IC shuts itself down, but the occurrence is stored as a flag allowing the MPU to identify the fault after the reset.

[1] The *LTC* still has an internal reference for its voltage measurements.

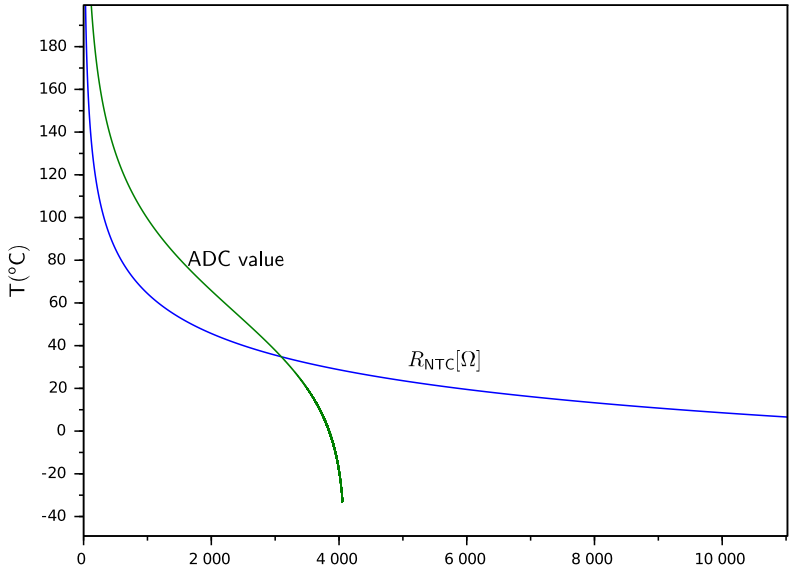


Figure 4.8: Measured temperature relative to NTC resistance and ADC value.

For every other sensor, the measurement is made through the same ADC and multiplexers as illustrated in figure 4.6. In case a NTC becomes damaged or disconnected from the *dsPIC*, it will always result in reading a voltage of V_{ref} (2^{12}-1 in ADC value) apart from noise variations.

The pull-up resistor and the NTC should have a resistance in the same order of magnitude to provide a good resolution. As long as this is true, such measured value will always be impossible to obtain in normal operation. In fact it corresponds to a temperature of $\leq -40\,^{\circ}$C for the chosen components, well below the temperature range being measured. Checking the correct operation of the temperature sensors is therefore 100 % reliable and doesn't require specific routines: any temperature below $\cong 40\,^{\circ}$C should be treated as a wire fault.

4.4 Balancing

The balancing circuit is controlled by the *dsPIC* through the *LTC*, which drives 12 p-mosfets independently to connect a bank of resistors to each cell. However, if the control was completely driven by the MPU, a Pulse-Width Modulated (PWM) wave could modulate the balancing current through an optocoupler, for instance.

While this would allow for greater flexibility in the achieved balancing current, it has a great impact on the PCB area required. After an initial layout mockup, this solution was clearly impossible for *FST-05e*. On the other hand, the alternative was simpler to manage and faster to implement both in hardware and software and could take advantage of another feature of *LTC* to increase voltage measurement accuracy.

To simplify the wire harness and / or PCB routing, the same two wires and PCB traces are used for both purposes: measuring and balancing. Since balancing draws current from the cell, this could result in a loss of accuracy in a similar way to the described in section 3.1.1.1, but not on the account of I_L: in this situation it's the current $I_m \neq 0$ the responsible for the artificial errors (see figure 3.3).

In this figure, for this situation, I_m includes the balancing current. This situation should be avoided if possible, specially since these resistors are so close to the *LTC* and the respective RC filters of each individual channel. By using the integrated functions of the *LTC*, balancing is interrupted automatically for the cell

being measured plus the contiguous cells.

This would be impossible to do from the *dsPIC* without making the system a lot slower since there is no direct control on which cell is being measured at any time. Effectively, to achieve the same result, the *dsPIC* would have to interrupt balancing for all cells until the acquisition of all 12 voltages was complete. Furthermore, a broken wire could be mistaken for an over/undervoltage if an adjacent balancing line was active. Guaranteeing the balancing is disabled during the measurements assures such mistake is impossible.

With previous considerations, measurement accuracy and better error detection was prioritized over modulation of the balancing current. However, depending on the installed cooling power, balancing may still be more or less aggressive. Since the adjustment cannot be done by software, this current is determined by the choice of resistors and the voltage of the cell.

The precise measurement of this current per balancing line would be a great advantage, yet it poses several technical challenges similar to measuring the voltages themselves. It was determined that for this architecture a better compromise was calculating the current from the ohms law (equation 3.1).

The error of this approach should be very small relative to the projected minimum balancing rate of 250 mA, despite resistors being significantly affected by temperature variations — usually around 200 ppm/°C. Worst case scenario, it would yield errors 2 orders of magnitude below the actual current, already taking into account voltage error as well and assuming resistors of 1 % of error.

4.4.1 Strategy

There were two usual options to control the activation of the balancing load. The most straightforward and common is activation upon full charge of a cell. This means there is no balancing throughout most of the charging process, until at least one cell reaches the upper voltage limit. At this point, the BMS would completely interrupt or severely limit the charging current and activate the balancing load for those cells.

This method is effective, but wasteful. If balancing starts earlier, the generated heat is better distributed over the charging time, reducing the need for cooling. This is especially important for latter stages when it's more likely for the car to race — the battery should not be warm at that point, if possible. In a container without active cooling like the one in *FST-05e*, this was expected to prolong charging for some hours had it not be implemented.

Moreover, the second method allows for offline balancing — i.e. with no charger —, a desirable feature. With it, the battery may balance overnight and allow for a faster and more complete charge in the morning after. In the competitions, charging is only allowed during the day, but so are all the dynamic events, making the optimization of the charging process very important. It's also a lot safer to leave the battery balancing with no charger attached, since it may be done with no high voltage exposed — i.e. the AIRs opened.

For these reasons, balancing is a process independent from the actual charging and is controlled by voltage alone. Given the typical voltage-charge characteristics of lithium-ion batteries, this process may only be useful above a certain voltage. Even if all voltages are matched in an intermediate value between V_{min} and V_{max}, after charging they may not be matched anymore given differences in capacity and internal resistance, not being a proper solution for a first iteration. A better alternative would be the use of the SoC value, but that would require good confidence in the SoC estimation algorithm[2].

Therefore, the chosen implementation was the use of voltage values directly. The optimal point for the start of balancing is not something that can be determined however, since it would highly depend on the cell chemistry and likely many other factors such as the health of the cells. The software was thus designed to

[2] In the proposed solution, this does not apply since SoC is currently interchangeable with voltage apart from a simple transformation (see section 4.5). On the other hand, there's no benefit in doing so either.

balance at any point and an hysteresis window to avoid balancing based on measurements error.

To avoid shifting the responsibility of starting and interrupting the balancing process to the user, a good improvement is achieved through an hysteresis window that changes according to the average voltage level of the cells. This allows for aggressive balancing on the steep ends of the voltage-charge profile and a greater interval across the middle section where the SoC may vary a lot for the same voltage (see figure 2.3). These hysteresis windows are defined by `BALANCE_HYSTERESIS_P` and `BALANCE_HYSTERESIS_C`, for *precision* and *coarse* balancing respectively.

Since the system is not intended (only) for single slave batteries, the balancing target must be coordinated between every slave. This is relayed on the master module that should broadcast the lowest voltage in the whole battery.

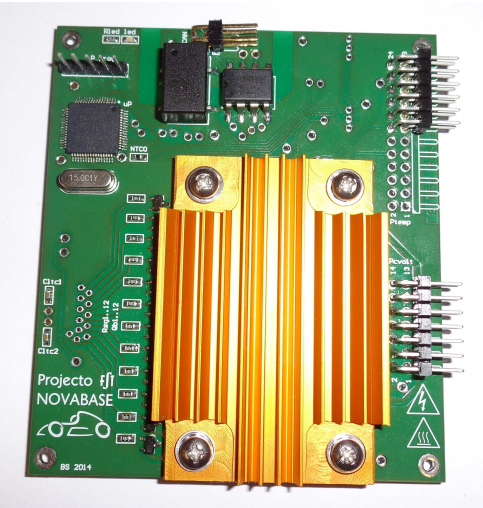


Figure 4.9: Slave module v3.1 (*FST-05e*) fitted with a low profile heat sink.

To manage the temperature during balancing, the slave module monitors the balancing load through two Surface Mounted Device (SMD) NTCs in opposite corners of the load's region. On top of them and the whole bank of resistors, a heat sink should be fitted. Without the heat sink, the NTCs are unable to properly monitor the resistors that are further away from them. Moreover, the heat sink increases the heat dissipation allowing for more aggressive balancing.

For the available space in the battery of *FST-05e* and considering the PCB shape of the module, it was chosen a standard ¼ brick heat sink. The low profile is a direct result of the small volume in which the module would fit. The slave module fitted with the anodized aluminium heat sink is presented in figure 4.9.

Most heat sinks are electrically conductive and even if they aren't (e.g. anodized heat sinks) an insulating compound should still be used given they might become so. In the event of surface damage to the heat sink, this pad should prevent short circuits across the balancing lines. For this isolation pad, it was used the Bergquist GP1500R [The Bergquist Company, 2011].

The slave also monitors the temperature of the *dsPIC* region through another NTC and the temperature of the *LTC* die through the SPI bus. If any temperature is ever above specified thresholds, balancing is interrupted and later re-enabled when the temperature violation is over. These thresholds are different for each region according to specific ratings and functionality.

4.4.2 Exceptions

In certain cases balancing is overridden by the slave module notwithstanding the master's commands or the condition of the slave itself. A quick overview of these emergency situations is given in the following list.

- ◇ The slave won't accept a target lower than `V_ALERT_L`. If the master tries to do so, the slave will set target to the minimum `V_ALERT_L` and emit a message reporting the actual value being used.
- ◇ If slave temperature is critically high (`TS_CRITICAL_H`), balancing is disabled as long as every cell is below `V_CRITICAL_H`.
- ◇ If an overvoltage is detected (`V_CRITICAL_H`), the respective balancing channel is activated regardless of master orders or even slave temperature. It's safer to burn the slave reducing as much as possible the amount of energy inside the cell.

  If the cell starts to burn, it will have a lower amount of energy to release and the slave module will have (likely) a better behavior burning than the cells. However, installation of the modules should be taken into account before opting for letting the slave burn in a 'controlled' manner, especially if, unlike in *FST-05e*, there isn't a fireproof barrier between the slaves and the cells.
- ◇ Any malfunction of the slave module will result in the deactivation of any lines currently balancing regardless of overvoltages, either through software error handling or through a watchdog timer built into the *LTC*.

  This is meant to protect the cells from discharging below their lower limit in case of a module malfunction. However, even under normal operation, the slave module has no way of protecting an external load from discharging the battery.

Finally, if a broken connection is detected, the slave's behavior is unspecified regarding the balancing. Depending on the last valid measurement, the balancing may remain active or not. Given the cell is not connected, this state is irrelevant from a safety point of view.

This unspecified behavior may have a negative influence on broken wire detection though, but the implemented routines and *LTC* features prevent it from being an issue by temporarily disabling balancing before any measurement. However, the *LTC* allows certain measurement commands that don't disable balancing load. These measurement modes are not used.

4.5 SoC estimation

SoC estimation may be done in a multitude of ways. These often require current and temperature measurements as well as high precision voltage measurements.

For *FST-05e*, the current sensor could not be connected directly to the master module given design constraints involving the rule '*Brake System Plausibility Device*' [SAE, 2014], unrelated to the BMS system. As this system was prototyped in *FST-05e*, there was no current sensor readily available and other solutions were dependent on making adaptations to another unrelated module: the torque encoder, which measures pedal actuation and implement all relevant safety features.

Since neither current measurement nor advanced SoC was a blocker issue, both in this thesis and the overall project, priority was given to stability, safety and testing. For these reasons, SoC estimation relies on voltage measurements alone so far.

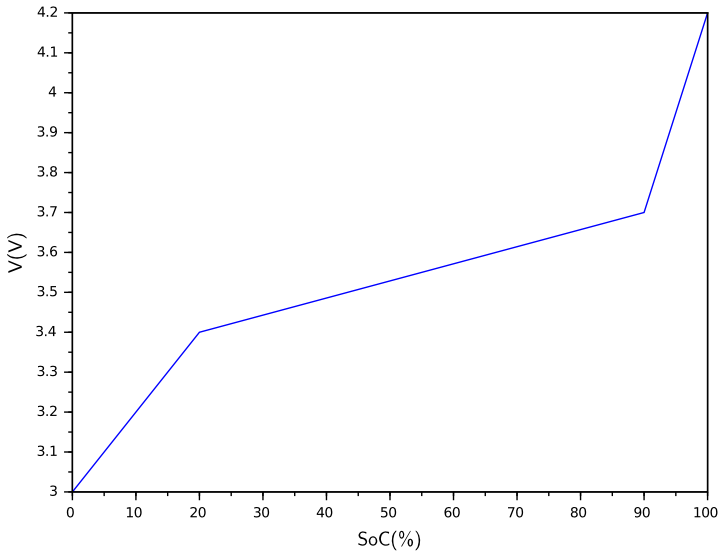


Figure 4.10: SoC mapping of voltage measurements for cells *EPS4500XP* (*FST-05e*).

The discharge curve for the chosen cells was mapped as three linear segments as shown in figure 4.10. Comparing to the actual curve — figure 2.3 —, this adjustment should give an output relative to the charge of the battery that is more linear than if a single segment was used. This means that, in the current code base, voltage and SoC are interchangeable. The systems were made independent nevertheless to facilitate further developments.

More complex algorithms should be developed against extensive data collected from the current systems. Only then validation and calibration of the algorithms may be achieved. More so, with enough gathered data, these algorithms may be developed and deployed in a simulated environment. This allows better flexibility for comparison of solutions and development to be separated from safety and stability concerns of the actual system.

Another important factor is non-volatile memory management which is not part of the slave module design. This is however relevant for storing initial conditions across power cycles of the system. Since the slave module is not supplied directly by the cells, it shuts down along the GLVS. Therefore, SoC values need to be stored if they rely on current integration.

Non-volatile memory is provided in the master module however, as the internal Electrically Erasable Programmable Read Only Memory (EEPROM) and Flash memory of the *dsPIC* are not adequate — see section 5.3.

All SoC values are internally represented, stored and communicated in parts per thousand (‰) 16 bit integers.

Chapter 5

Master module

The master module was developed in two phases. The first one comprised the development of the CAN interfaces — both the internal and external one — and implementing a minimum set of features to fulfill all applicable rules and indispensable functionality. Meanwhile, the hardware of the original master from *FST-05e* would be used.

This version was the v3.4, but further undocumented modifications were made. None of these modifications are covered in this document since they were aimed at fixing small issues with the existing hardware and removing / disabling some rule breaking circuitry. The new design already takes into account any of these modifications that are still relevant.

This new hardware was the focus of the second phase, after the British competition. This targeted an improved feature set and, above all, greater safety features. Similarly to the slave module, the new hardware is versioned from *v4.0* onwards[1].

Regarding the hardware design, the components were chosen for 24 V systems given that it's targeted more closely to the fabrication of the *FST-06e* prototype, which is due in the start of 2015. Still, these components were carefully chosen to have 12 V supply counterparts with the same footprint, including DC/DC converters and relays. All other components are indifferent to this requirement, whether from being chosen to be so or from being architecturally indifferent — e.g. a component supplied by a DC/DC converter within the module.

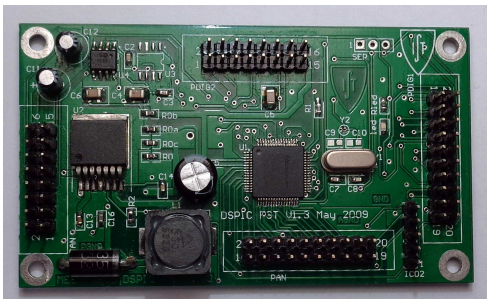
The schematics for the master module are presented in appendix D and the production masks of the first version are documented in a 1:1 scale in appendix C. Both for hardware and software development, it was used the same tool chains used for the slave modules.

5.1 Architecture

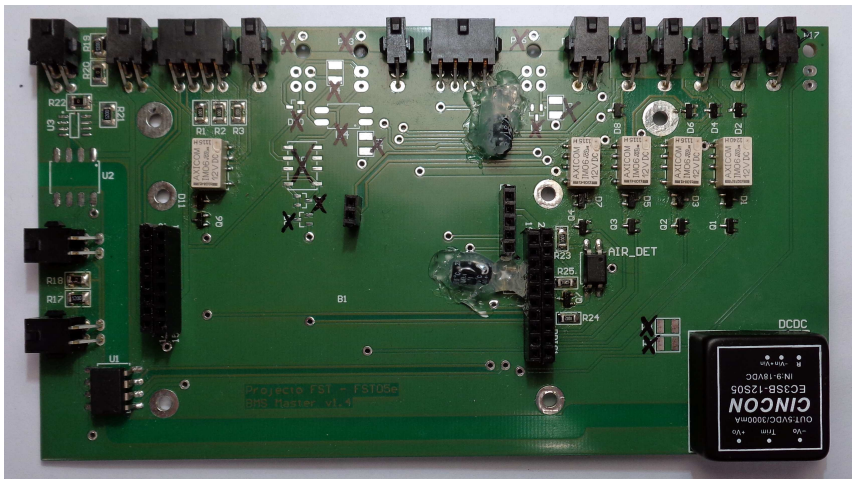
Both the original and the newly designed master module share the same *dsPIC* microcontroller with the slave modules. The reason was once more the faster development time provided by a widely adopted platform.

The original module only had CAN and I/O functions, whereas some others were never implemented in software, such as the current intensity measurement through a Hall-effect sensor. Therefore, this module was hardware limited to CAN communications, internal and external to the battery, as well as the control of the AIRs, pre-charge and discharge circuit.

[1] Given previous rule changes, the master module went through more iterations than the slave. For this reason, the master module is currently a full version number ahead of the slave — v4.x versus v3.x. There's no direct correlation between slave and master hardware version numbers, the compatibility is given by the software interface between the two.



(a) *Módulo CAN PIC FST* v1.3. Top layer.



(b) Master module shield. Top layer.

Figure 5.1: Original master module (v3.4) components.

The module was composed of two smaller modules. One is a *Módulo CAN PIC FST* [Almeida, 2009], which holds the *dsPIC* and non isolated CAN functions, being extensively used as a (sub)module in several systems within the EVs prototypes from Projecto FST. The other one is an I/O extending shield that also provides galvanic isolation to the power supply of the slave modules and internal CAN channel. These submodules are shown in figure 5.1.

The proposed module has a similar architecture except for the greater feature set. This architecture is illustrated in figure 5.2, where the blue blocks correspond to improved or revised functionality and the red ones to new features.

This new module was not prototyped given the greater care put into the software and hardware design as well as the time constraints of this thesis and the calendar of the team. However, this module aims at full compatibility with the software developed for the original module. The necessary adjustments consist only of changing some I/O ports since some pins being used in the original module were specifically required for new features. Implementation in a different car than *FST-05e* may however require changing some specific behavior — e.g. the control of a different pre-charge and discharge circuit such as in *FST-06e*.

Among the additional features, there are a Secure Digital Card (SD card) and a Real Time Clock & Calendar (RTCC). These also require an extra Low-dropout regulator (LDO) to provide them with power and, for the RTCC, a small battery to provide power while the master module is turned off. This will hopefully enable more meaningful statistics over a complete season and allow more options for SoC estimation algorithms.

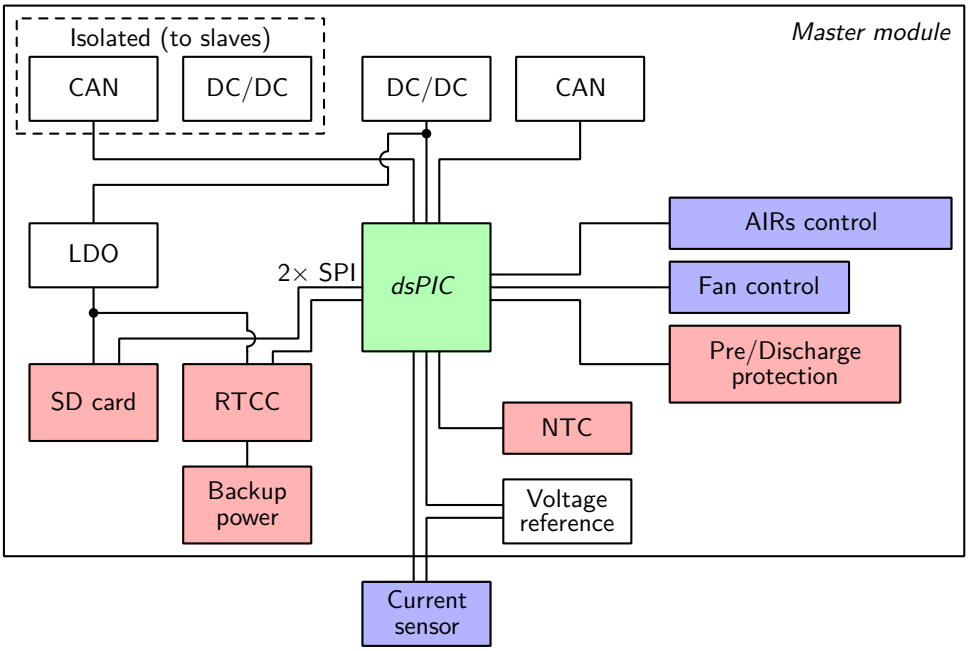


Figure 5.2: Master module architecture.

5.1.1 Layout

For the layout of the new module, like in the original one, it was considered the use of the *Módulo CAN PIC FST* and developing a shield to add the desired features. However, initial geometric specifications for the module favored the integration of the *dsPIC* in the master module for a reduced PCB.

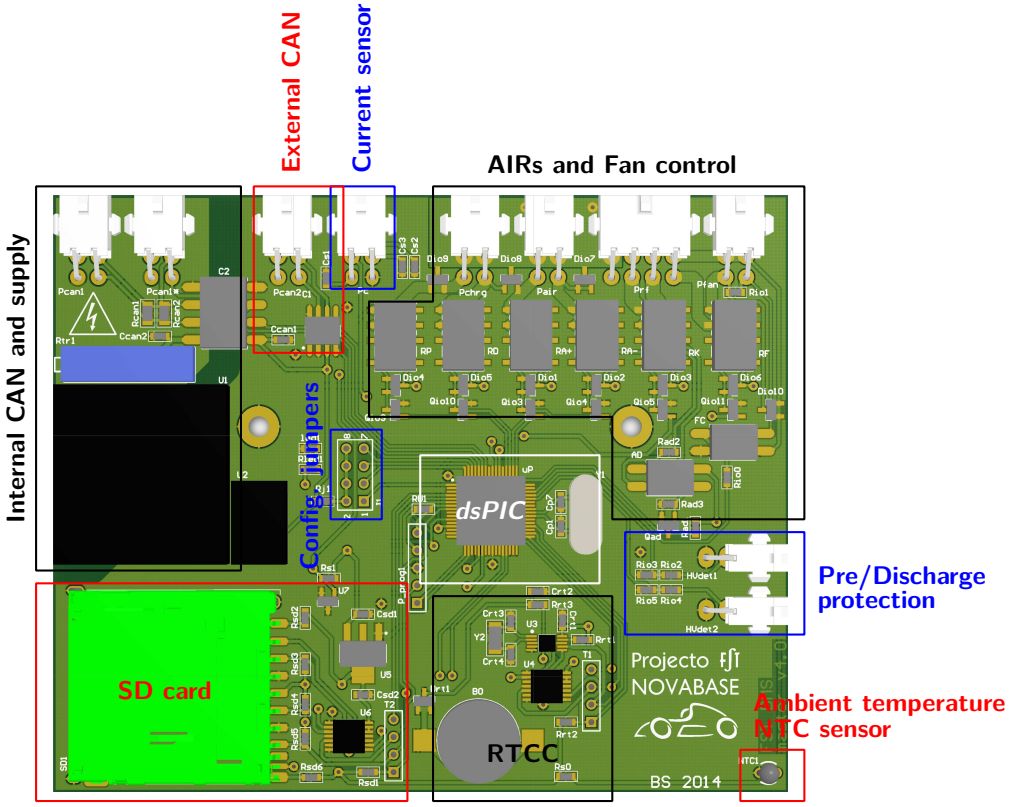
The main disadvantages of this approach are the additional points of failure as well as the greater cost of replacing the module in case of breakdown. However, the version proposed here is completely integrated, which yielded a PCB significantly smaller and of equivalent complexity. More so, this allows more versatility in the licensing of the module, which was also a concern for the version proposed within this thesis.

This new module aims primarily to be installed in *FST-06e*, for which reason it's in a pre-production stage. The actual prototyping will happen at discretion of the team from Projecto FST, therefore the images presented here are renders of the initial proposed version of the module. These renders, properly annotated, are presented in figure 5.3.

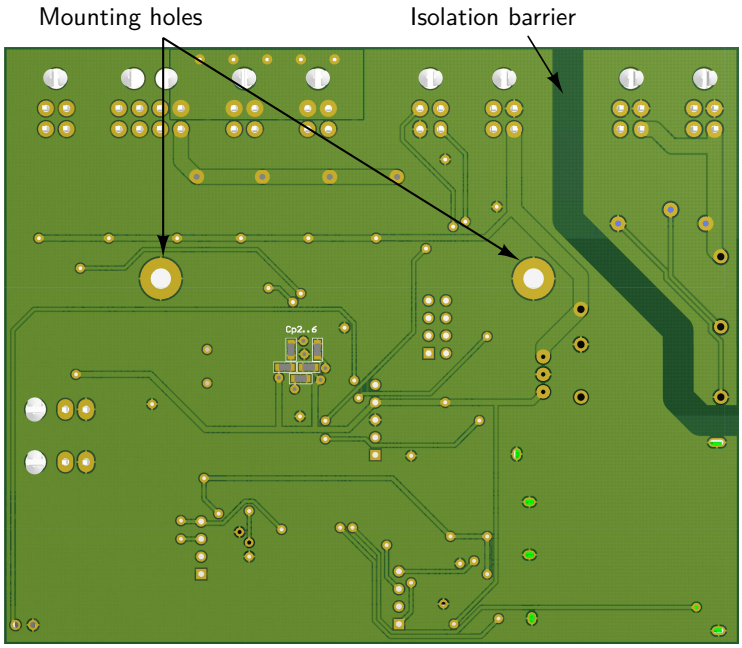
5.1.2 Routines

Similarly to the slave module, a RTOS was considered for the master, especially since the master was predicted to have a more complex code base. As a first approach, the tasks were enumerated, namely the number of loops and specific functions required. In an interruption driven code, the chosen architecture was highly dependent on only three timer interruptions (out of 5 available) and two CAN channels and respective interruptions.

The timers required would provide time keeping, current intensity measurements and slave monitoring. The remaining interruptions needed were for incoming messages (CAN) and several safety related I/O functions like detecting a power failure on the AIRs supply line. Other required functions were a single ADC to measure current and temperature (two were used for simplicity) and several I/O ports. All other sub-tasks would belong within one of these task groups.



(a) Top layer.



(b) Bottom layer.

Figure 5.3: Render of master module v4.0.

The chosen *dsPIC* is more than capable of providing all these interruptions and features without extra abstraction layers. Besides, once more, unfamiliarity with RTOS deployment also favored an interruption based approach. However, because of this decision, nested interruptions are inevitably more prevalent in the master module, which may create complex debugging situations.

In a latter development stage, some (wrong) assumptions on the best way to implement certain routines proved this analysis to be optimistic. Indeed, the software of the master significantly increased in complexity as small adjustments to the implementation were introduced, despite the hardware capabilities and simple base design.

Tools provided by a RTOS like multitasking, scheduling and an 'unlimited' number of routines could provide a more scalable and easier solution on a long term plan. Routines ranging from a message forwarding sub-system between each CAN channel, to TS activation were all hindered by the lack of these features. However, no attempt was made to revert this decision within the time frame of this work though. Doing so would result in a loss of stability or stagnation of development and testing of the system as a whole.

Finally, some code paths still have limited functionality in the current stage.These are non-blocker features that were waived or only partially implemented in favor of more critical features and improving stability — e.g. live reconfiguration. Additionally, some features were not possible to implement and test with the original module, therefore those are not fully developed either — e.g. RTCC functions.

5.1.3 Master-slave coordination

The main loop comprises most of the master-slave interaction. It consists of checking for abnormal values or unresponsive slaves, synchronizing all modules to the same function and placing new queries. The main loop is completed by broadcasting a resume of the battery state and reseting the watchdog timer before returning to an idle state.

The response handler is triggered by the CAN module, making the query process completely asynchronous. However, the hardware buffer can only hold four messages at a time, but each slave is queried for more than ten. For this reason, nested interruptions are enabled and the CAN module is always given a higher priority.

To handle missed responses — e.g. when a slave module is unresponsive —, a maximum number of faults is allowed for each slave and each query individually. This allows to adjust with greater granularity the conditions that result in a BMS error and subsequent shutdown of the TS.

Additionally, any abnormal value is queried two more times before committing to raising a BMS error. Therefore, the response time to a parameter violation should never surpass $\approx 3\,\mathrm{s}$ by design, but sporadic emergencies are still ignored by the system. This feature is implemented as a cool-down effect for each fault.

The master is also responsible for forwarding queries to the slave modules upon external request. The internal messages are not usually present in the outside CAN, but the master provides abstraction values for the overall state of the battery, both under normal operation and emergencies. The user may request for this data however, which prompts the master to make all parameters available through the isolated interface.

5.1.4 Master-master coordination

For systems with more than one master, the proposed approach is a set of jumpers that are able to set which master is responsible for certain tasks. Effectively, one or more masters are slaves to another one in a number of critical functions. This avoids race conditions in the decision trees and, consequently, the complex logic that would be required otherwise.

The only parameter set in a distributed manner is the balancing target. Since every master reports the state of its battery, all other masters are able to track the lowest voltage of each container / master and compare with their own. Therefore, each master has the responsibility of obeying this rule independently to achieve a balanced state.

This exception is possible since a loose synchronization of the balancing target has no safety or even functionality concerns. The only problematic case would occur for batteries that were already closely balanced. The synchronization delay could resonate with the voltage variation produced through balancing and prevent the system from stopping the balancing process. However, these variations are very slow making these conditions impossible to be met without greatly increasing the balancing current.

5.2 Tractive system control

TS control is achieved by a set of small relays to power the AIRs and the pre-charge / discharge relays. The choice of using relays in the master module instead of power mosfets is to allow greater flexibility in the power and isolation options of the high current / high voltage relays, also improving overall safety.

More so, all these relays are supplied by a single line that creates a large safety loop across the car.Many modules, including the master, may interrupt this line independently forcing the TS to shutdown. Ensuring the correct polarization of successive mosfets would be difficult, therefore all modules use relays instead.

The actuation of these relays is a safety critical system that can endanger the whole battery, car and driver. Therefore, the master has extra I/O features to avoid or allow an early detection of dangerous situations, including the failure modes described in section 3.6.2. These features and respective redundancies and failure coverages are detailed below:

1. Relay supply detection (in-master; interruption driven):

   Detects situations when the AIRs were opened by any another module, a manual switch, an interlock or a wiring fault. This situation, as dictated by the rules, triggers the fail-safe discharge of the capacitance in the TS. Without further action, a sudden recovery of power may close the AIRs without performing the pre-charge routine — i.e. the master module wouldn't notice that the battery had been shutdown.

2. High voltage detection on battery poles (external module; preemptive checking; interruption driven):

   This allows the master to check if the pre-charge was completed. If the voltage at the battery poles doesn't match the internal voltage, the AIRs should not be closed.

   Assuming the external module only detects a high voltage — i.e. doesn't compare voltages —, there may still be a noticeable difference, not covering partial pre-charges.

   Serves as a redundancy test for feature 1.

3. High voltage detection on pre-charge / discharge resistors (external module; preemptive checking; interruption driven):

   Makes it possible to monitor the discharge circuit, i.e. if the discharge relay is closed when it should be open. In this situation, the discharge circuit would be permanently connected in parallel with the motor controllers. The TS should be shut down or kept that way to prevent discharging the battery or damage to components.

   Makes feature 2 redundant for checking the pre-charge, and also makes it possible to detect partial pre-charge cycles. However, it doesn't make item 1 redundant.

A Tractive System Active Light (TSAL) module has to be installed in each battery container for additional safety and rule compliance. This module monitors the battery poles for the presence of high voltage outside the battery container and provides visual cues to inform of this state.

This module can provide the detection for item 2 in the current implementation by Projecto FST. For feature 3, a slightly modified or even the same TSAL module may be installed across the discharge resistor — here, however, the TSAL name has no relation to the module's function.

5.3 Data storage

A data logger is important for development purposes, but should be provided in a different module. Nevertheless, there are other reasons for having non-volatile memory within the BMS other than data logging for posterior analysis. The predicted situations requiring such feature are listed below:

- ◇ Storing configuration parameters, status flags and emergencies events that are relevant after a complete power cycle.
- ◇ Storing initial conditions and parameters required for SoC estimation or any other performance value calculated within the BMS.

In the first case, the memory requirements are low in capacity and longevity since these parameters should be few and are not changed too often. Therefore, the EEPROM or the flash memory within the *dsPIC* are appropriate to store them. Indeed, this is the only option for the original master that implements some safety critical functions with the internal EEPROM.

For more demanding tasks, the memory of the *dsPIC* has several downsides. For instance, advanced SoC estimation techniques involve current integration, requiring initial values, and some depend on other parameters like confidence values — e.g. Kalman filter method. All of these requiring non-volatile memory that should be provided by the master module in the chosen architecture.

The flash memory provided by the *dsPIC* could be enough, depending on the specific implementation, if the data was stored within the slave modules individually. However, the internal memories of the *dsPIC* have very low performances for constant access. More so, memory alignment issues makes ⅓ of the memory even more expensive in terms of performance.

Additionally, the lack of a memory controller, relative small capacity and the minimum addressable block size drastically reduce life expectancy of the *dsPIC*. Indeed, for storing SoC values alone, a slow 0.1 Hz logging rate would probably result in bad sectors within ten days of uptime — too short even for a racing prototype.

For these reasons, a SD card is provided in the master module. The SD card has a very large (variable) amount of memory, a memory controller that significantly extends the life expectancy of the card and provides very fast access times through a SPI bus.

Relative to soldered solutions, the SD card provides a swappable volume and thus greater code unification. With it, every battery specific behavior may be provided as configuration parameters inside the SD card. Replacing master modules or recovering data from a damaged one are two examples facilitated by this approach.

Finally, this SD card is to be accessed through *Petit FatFS* library [ChaN, 2014], which is also used by the data logger — see section 6.1. This means, at the time of writing, there's already a tested code base with this library in the same *dsPIC* although the new master hardware is yet to be prototyped.

5.4 Current intensity

The chosen sensor intensity for current measurement was a *HTFS 200-P/SP2* Hall effect sensor [LEM, 2007]. The nominal current range of the sensor is $I_{PN} = \pm 200\text{A}$, with the absolute maximum $I_{PM} = \pm 300\text{A}$. Additionally, it has a fast response and allows for high acquisition rates.

This sensor provides complete galvanic isolation and the only limiting factor is the error of 1 % relative to I_{PN}. Not withstanding offset errors — characteristic of Hall effect sensors —, this yields a precision of 2 A. This is a small error under large loads but significant, for instance, when charging. However, this precision is good considering the technology being used and the design considerations.

For the *dsPIC*, this yields a minimum voltage to read of 16.7 mV as shown in equations 5.1 and 5.2.

$$\frac{2 \cdot I_{PM}}{5\text{V}} = 0.12\text{A mV}^{-1} \tag{5.1}$$

$$\frac{0.01 \cdot I_{PN}}{0.120\text{A mV}^{-1}} \cong 16.7\text{mV} \tag{5.2}$$

The sensor is read directly through the 12 bit ADC of the *dsPIC* and, for improved precision, a voltage reference is generated within the master module. This ADC is then able to read a minimum voltage of:

$$\frac{5\text{V}}{2^{12}} \cong 1.22\text{mV} \tag{5.3}$$

Therefore the minimum current possible to measure would be given by equation 5.4. Given that $0.15\,\text{A} \ll 2\,\text{A}$, it's safe to assume the ADC doesn't limit the precision of the current intensity measurement further than the error of the sensor.

$$I_{min} \cong 0.120\text{A mV}^{-1} \cdot 1.22\text{mV} \cong 0.15\text{A} \tag{5.4}$$

5.5 Charging

The master module, when set to charging mode, has the function of coordinating the balancing of the cells. Additionally, if a charger module is present on the CAN bus, the master is also able to request a lower current.

The master (the main one, if more than one is present) only informs of the maximum current it can be safely sourced to the battery. It's the slave modules that eventually report that cells are going over their limits, prompting the master to interrupt the charging process through the AIRs.

An exception should be implemented for absolute maximum current ratings, but this was not possible in *FST-05e* without the current sensor installed. More so, the charger being used by the team is unable to charge the battery at the maximum rate. This could however be relevant for a different battery and / or charger.

The BMS achieve complete protection of the cells through the AIRs. In the worst case, with a charger that doesn't interact with the master module, the control of the AIRs enables a full charging cycle with one of the most common profiles: constant current followed by constant voltage. Optimized fast charges may not be possible however.

If any irregularity is detected in the charging process, the master may interrupt the influx of current. Any extra safety measures should be implemented by the charger module and respective installation.

Chapter 6

CAN tools

There were a set of tools required for interfacing with any embedded system developed within the team and, specifically, with the BMS. These were meant for development, debug and everyday usage purposes. This set of tools should comprise several components listed below.

- ◇ CAN-USB translator: provides an ordinary personal computer with a CAN interface.
- ◇ CAN console: a console or console-like application for receiving and sending CAN messages implementing the required protocol to communicate with the translator.
- ◇ BMS module: an abstraction layer built on top of the console and underlying protocol.
- ◇ Other (independent) modules for interfacing with specific systems or add functionality to these tools — e.g. database module or telemetry. These are well beyond the scope of this thesis and are mentioned for completeness.

The existing tools at the start of this project covered the first two items. However, these had a few shortcomings that made them unfit even for the tasks they were developed to fulfill. The shortcomings were few, but had great repercussions:

- ◇ Communications were too slow loosing most messages within the CAN bus.
- ◇ Protocol was unable to handle all messages — e.g. only 8 bit are reserved for 11 bit IDs.
- ◇ Translator required constant power cycles to clear software lockups.
- ◇ Aging software required Windows XP.

An alternative was then needed at least for the software, which was clearly the main source of these shortcomings. However, improved or redesigned tools should fulfill the requirements for debugging or interfacing with any module, rather than the BMS alone. The low level tools should therefore be agnostic to any BMS specific protocol.

The solution was a complete redesign of the tools, for convenience and development time considerations, but also for technical issues. The following sections detail the translator module and the software developed to fulfill this purpose.

The translator module makes use of the same Microchip tool chain used for the slave and master modules. As for the software targeted at the computer, C++ was chosen for the versatility and performance.

To provide a GUI, the Qt framework 5.3.2 was used for ease of portability and future integration with other tools from Projecto FST. This software stack was developed in C11 compliant C++ using the GNU is Not Unix (GNU) tool chain (gcc v4.8.3). The developed software targets x86_64 Linux. This program will be hereby referred to as *FST CAN interface*.

6.1 CAN-USB translator

This module was intended to overcome the limitations of the translator being used by the team up to this point. The main challenges for this module were then the performance and the communication protocol. While the same hardware could be used, a similar platform was available: the *Módulo CAN PIC FST* (figure 5.1a).

Familiarity with the hardware was crucial to speed the development of this critical module. The translator was then developed with a *Módulo CAN PIC FST* together with a Universal Asynchronous Receiver/Transmitter (UART) to USB adapter. Furthermore, a USB-USB galvanic isolator to protect the computer and car electronics is required.

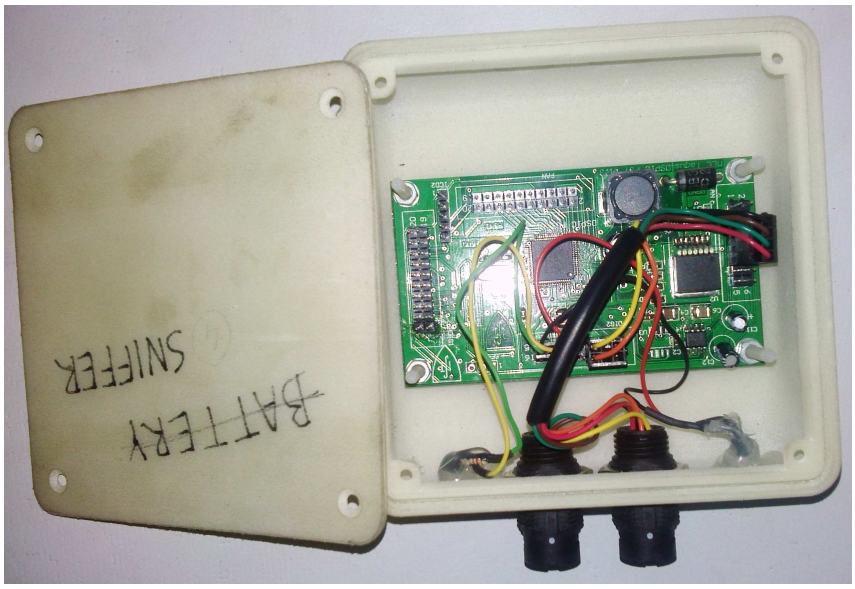


Figure 6.1: Translator module. Left connector connects to CAN and the right one to USB; Light Emitting Devices (LEDs) indicate bus activity.

Since one of the issues with the original translator was performance, this decision was necessarily backed by a proof of concept. An earlier analysis proved, as suspected, that the hardware was not the issue, but rather the protocol.

The original module encoded the messages in the UART/USB bus in American Standard Code for Information Interchange (ASCII). This has a time penalty in the message encoding and in the required bandwidth, leading to less than 50 messages per second. Indeed, with the new hardware, there was no significant improvement when using ASCII encoding and running at 60 MHz.

With a binary frame and an appropriate Cyclic Redundancy Check (CRC), this figure improved to more than 4000 balanced messages per second — i.e. messages uniformly spread through time. A full implementation reduced this value to just under 3000 messages per second. The bottleneck was found to be in CRC calculation and the UART baudrate by of factor of 5 to 1, but only 2 to 1 after compilation with the

optimizing flag -O1[1].

The use of optimization flags in this module was considered acceptable since the code is significantly simpler than the code of the slave module, for instance. More so, the translator has no safety critical functions and could be easily tested under all use cases for the different compilation flags.

Further improvements could be made with a different CRC and increasing the frequency of the MPU. The first one could explore improvements that were overlooked in favor of a faster development of the initial solution. However, an increase in frequency alone, which may be as high as 120 MHz, is able to halve the CRC calculation time and double the baudrate of the UART interface.

However, the obtained performance with buffers to handle messages bursts was still above 3000 messages per second, and higher frequencies could require heat sinks for the MPU. Since the performance was already enough to comply with the design parameters, no further optimizations were made and should only be required for busier CAN buses.

These speeds refer to the download from the CAN bus to the computer. For the opposite direction, communications should be very limited in the expected use cases, and it was consequently limited to just 5 messages per second. Considering these are manually sent by the user, this is a reasonable rate that favors the throughput of the download link.

Figure 6.1 illustrates this module as developed within this thesis. Since this revision, contributions from members of Projecto FST expanded this module with logging features. The module has since then two operating modes, where the second one allows logging all data within the CAN bus to a SD card and supports 2 CAN channels.

6.2 FST CAN interface

All software developed to interface with the translator was designed as a single application with a GUI. This interface currently exposes to the user two main 'modules': the *Console* and *Battery*, currently arranged as two tabs.

There is only one source / sink implemented: a serial port (USB). However, the software is layered such that there is a module responsible for handling the serial port, but others could be implemented through a class abstraction layer.

Although greater functionality was accounted for in the design, to this date, the features implemented only cover the objectives of this thesis. The overall architecture and implemented features are described in figure 6.2.

6.2.1 Console module

The console provides a simple way of observing the stream of messages within the CAN bus and send any valid message. No assumptions are made regarding the signed or endianness nature of any value or indeed their length. This module is illustrated in figure 6.3.

This makes it still hard to evaluate negative parameters for instance and further switches to configure how the parameters are printed and inputed could be beneficial. However, these features were deemed as future improvements.

[1] This corresponds to the maximum optimization provided by the free version of XC16 compiler.

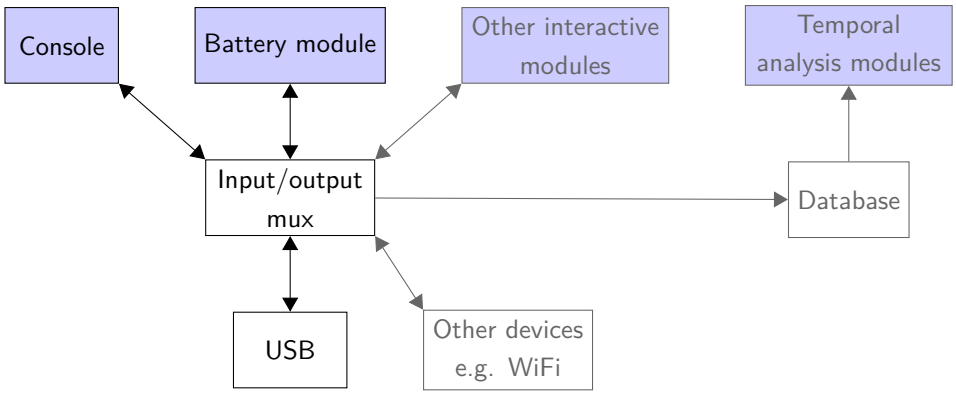


Figure 6.2: *FST CAN interface* architecture. Blue blocks correspond to GUI interfaces. Grayed out blocks represent features not implemented nor covered in this thesis.

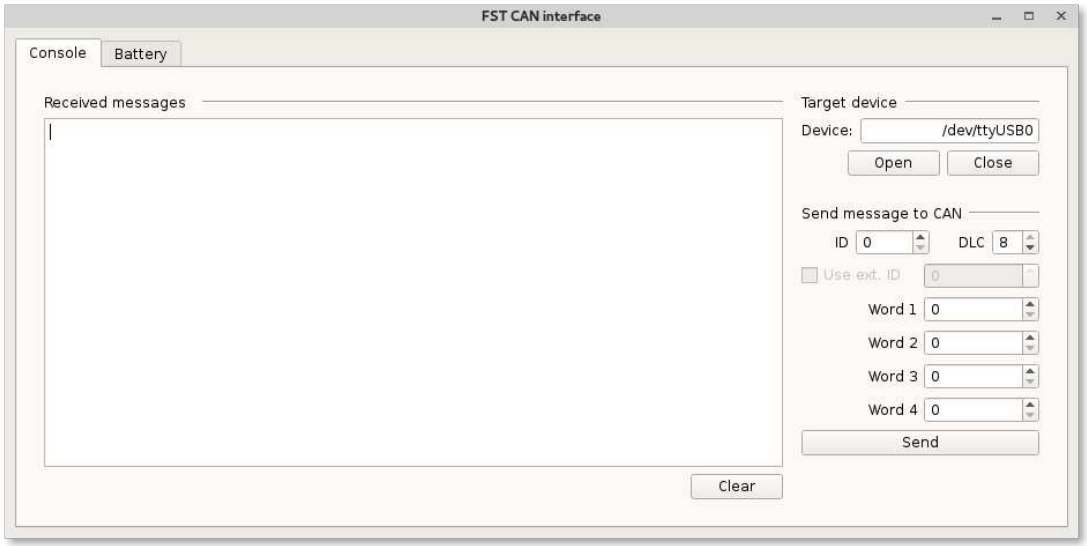


Figure 6.3: Console tab.

6.2.2 Battery module

The battery tab provides all the abstractions necessary to interpret BMS specific messages as well as issuing battery specific commands. This module aims to provide a control panel for the BMS subsystems and is illustrated in figure 6.4.

Some features were not fully developed and are grayed out as placeholders for future features. Notably, the lower scrolling zone that should provide an overview of each slave module was not developed. This allowed to shift the focus to the detailed views and the embedded software that enabled the output of the relevant information.

The current value and energy consumption rate are also grayed out since in the development time frame there was no current sensor installed and both values depend on it. However, the balancing rate is fully reported by the BMS through a set of binary masks, but again, was deemed not to be a priority feature for the panel interface and full integration was pushed beyond `git:tag:THESIS_DELIVERY`.

Since safety is a priority, the battery module has a set of watchdog timers that clear the information displayed to a default state. This is essential to prevent the user to assume the BMS was online and working

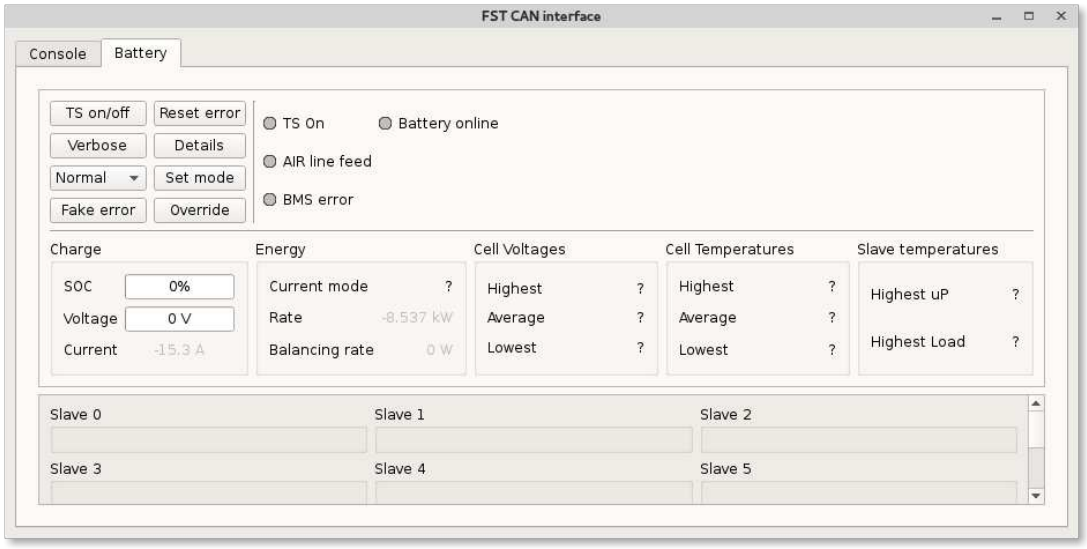


Figure 6.4: Battery tab. Battery offline.

correctly when it wasn't.

The default state is characterized by interrogation marks in place of values and grayed out 'lights'. These 'lights' represent status flags emulating multicolored round LEDs in a panel — these are gray circles next to the buttons in figure 6.4.

Display of detailed data for every cell was a required feature for rule compliance and an important one for development and tests. The required data consisted of individual voltages and temperatures. On top of these, wire faults (or lack thereof) are also indicated both both cell poles and NTCs connections — a relevant feature considering *FST-05e* has a total of 300 wires connecting cells and temperature sensors. This interface was made in a different window for visibility purposes — see figures 6.5 and 6.6.

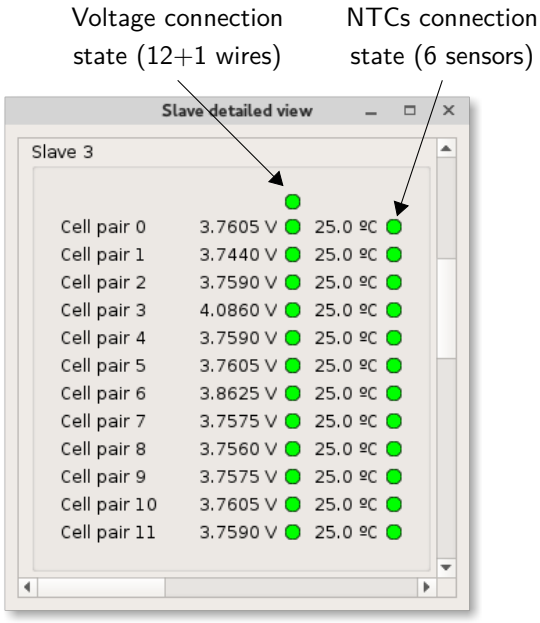


Figure 6.5: Example of detailed information for slave 3 under normal conditions.

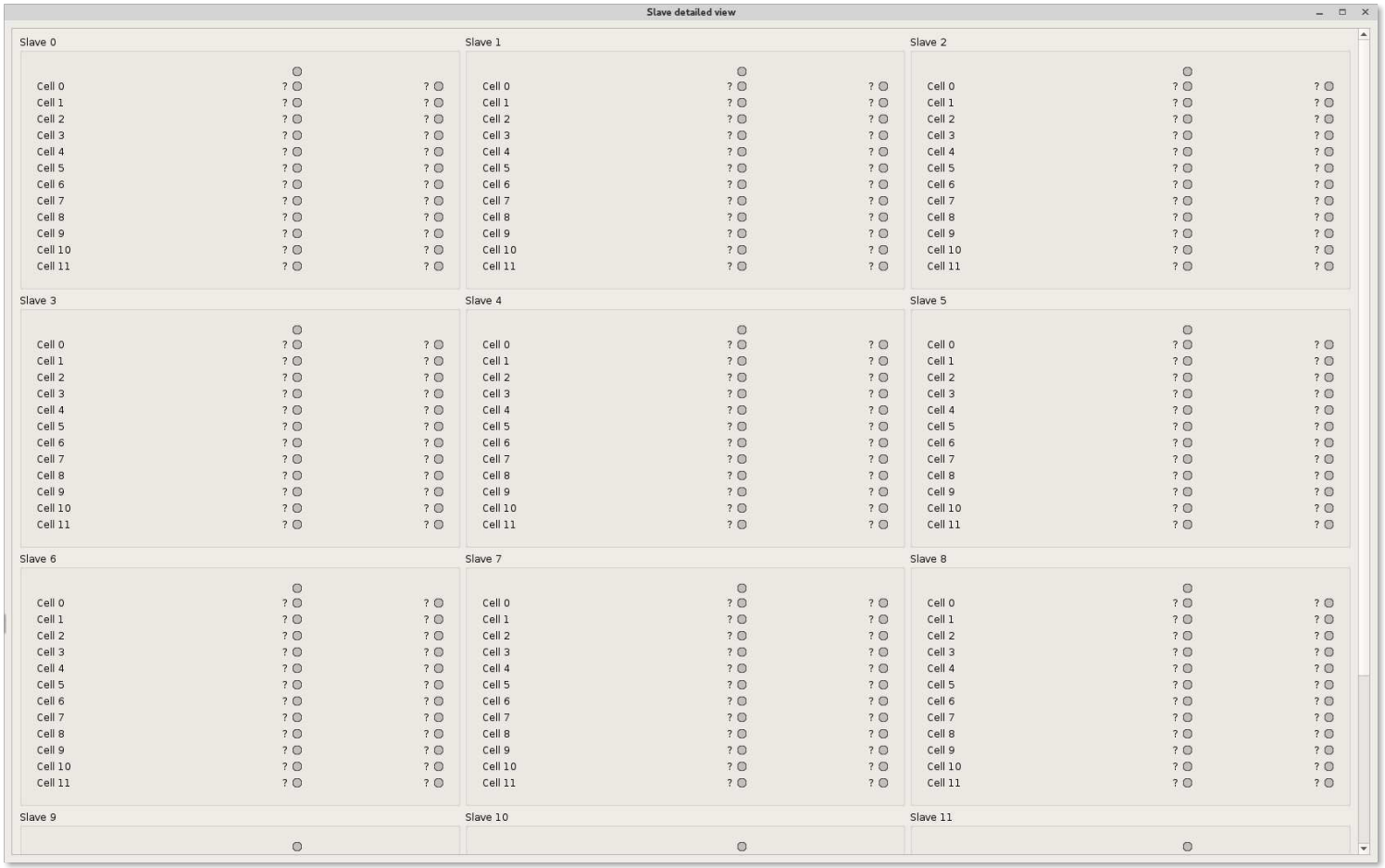


Figure 6.6: Battery detail window.

Chapter 7

Tests and deployment

The new BMS underwent a progressive integration phase until the final deployment in *FST-05e*. The approach secured safety critical features early on and pushed others towards the final stages of integration — e.g. voltage monitoring versus balancing strategy.

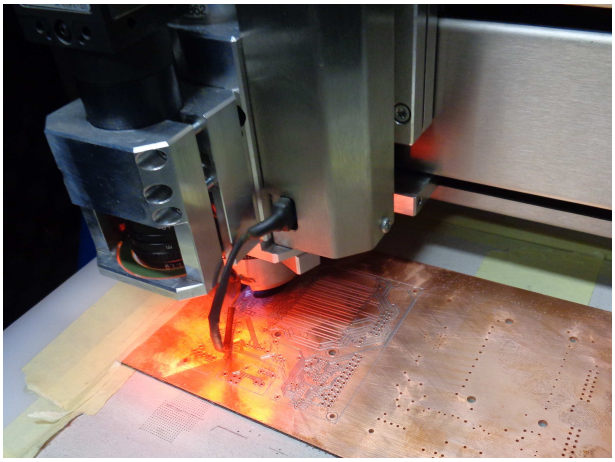
The development of all systems followed the stages represented below, although the list doesn't intend to document the exact development order. The integration in the actual battery was made possible once the master module had the basic monitoring loops and communication features implemented.

1. *Slave module:*
   - ◇ Prototyping and individual tests to each hardware function.
   - ◇ Early assessment of design compliance — e.g. confirming accuracy and range of measurements.
   - ◇ Revision and send for production (PCBs only).
   - ◇ Soldering, testing and validation of each slave module.
2. *Master module:*
   - ◇ Development of the master-slave protocol and monitoring loops.
   - ◇ Development of BMS control interfaces, i.e. interface with control panel, etc.
   - ◇ Development of TS control interfaces, i.e. AIRs control, etc.
3. *Interfaces:*
   - ◇ Development of the CAN-USB translator. Early usage through program *CuteCom* [Neundorf, 2014].
   - ◇ Development of custom console module and underlying modules.
   - ◇ Polishing and feature expansion through the battery module.

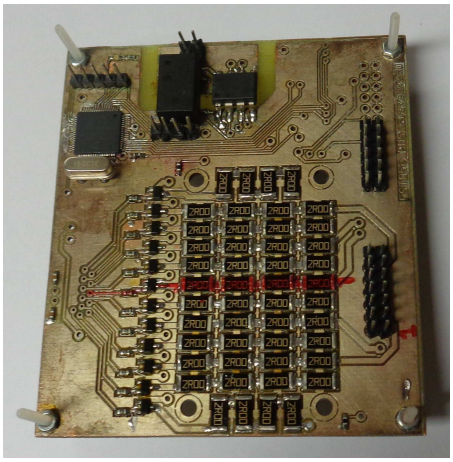
For the entirety of this chapter, references to the master module imply version v3.4 since it's the only version effectively integrated within the system. Additionally, all references to the slave module are in the context of *FST-05e* prototype.

7.1 Tests and development

The prototyping phase targeted two slave iterations, but fortunately only one was required. This was a great advantage of the chosen components and crucial for the early deployment in the actual battery.



(a) Production of prototype in IST Taguspark PCB laboratory.



(b) Top layer.

Figure 7.1: Prototype slave v3.0.

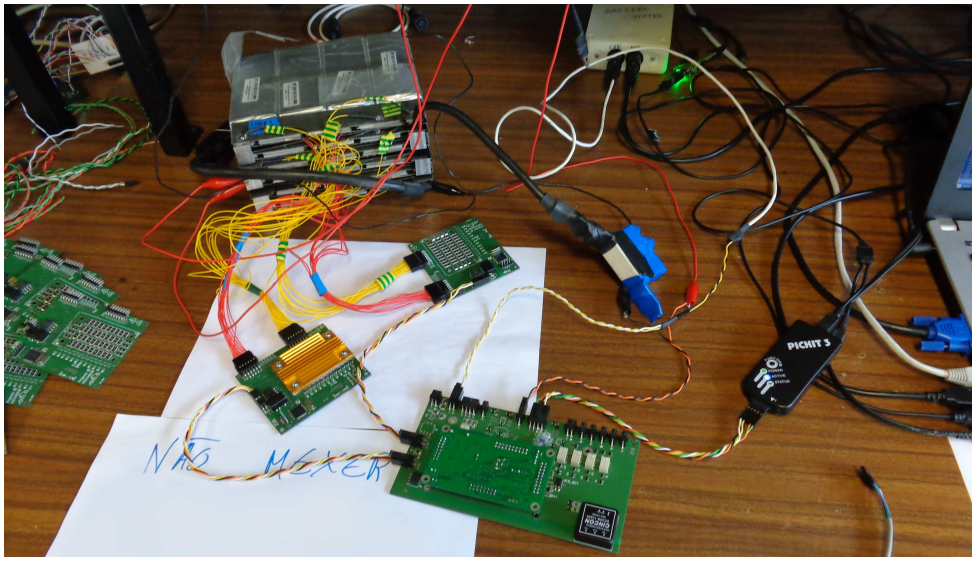


Figure 7.2: Master v3.4 and slave v3.1 integration tests.

The tested basic functions were the CAN interface, voltage measurements and temperature monitoring. Voltage measurements and supply requirements were validated against an *HP 34401A* bench-top multimeter and temperatures were validated against a *Raytek RAYRST2L* infrared thermometer.

Tests targeted initially a single slave module (v3.0), which was progressively connected to a stack of batteries and basic functions developed and validated. The integration of the master module allowed for adding a second slave and develop further features regarding balancing and the master monitoring loop, including emergencies handling. This setup is illustrated in figure 7.2.

Along these developments, the translator and console were also developed as the only means of communication with the modules.

Finally, the battery wide balancing strategy was only defined once the whole system was installed. This allowed to evaluate the best options in terms of temperature management and charging performance. However, the balancing current and temperature management within the slave module was defined early on to guarantee a safe operation and integration in a 144 cell battery.

7.1.1 Design validation

Already using version v3.2 of the slave modules, these were subjected one by one to a validation process. These were meant to identify hardware failures — e.g. soldering issues —, and to verify that measurements' accuracy was within the expected values. The maximum voltage and temperature deviations observed were 5.31 mV and 8 °C.

The temperature measurements had an error slightly above the requirement, but only for the NTCs within the slave modules. These SMD NTCs have a slightly different curve, but, for simplicity, the same conversion is made for both types of NTCs. This introduces a small deviation, unnoticeable in a large temperature range, but as large as 8 °C in the highest temperatures. Indeed, for cell temperature measurements, the error was below 3 °C for tests conducted in laboratory.

This was a compromise that should be eliminated once the temperature conversion is not done within the slave modules anymore. However, for safety purposes, the errors are not significant since they're predictable and accounted for, i.e. the error is not from the measurements, but rather the conversion to Celsius degrees.

In order to test the voltage measurements, since the equipment to test the slaves was very limited, a strategy was developed to test all slaves in an efficient manner. For these tests, a controlled supply was used to simulate a single cell and one channel of each slave was measured against a precision bench-top multimeter. Two extra power supplies were needed for the CAN bus and the *LTC* itself, supplied with 40 V.

In order to avoid testing all twelve channels per slave for each slave, each module was characterized for a different channel according to the production order. Since the ADC is the same, only the multiplexer errors may differ slightly. As an example, for the first produced slave, the 1^{st} channel was used; for the second, it was used the second channel and so on. After the 12^{th} slave, a channel was randomly chosen.

The range of voltages used in the characterization was 2.5 V to 4.5 V with 10 intervals (11 measurements). The initial prototype was tested in a wider range, but for ease of testing the complete set of slaves, it was used a reduced range that represent the working conditions in *FST-05e* and *FST-06e*, with a safety margin naturally.

The errors obtained for each slave are represented in modulus in figure 7.3 and analyzed in table 7.1. In the figure, each line corresponds to one slave and, as seen in the plot, the error for any slave was always inferior to 6 mV and systematic — i.e. for one module the error was almost constant along the input range.

An important notice is that the slaves were tested at room temperature of $\cong$22 °C. However, these tests should be repeated in a temperature controlled oven to analyze deviations introduced by temperatures.

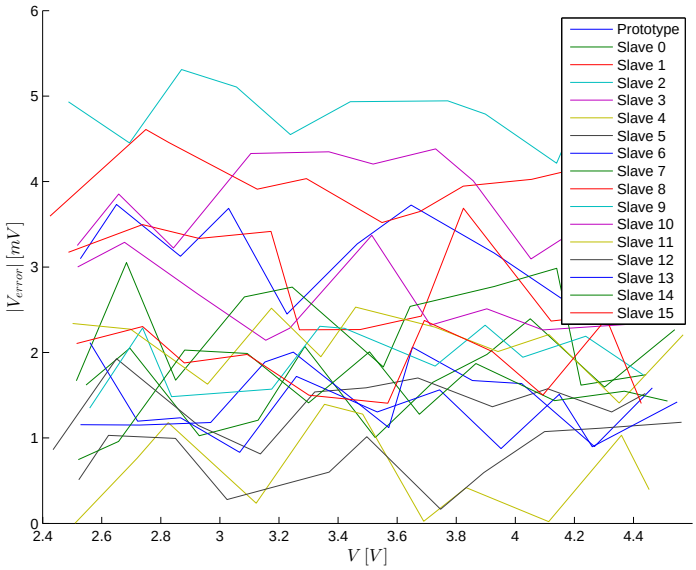


Figure 7.3: Results of voltage characterization of all produced slaves. Out of 20 slaves plus the prototype, one was damaged before characterization and 3 were never soldered.

Table 7.1: Characterization summary. These values apply to $|V_{error}|$ as in figure 7.3.

Parameter	***Value***
Maximum	5.31 mV
Average	2.26 mV
Standard deviation	1.21 mV

For this project, the difficulty in accessing the required devices resulted in these tests being waived for the necessity of deploying the system within the prototype.

Nevertheless, the manufacturer of this IC states an induced error below 3 mV up to 120 °C in die temperature (thermal shutdown at 145 °C) and absolute maximum errors of 10 mV. Validating these results would be important, but given the slave's temperature management, the *LTC* shouldn't reach temperatures higher than 60 °C. For these reasons, it was assumed the criteria of 10 mV was met under most normal operating ranges.

Despite the team's experience with CAN bus and the complete cable and power electronics shielding in *FST-05e*, EMI induced noise and glitches was another parameter that would be interesting to test. However, lack of laboratory conditions for reproducible tests and results resulted in these tests being waved for this development cycle. Nonetheless, track tests in *FST-05e* proves the system to have no glitches of any kind over several run sessions, but measurements errors were impossible to evaluate given the amount of uncontrolled variables.

Finally, another important parameter that was measured for the whole system was its power consumption. Under normal operation, this system required 0.4 A@12 V for the whole BMS electronics — this excludes the supply of fans, AIRs, pre-charge and discharge relays.

This value is well within the expected power requirements and initial measurements. Nevertheless, these values were obtained in optimal conditions, and error handling can slightly increase the energy consumption. These increases are related to higher monitoring frequencies and more self-checks and, thus, less idle time.

For the slave modules alone, elithion Lithiumate™ pro advertises a consumption average of 5.55 mW per reading per second per cell[1] [elithion, 2014]. The developed slave module consumes an average of ≅18 mW per reading per second per cell, under normal conditions.

While there's possibly a greater waste of energy in the proposed solution from the unoptimized code — e.g. converting units within the slave module —, the architecture is probably the main responsible factor for the energy efficiency[2]. In fact, Li-BMS v3 prototype advertises slightly over 40 mW per reading per second per cell and uses the same *LTC* and a PIC32, a very similar architecture to the one proposed in this thesis. However, it has consumption significantly higher, presumably because of the higher grade MPU [LION E-Mobility AG, 2014].

7.2 Deployment

The system was fully deployed in *FST-05e* and, with further recent work by several members of the team, it's expected the complete integration of the system in the new prototype by the end of the year. The battery of *FST-05e* completely assembled with the new BMS system is presented in figure 7.4.

While early integration was essential to have the battery in a usable state, it hindered further feature developments. Time on track was prioritized over time spent in laboratory conditions. This limited the number of configurations tested and iterations possible and also led to some feature branches of the source code not being merged to prevent breaking compatibility and stability of the modules.

Regarding the scalability of the system, it was not possible to stress test the master module when commanding a network of more than 12 slaves. Although it's expected to exist a bottleneck in the CAN bus, a few software limitations may be more relevant in limiting the scalability of the system.

Such limitations are bound to limited memory and processing power of this particular *dsPIC*. However, one of the best commercial alternatives for batteries of these dimensions, elithion Lithiumate™ pro is only able to scale up to 255 cells in series.

In *FST-05e*, the internal CAN bus is only being used to ≅40 % of its capacity. The master itself should be able to support transparently more than double the amount of slaves. Therefore, the proposed system with one master should support an equivalent number of cells or higher with proper configuration — i.e. tweaking packet distribution throughout time and the monitoring loop.

Moreover, the proposed system may scale well beyond 255 cells if more than one master is used. The greatest limitations in this approach would be the assignment of CAN IDs and bus occupation.

7.2.1 Charging

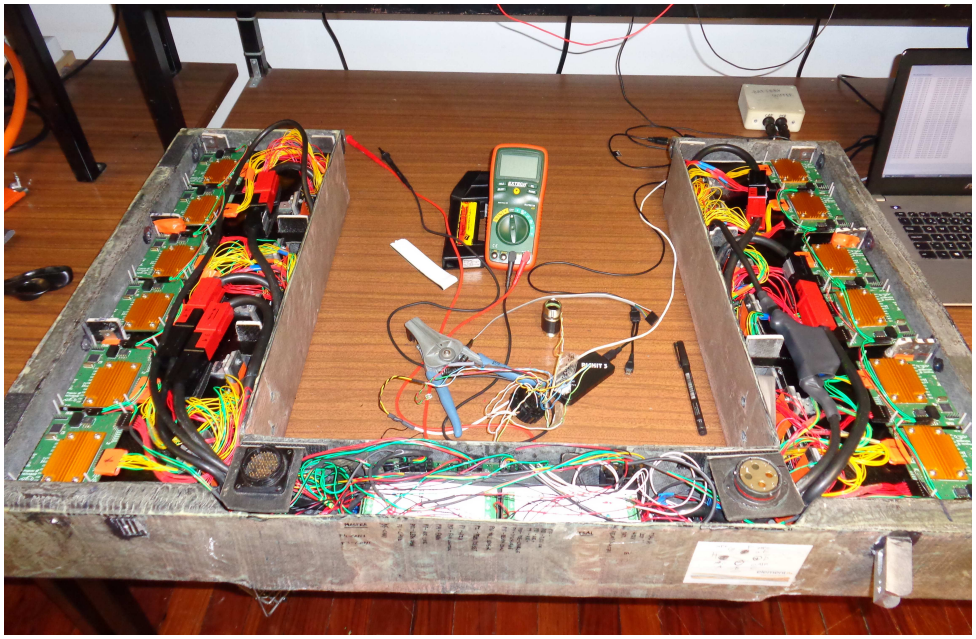
After the first assembly and later, when replacing cells and stacks, the balancing was specially relevant. In these situations the cells were the most unbalanced and several charging strategies were tested, including overnight balancing.

Figure 7.5 shows the battery almost completely balanced. In this figure, there's a single cell clearly unbalanced (in orange since it's the only cell near the maximum voltage), which was a case of a damaged balancing line at the time.

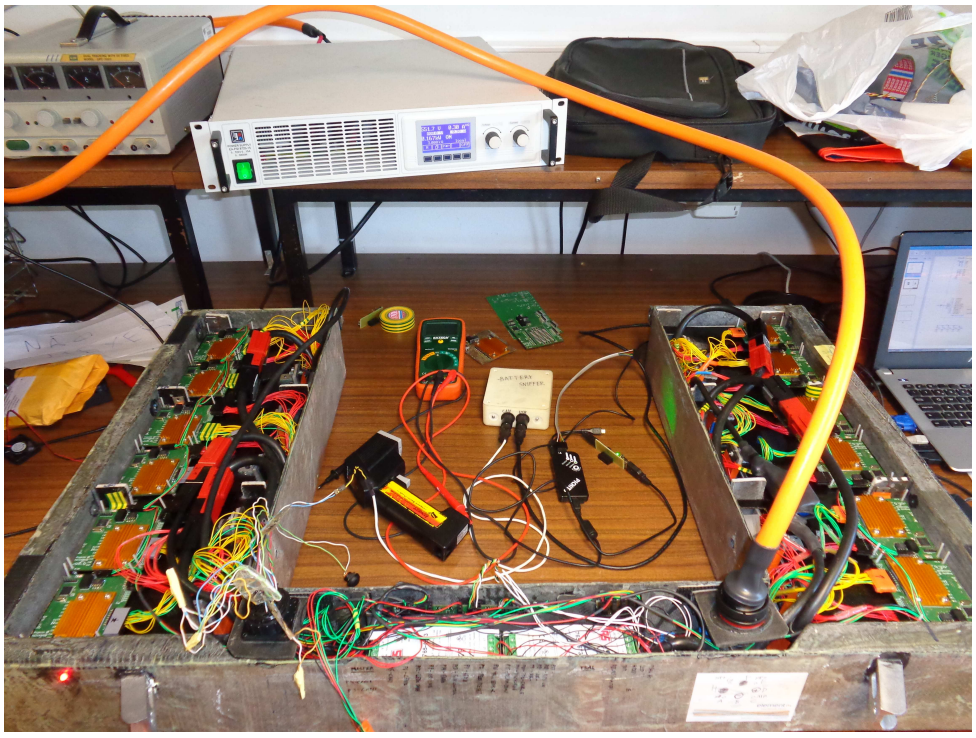
After some adjustments to the charging process, an optimally simple solution was found to generate the most heat — i.e. activate more balancing channels — in the early stages of charging. In later stages, this

[1] Value adjusted from documentation taking into account the characteristics of *FST-05e* battery.

[2] elithion Lithiumate™ pro is fully distributed with one slave per cell in series.



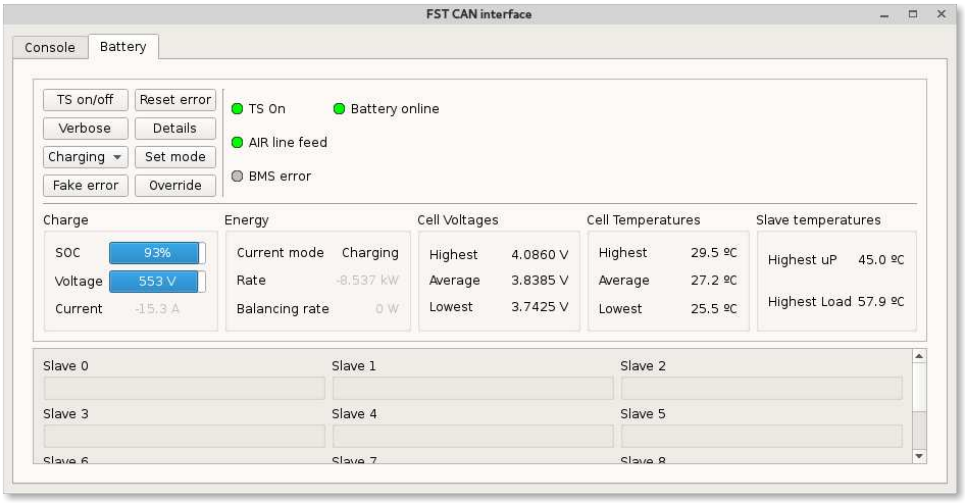
(a) Fully assembled battery without lid. Connector on the left of the middle section is low voltage only. Connector to the right is for high voltage and current.



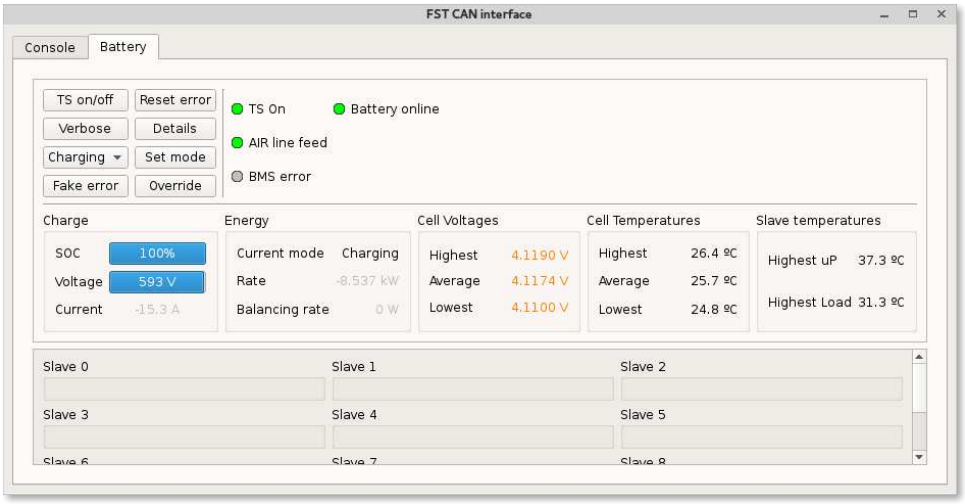
(b) Charging test. Special connector emulates a number of the car's systems. LED (bottom left) indicates high voltage.

Figure 7.4: *FST-05e* battery.

Figure 7.5: Cell voltages in a final stage of balancing. Green ‘lights’ indicate good connections to cells and sensors.



(a) Battery overall state a few minutes after starting a charging process. Cells are unbalanced and slave temperatures are high.



(b) Battery overall state just before charging cutoff. Total voltage is not 600 V since it's software limited to 593.28 V (4.12 V per cell).

Figure 7.6: Battery interface showing battery under charging conditions.

meant the slaves would be barely balancing the cells at all, contributing to lower temperatures by the end of the charging process as well as to improving the charging rate.

Figure 7.6 shows the overall battery state in two stages of the charging / balancing process. This charge cycle began with the battery almost charged already, but clearly unbalanced. It's possible to see how, by the end of the process, the slave temperatures are very low.

7.2.2 Discharging

Discharge tests within the car were mostly made within IST. Through successive tests, the system proved to be increasingly stable and able to protect the cells. Finally, in figure 7.7 it's seen a photo of *FST-05e* during the endurance event in Silverstone.

Unfortunately, late assembly issues in the high current path severely compromised the performance of the car during the competition. Indeed, the prototype was unable to finish any dynamic events for a variety

of mechanical and electrical reasons.

After the diagnosis of the problem, some cells were also found to be near their end of life, probably because these were still the same cells from a year ago which were never properly charged or handled given the limitations of the BMS system. All these factors contributed to a poor performance in *FSUK2014*.

In August, the battery was rebuilt with completely new cells in a configuration 144s1p given there were limited financial resources to invest in the now aging prototype. While this results in half the capacity, there is now extra space within the container that allowed to solve some of the assemblage problems of the previous solution.

Indeed, the battery is now proving to be more consistent and is expected to allow extensive tests to *FST-05e*, invaluable for the design of new prototypes.

Figure 7.7: Manuel Ferreira driving *FST-05e* in endurance event, Silverstone, *FSUK2014*. [Courtesy of Projecto FST]

7.3 Cost analysis

The budget for the proposed system comprises 20 slave modules v3.1, as implemented in *FST-05e*, and 4 master modules v4.0. The master's cost is estimated to a certain degree since it hasn't been prototyped yet, but components were chosen and the cost of the PCB calculated correctly for the chosen manufacturer. For BMS systems developed in-house, FS team from Zürich also reports a similar price — see table 2.4

These costs don't include the soldering of the components nor the manufacture of the cable harnesses. It also doesn't reflect the costs of the solutions being studied for *FST-06e*. These should have significantly higher cost per slave module since the PCBs have non-standard features. Correct figures are not available at this time.

With these considerations, the proposed system may be implemented in a FS prototype with several spare parts for 1464.28 €. This is already significantly lower than any commercial solution and not significantly more expensive than the solution from *FST-04e*. However, actual costs may be significantly reduced with larger order quantities. Indeed, series production would make this prototype significantly cheaper.

Figure 7.8 summarizes several other interesting estimates. While no attempt was made at providing an

estimate of labor costs, it's certainly impossible that this project cost that much, even though the development time is only estimated to be roughly twice as long. However, it's important to notice that the proposed system is not ready for a production environment and its installation requires some knowledge of the implementation and even reprogramming of the modules. Usually, commercial offers are ready for installation and only require high level configurations.

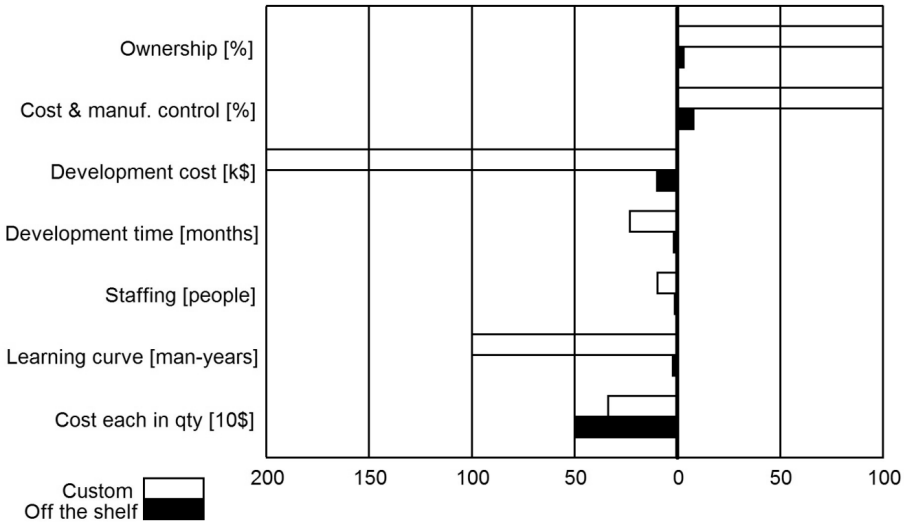


Figure 7.8: Comparison of commercial (off the shelf) and custom BMSs. [Andrea, 2010]

This is in fact possible to achieve, but something that would likely increase the costs. Considering the amount of development for accurate SoC estimation among other features and test costs for market introduction, this cost is much more understandable. More so, the development time, when put into perspective with the learning curve, is also significantly longer. Given the progress made throughout this work, this seems to be plausible.

Finally, the ownership and cost/manufacture control are consistent with the initial premise of this work and the motivation for a custom BMS.

Chapter 8

Conclusions

This thesis addresses the problem of battery management within a high performance vehicle with a partially distributed solution. This choice bears the intention of providing a scalable, yet cost effective solution with easier maintenance and installation.

While the final solution is not at feature parity with commercial solutions just yet, it has clearly all the capabilities to do so. Most of the objectives were fully addressed, providing a stable, safe and accurate system. Major exceptions were the installation problems and the lack of a fully fledged charger module and advanced estimation algorithms for parameters like SoC, DoD and SoH.

On one hand, these installation issues are highly dependent in a new battery design and not only in a new BMS architecture. On the other hand, both the charger and better performance parameters had to be sacrificed given the large scope of the project. However, in a fast paced environment like FS, priority was rightfully given to the actual testing, safety and stability of the car.

More so, better tools were developed to provide an expandable interface, intuitive controls and monitoring of the BMS subsystems, being indispensable to the achievements of this work. These tools also enable faster development and integration of new features in the BMS, but also of new modules within the specifications of Projecto FST.

In hindsight however, one option seems to be detrimental to further developments. Given the rapidly increase of the software complexity within the master module, it could benefit a lot from a RTOS. For the scope of this work, choosing against a RTOS solution allowed a faster development of the initial feature set. However, on a long term plan, a RTOS may prove to scale better and also improve the migration process to more powerful platforms if needed — e.g. a PIC32. Moreover, a large part of the code should be easily adapted to an RTOS, avoiding a complete reimplementation.

On a competition perspective, the recovery of *FST-05e* has been a difficult process with a myriad of mechanical and electrical problems, mainly EMI related. This prevented a better performance in *FSUK2014*, but each failure proved to be an invaluable lesson, also reflected in the proposed system.

The analysis of *FST-05e* faults also allowed the identification of potential performance degrading attributes. And while the removal of wires between slaves and cells may not always be possible depending on the design, it proved to be an important goal, if not a priority, in a new design.

Finally, tests proved quality assurance and good mechanical design to be the main priorities in a battery design as no BMS system is able to replace these. It can help however.

8.1 Future work

Other than specific work relative to *FST-06e*, there are several improvements to be made to the presented modules as well as the creation of others. Below, some future objectives are proposed, in no particular order, for improving functionality and performance of the system.

Regarding the master module, the full prototyping of v4.0 should be an immediate target to improve and expand the features available for *FST-06e* and future cars. These include the full integration of the RTCC module and current sensor for instance, whose implementation is only partially at this time. The added safety features are also a priority that may save a significant amount of financial and time resources later on.

The master's software would also benefit greatly from the integration of FreeRTOS, for instance. After some design assumptions were invalidated, the code for this module became increasingly harder to expand without breaking peripheral features. This solution may be implemented as an independent branch of the source code, with no drawbacks for the current code base. The timely preparation of this alternative could make this new revision ready for testing and deployment for the seventh prototype from Projecto FST.

Regarding the save modules, proper revisions should be made to integrate the newer *LTC6804*. This IC should provide significantly better and mainly faster measurements, among other improvements — *LTC6803* takes 13 ms opposed to 290 µs to read all cell voltages. This extra performance could yield a lower power requirement overall and it could be used for better diagnosis and self checks that are currently very costly.

The formal analysis of the code of the slave and master modules is also important to identify possible bottlenecks for scalability of the system, for instance. The instrumentation of the code with performance counters should detect critical code paths and misjudgments of software architecture, again providing another improvement vector.

SoC estimation is still an open issue for modern BMSs and has no definite answer. However, Coulomb counting is definitely an easy alternative once a current sensor is installed, and a step on the right direction. More so, extensive logging of parameters from *FST-05e* and *FST-06e* may provide invaluable data for offline analysis of several solutions. DoD and SoH are two other parameters that may studied and implemented in this way.

For extensive tests of both, the modules and the battery as a complete system, appropriate test benches should be developed or acquired. With simple and cheap solutions, it's possible to create (partially) automatized tests for the slave modules, for instance, with full control over them. Although more expensive, a bank of resistors and/or bench with test motors would allow the test of the battery as a whole, even before the construction of the respective vehicle.

Not any less important, the *FST CAN interface* and related tools should keep being expanded and improved. The proposed work suggests a possible architecture to complement these tools with extra features, and some of the basic infrastructure was already created. However, other features like proper abstraction of configurations (e.g. for different cars) is still not part of the design. Providing better, constantly evolving and car independent tools should be a target for Projecto FST in order to avoid the same issues faced along this work.

Finally, although expensive and dangerous, controlled destruction of cells may also provide invaluable information for BMS and container design improvements. This requires extra cells and even slave modules in a controlled environment where fires and explosions may be contained.

Among the relevant tests, material selection seems, at this time, to be the most relevant one, but the reactions of the BMS to such critical failure may also be relevant. Determining, for instance, if a fire causes a relatable fault — i.e. critical temperature — is one of several possible tests. Timely detection of these faults could greatly reduce risks for the driver and the car itself.

References

Abe, H., Murai, T. and Zaghib, K. [1999], 'Vapor-grown carbon fiber anode for cylindrical lithium ion rechargeable batteries', *Journal of Power Sources* **77**, 110–115.

Almeida, V. [2009], Volante Electrónico para o FST, Master's thesis, Instituto Superior Técnico.

Andrea, D. [2010], *Battery Management Systems for Large Lithium Ion Battery Packs*, 1st edn, Artech House.

Andrea, D. [2014], 'Comparison of BMS ICs for Large Li-ion Battery Packs'.
URL: *http://liionbms.com/xls/BMS_IC_table.xls [online, accessed 15-Aug-2014]*

ARINC [2014], 'ARINC 825'.
URL: *http://www.arinc-825.com/arinc825-standard [online, accessed 18-Aug-2014]*

Bergveld, H. J. [2001], Battery Management Systems Design by Modelling, PhD dissertation, Universiteit Twente.

ChaN [2014], 'Petit FatFS'.
URL: *http://elm-chan.org/fsw/ff/00index_p.html [online, accessed 10-Sept-2014]*

CHEVROLET [2014], 'CHEVROLET Volt specifications'.
URL: *http://www.chevrolet.com/volt-electric-car/specs/trims.html [online, accessed 9-Aug-2014]*

Codecà, F., Savaresi, S. and Manzoni, V. [2009], The mix estimation algorithm for battery state-of-charge estimator — analysis of the sensitivity to measurement errors., *in* 'Proceedings of 48th IEEE Conference on Decision and Control', pp. 8083–8088.

E Propulsion Systems [2012], *Data Sheet for EPS4500XP*.

Electropaedia [2014], 'State of Charge (SOC) Determination'.
URL: *http://www.mpoweruk.com/soc.htm [online, accessed 8-Sept-2014]*

elithion [2014], 'Lithiumate™ pro Product technical information'.
URL: *http://elithion.com/lithiumate__pro.php [online, accessed 23-Sept-2014]*

Figueiras, J. [2014], Formula Student Championship: an Integrated Data Processing Module Adapted for Pit-Stop Scenarios, Master's thesis, Instituto Superior Técnico.

Green Car Reports [2013], 'Tesla fires: what we know, and what we need to find out'.
URL: *http://www.greencarreports.com/news/1088281_tesla-fires-what-we-know-and-what-we-need-to-find-out [online, accessed 5-Aug-2014]*

Guedes, M. [2011], Battery Management System for Formula Student, Master's thesis, Instituto Superior Técnico.

LEM [2007], *Current Transducer HTFS 200..800-P/SP2*.

Linear Technology [2011], *LTC6803-2/LTC6803-4 Multicell Battery Stack Monitor*.

LION E-Mobility AG [2014], LION Smart Li-BMS v3 — Allgemeine Beschreibung des modularen Batterie-Management-Systems, Technical report, LION E-Mobility AG.

Microchip [2006], *dsPIC30F Family Reference Manual*.

Microchip [2010], *PICkit™ 3 Progammer/Debugger Users' Guide*.

Microchip [2013*a*], *Integrated Programming Environment (IPE) Users' Guide*.

Microchip [2013*b*], *MPASM™ Assembler, MPLINK™ Object Linker, MPLIB™ Object Librarian User's Guide*.

Microchip [2013*c*], *MPLAB XC16 C Compiler Users' Guide*.

Neundorf, A. [2014], 'CuteCom'.
URL: *http://cutecom.sourceforge.net/ [online, accessed 5-Oct-2014]*

Opel [2014], 'Opel Volt specifications'.
URL: *http://www.opel.de/fahrzeuge/modelle/personenwagen/ampera/index.html [online, accessed 9-Aug-2014]*

Piller, S., Perrin, M. and Jossen, A. [2001], 'Methods for state-of-charge determination and their applications', *Journal of Power Sources* **96**, 113–120.

Poly eRacing [2011], 'BMSafe'.
URL: *https://code.google.com/p/bmsafe/ [online, accessed 15-Mar-2014]*

Pop, V., Bergveld, H. J., Danilov, D., Regtien, P. P. and Notten, P. [2008], *Battery Management Systems — Accurate State-of-Charge Indication for Battery-Powered Applications*, Springer Science.

Projecto FST [2013*a*], Electrical Safety Form, Technical report, Projecto FST.

Projecto FST [2013*b*], FST-05e Design Reports, Technical report, Projecto FST.

Real Time Engineers Ltd. [2014], 'Free RTOS'.
URL: *http://www.freertos.org [online, accessed 8-Sept-2014]*

SAE [2014], '2014 Formula SAE Rules'.
URL: *http://www.fsaeonline.com/page.aspx?pageid=e179e647-cb8c-4ab0-860c-ec69aae080a3 [online, accessed 1-Sept-2014]*

Tesla Motors [2014], 'Tesla Model S specifications'.
URL: *http://www.teslamotors.com/models/specs [online, accessed 9-Aug-2014]*

The Bergquist Company [2011], *Gap Pad 1500R*.

Wikipedia [2014], 'Boeing 787 Dreamliner battery problems'.
URL: *https://en.wikipedia.org/wiki/Boeing_787_Dreamliner_battery_problems [online, accessed 5-Aug-2014]*

Appendix A

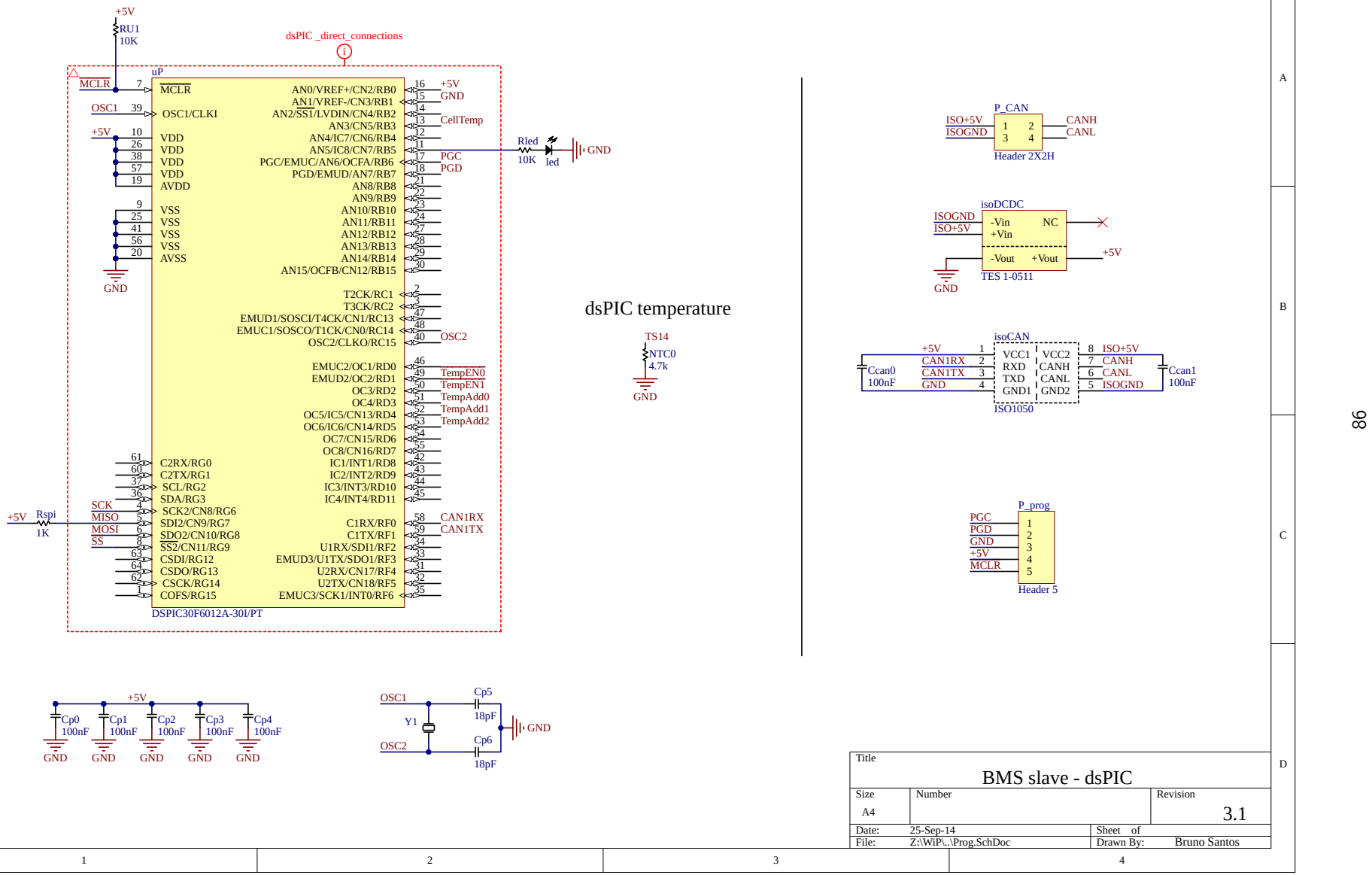
BMS Parameters

```
/*****************************************************************
 *   BMS general configurations
 *   ------------------------------------------------------------
 *   Bruno Santos
 *   2014 - Projecto FST Novabase
 *****************************************************************/

#ifndef __COMMON_BMS_CONFIG_H__
#define __COMMON_BMS_CONFIG_H__


/*
 * Limits and parameters
 */

#define CELLS_S_N       12  /* number of cells in series per slave; unlike next
    definitions, this one has implications in hardcoded functions within the
    slave module */
#define CELLS_P_N       2   /* number of cells in parallel per slave;
    implications on capacity calculations only */
#define TEMP_SENSORS_N  3   /* number of cell temperatures to monitor
    sequentially from the highest index; possible values: 1-12 */

/* voltages are in mV*10 */
#define V_CRITICAL_H    41200   /* critical high voltage */
#define V_ALERT_H       40900   /* alert high voltage */
#define V_ALERT_L       32000   /* alert low voltage */
#define V_CRITICAL_L    31500   /* critical low voltage */

/* temperatures are in ºC*10 */
#define TCD_CRITICAL_H  600     /* critical high cell temperature @ discharging
    */
#define TCD_ALERT_H     550     /* alert high cell temperature @ discharging*/
#define TCD_ALERT_L     (-150)  /* alert low cell temperature @ discharging*/
#define TCD_CRITICAL_L  (-200)  /* critical low cell temperature @ discharging*/

#define TCC_CRITICAL_H  450     /* critical high cell temperature @ charging*/
#define TCC_ALERT_H     400     /* alert high cell temperature @ charging*/
```

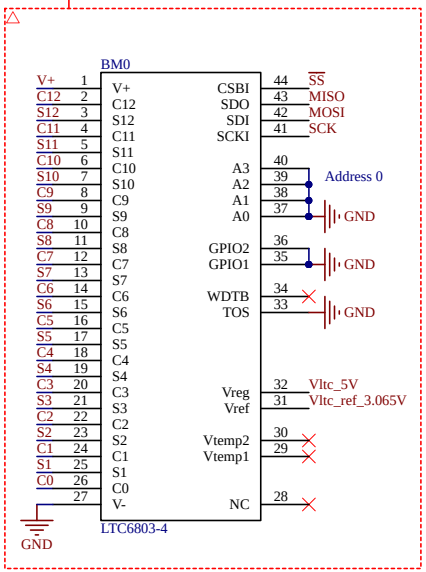
```
#define TCC_ALERT_L     50      /* alert low cell temperature @ charging*/
#define TCC_CRITICAL_L  0       /* critical low cell temperature @ charging*/

#define TS_CRITICAL_H   800     /* critical high slave temperature */
#define TS_ALERT_H      600     /* alert high slave temperature */
#define TS_ALERT_L      (-350)  /* alert low slave temperature */
#define TS_CRITICAL_L   (-400)  /* critical low slave temperature */

#define BALANCE_HYSTERESIS_P 100    /* precision hysteresis for balancing (10*mV
    ) */
#define BALANCE_HYSTERESIS_C 2000   /* coarse hysteresis for balancing (10*mV)
    */


/*
 * Modes of operation
 */

#define OP_INITIALIZING 0
#define OP_NORMAL       1
#define OP_REGENERATION 2
#define OP_CHARGING     3


/*
 * Status flags mask offset
 * These represent the position of the corresponding flag bits in a 16 bit mask
 */

#define MASK_TS_ON          0
#define MASK_AMS_ERROR      1
#define MASK_AIR_FEED       2

#endif
```

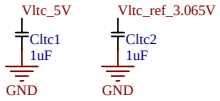
Appendix B

Slave module v3.1 schematics

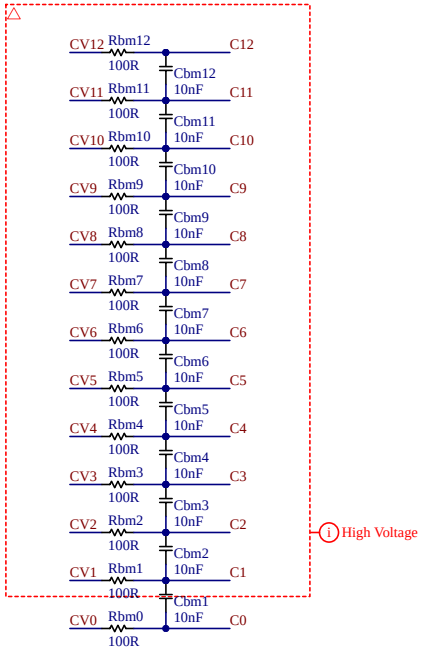

+5V
RU1
10K
dsPIC _direct_connections
uP
MCLR
OSC1
+5V
VDD
VDD
VDD
VDD
AVDD
VSS
VSS
VSS
VSS
AVSS
GND
AN0/VREF+/CN2/RB0
AN1/VREF-/CN3/RB1
AN2/SS1/LVDIN/CN4/RB2
AN3/CN5/RB3
AN4/IC7/CN6/RB4
AN5/IC8/CN7/RB5
PGC/EMUC/AN6/OCFA/RB6
PGD/EMUD/AN7/RB7
AN8/RB8
AN9/RB9
AN10/RB10
AN11/RB11
AN12/RB12
AN13/RB13
AN14/RB14
AN15/OCFB/CN12/RB15
T2CK/RC1
T3CK/RC2
EMUD1/SOSCI/T4CK/CN1/RC13
EMUC1/SOSCO/T1CK/CN0/RC14
OSC2/CLKO/RC15
EMUC2/OC1/RD0
EMUD2/OC2/RD1
OC3/RD2
OC4/RD3
OC5/IC5/CN13/RD4
OC6/IC6/CN14/RD5
OC7/CN15/RD6
OC8/CN16/RD7
IC1/INT1/RD8
IC2/INT2/RD9
IC3/INT3/RD10
IC4/INT4/RD11
C1RX/RF0
C1TX/RF1
U1RX/SDI1/RF2
EMUD3/U1TX/SDO1/RF3
U2RX/CN17/RF4
U2TX/CN18/RF5
EMUC3/SCK1/INT0/RF6
C2RX/RG0
C2TX/RG1
SCL/RG2
SDA/RG3
SCK2/CN8/RG6
SDI2/CN9/RG7
SDO2/CN10/RG8
SS2/CN11/RG9
CSDI/RG12
CSDO/RG13
CSCK/RG14
COFS/RG15
+5V
GND
CellTemp
PGC
PGD
OSC2
TempEN0
TempEN1
TempAdd0
TempAdd1
TempAdd2
CAN1RX
CAN1TX
RLed
10K
led
GND
DSPIC30F6012A-30I/PT
+5V
Rspi
1K
SCK
MISO
MOSI
SS
dsPIC temperature
TS14
NTC0
4.7k
GND
P_CAN
ISO+5V
ISOGND
CANH
CANL
Header 2X2H
isoDCDC
ISOGND
ISO+5V
-Vin
+Vin
NC
-Vout
+Vout
+5V
TES 1-0511
GND
isoCAN
+5V
CAN1RX
CAN1TX
GND
VCC1
RXD
TXD
GND1
VCC2
CANH
CANL
GND2
ISO+5V
CANH
CANL
ISOGND
Ccan0
100nF
Ccan1
100nF
ISO1050
P_prog
PGC
PGD
GND
+5V
MCLR
Header 5
+5V
Cp0
100nF
Cp1
100nF
Cp2
100nF
Cp3
100nF
Cp4
100nF
GND
OSC1
Y1
OSC2
Cp5
18pF
Cp6
18pF
GND
Title
BMS slave - dsPIC
Size
A4
Number
Revision
3.1
Date:
25-Sep-14
Sheet of
File:
Z:\WiP\..\Prog.SchDoc
Drawn By:
Bruno Santos

LTC _direct_connections

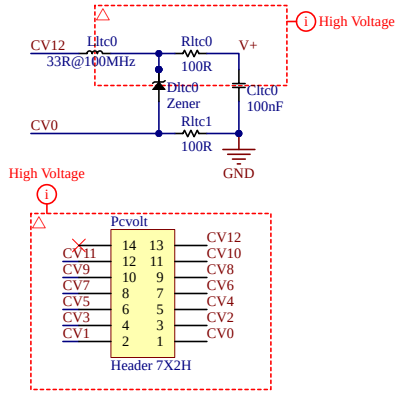




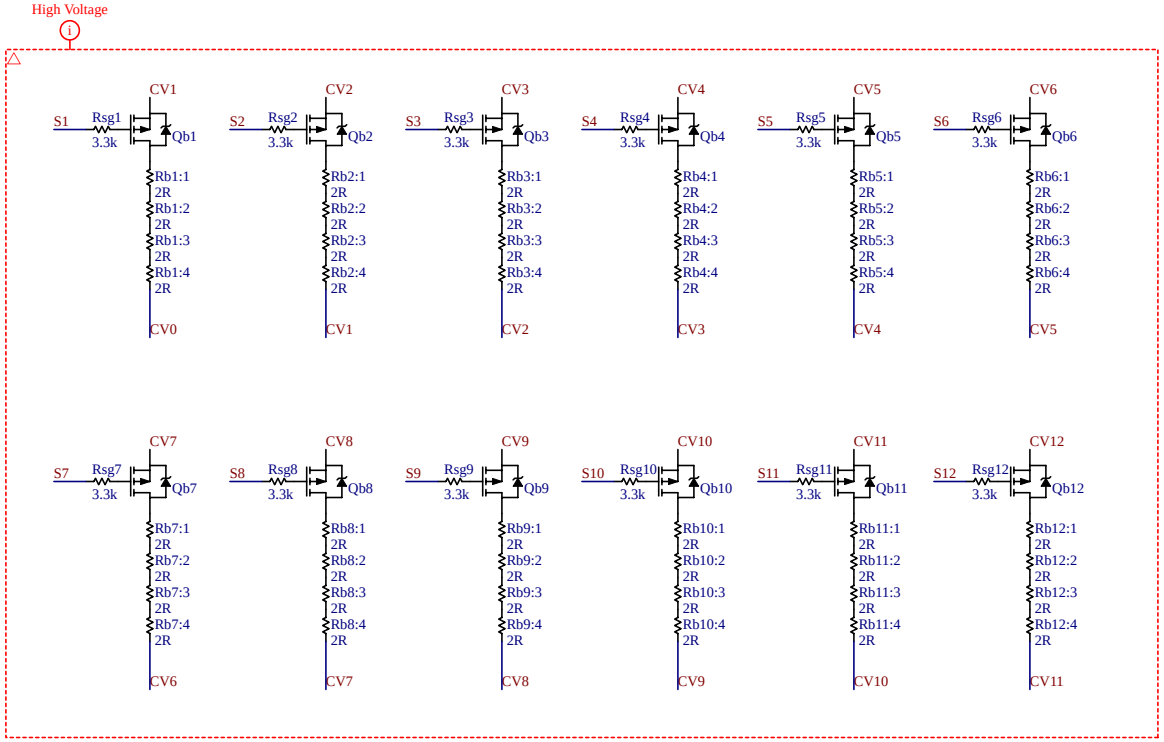
Cell voltage filtering



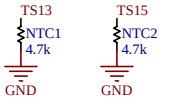
HV LTC's Supply & cell's connections



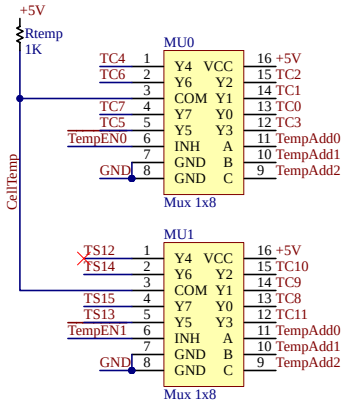
Title	BMS slave - LTC	
Size: A4	Number:	Revision: 3.1
Date: 25-Sep-14		Sheet of
File: Z:\WiP\..\Vsense.SchDoc		Drawn By: Bruno Santos



Balancing load temperature

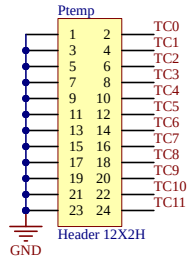


Title: BMS slave - Balancing		
Size: A4	Number:	Revision: 3.1
Date: 25-Sep-14		Sheet of
File: Z:\WiP\..\Vbalance.SchDoc		Drawn By: Bruno Santos



[T]emp [C]ell/[S]lave [#]address

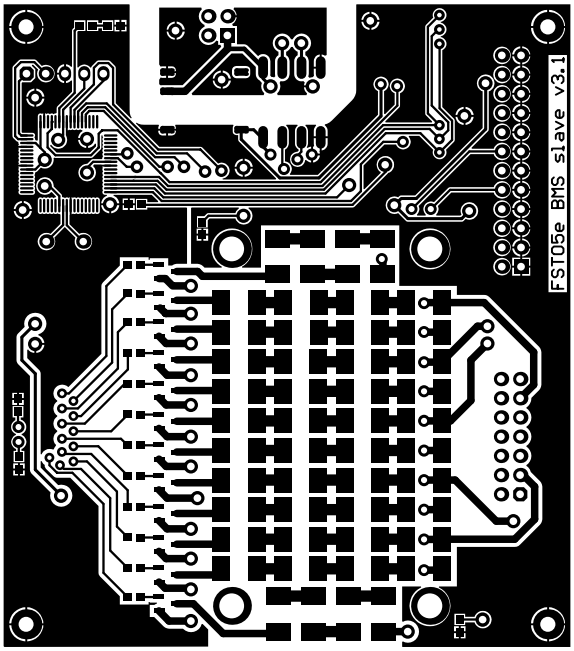
Temperature sensors' connector



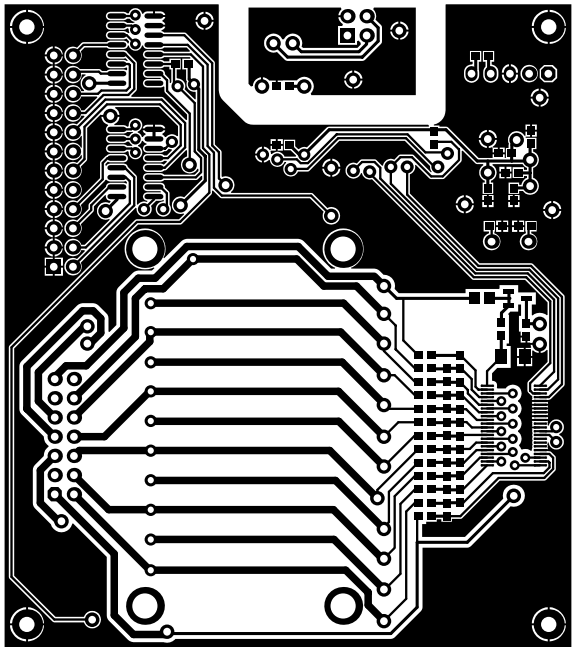
Title	BMS slave - cell temperatures	
Size: A4	Number:	Revision: 3.1
Date: 25-Sep-14		Sheet of
File: Z:\WiP\..\Tsense.SchDoc		Drawn By: Bruno Santos

Appendix C

Slave module v3.1 masks



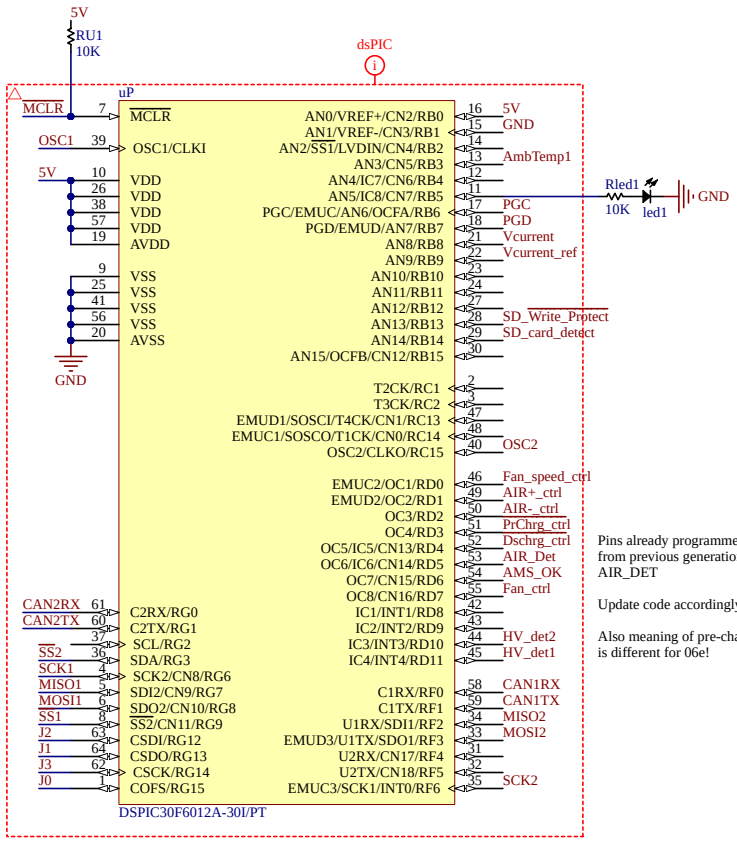
(a) Top layer 1:1.



(b) Bottom layer 1:1.

Appendix D

Master module v4.0 schematics

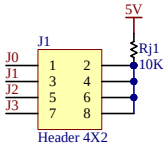


Pins already programmed that changed from previous generation:
AIR_DET

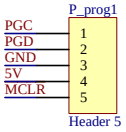
Update code accordingly!

Also meaning of pre-charge and discharge is different for 06e!

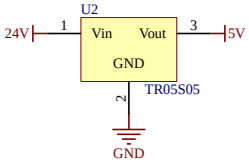
Configuration jumpers

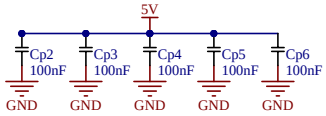


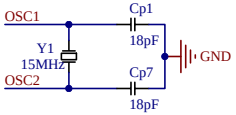
Programming



24V/5V converter

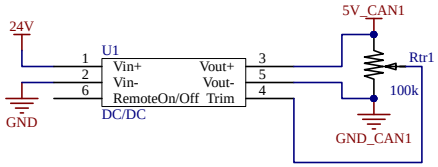




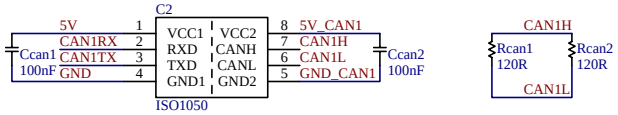


Title	BMS master - dsPIC	
Size: A4	Number:	Revision: 4.0
Date: 29-Sep-14	Sheet of	
File: Z:\WiP\..\Prog.SchDoc	Drawn By: Bruno Santos	

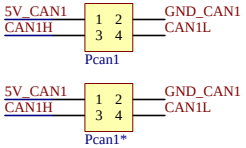
CAN 1



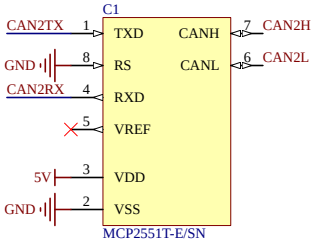
CAN 1 termination



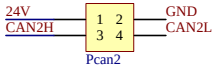
Slaves' supply and internal CAN



CAN 2

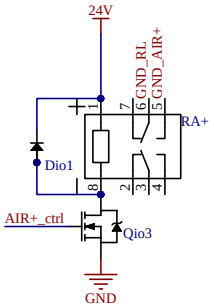


Master's supply and external CAN

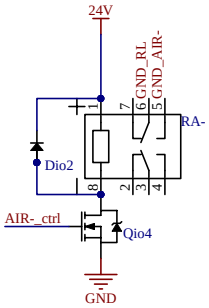


Title	BMS master - CAN	
Size: A4	Number:	Revision: 4.0
Date: 29-Sep-14	Sheet of	
File: Z:\WiP\..\CAN.SchDoc	Drawn By: Bruno Santos	

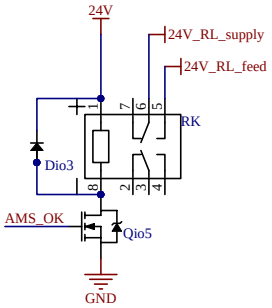
AIR+



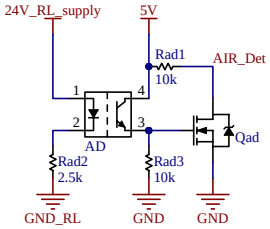
AIR-



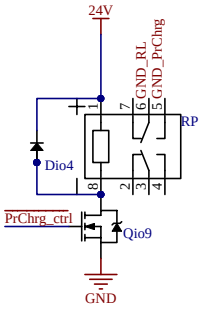
AMS AIR control



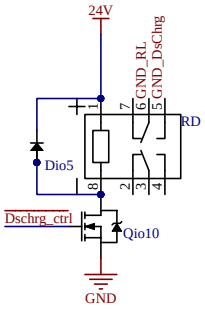
AIR supply detection



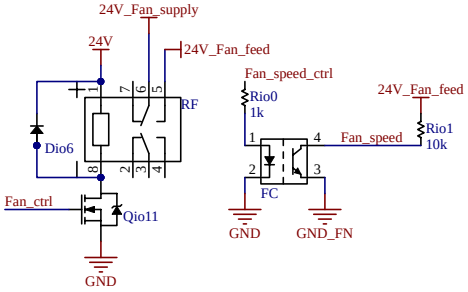
Pre-Charge



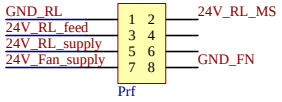
Discharge



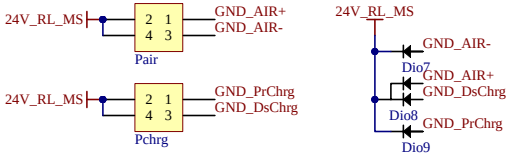
Fans

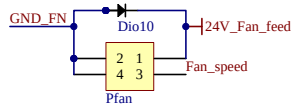


Supplies

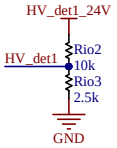


Device connectors

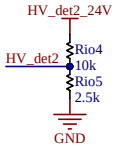


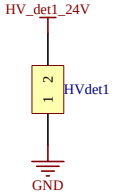


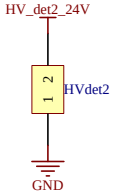
HV present 1



HV present 2







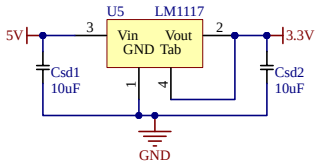
HV present signals should be generated by a TSAL like circuit (container LED signal).

This is intended to detect HV at battery terminals (TSAL may be used) and across pre-charge/discharge resistor (same circuit, diferent placement).
If pre-charge and discharge are two independent circuits, there should be a third HV_det.

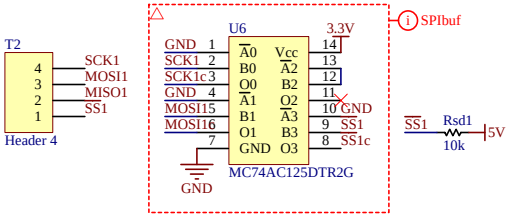
This is vital in diagonostics of pre-charge, discharge and AIRs.

Title: BMS master - I/O		
Size: A4	Number:	Revision: 4.0
Date: 29-Sep-14		Sheet of
File: Z:\WiP\..\IO.SchDoc		Drawn By: Bruno Santos

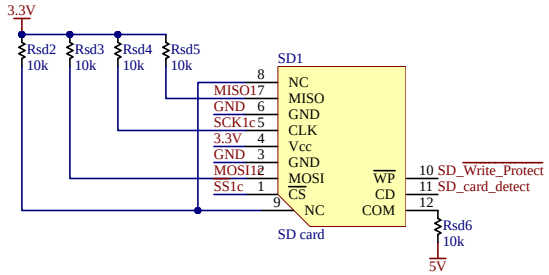
3.3V supply (shared with RTCC)



SPI bus

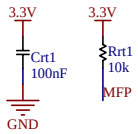


SD card holder

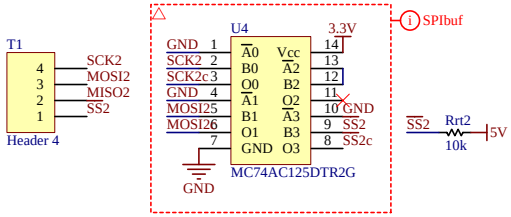


Title	BMS master - SD card	
Size: A4	Number:	Revision: 4.0
Date: 29-Sep-14	Sheet of	
File: Z:\WiP\..\SDcard.SchDoc	Drawn By: Bruno Santos	

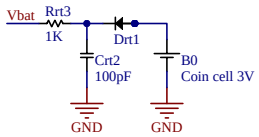
3.3V supply (shared with SD card)



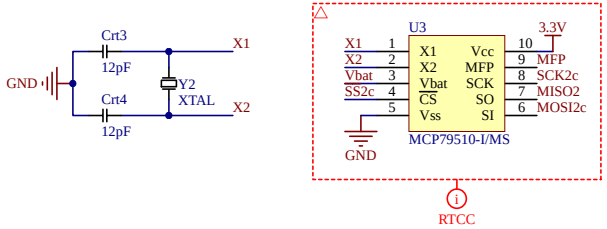
SPI



Backup power supply

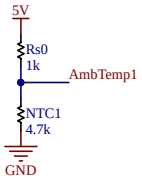


RTCC

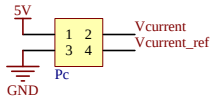


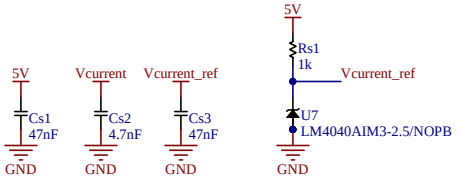
Title	BMS master - RTCC	
Size: A4	Number:	Revision: 4.0
Date: 29-Sep-14		Sheet of
File: Z:\WiP\..\RTCC.SchDoc		Drawn By: Bruno Santos

Ambient temperature



Connector for external current sensor

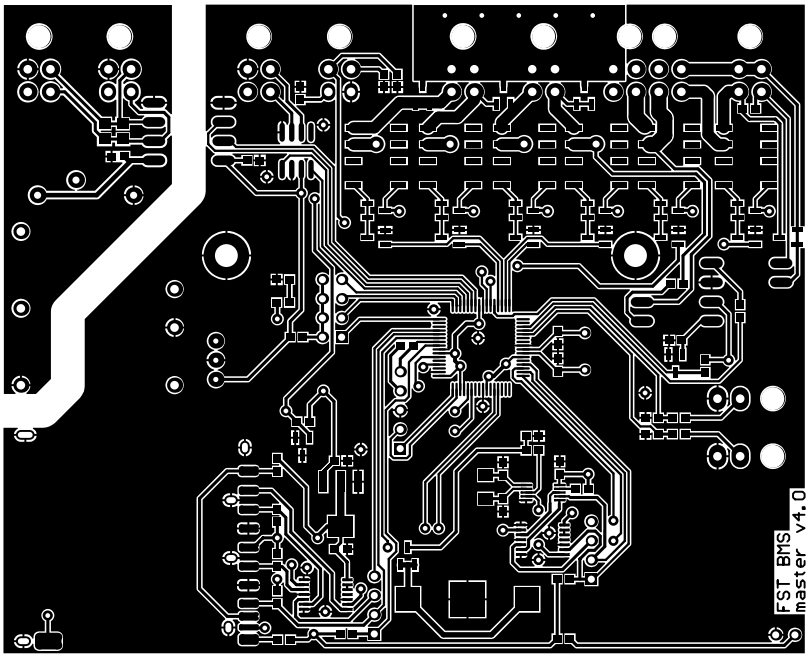




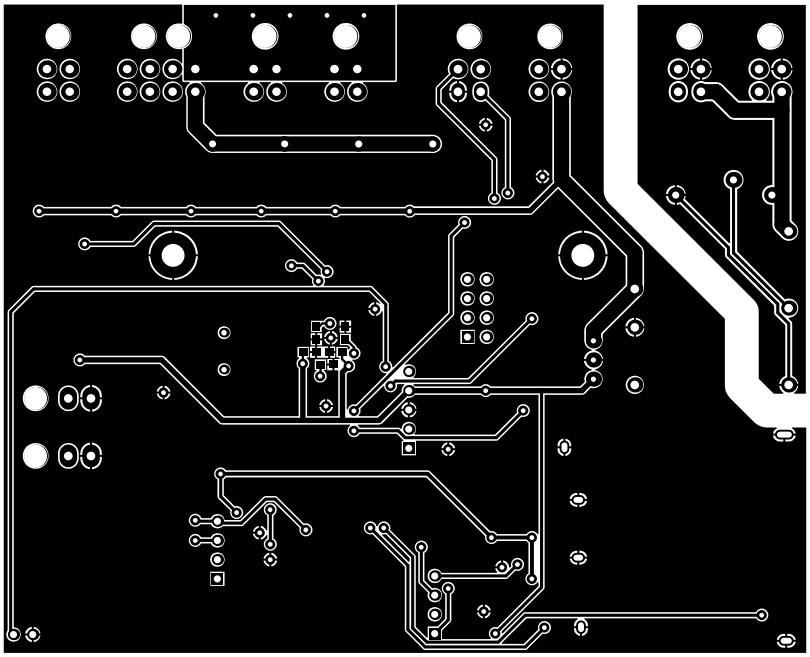
Title	BMS master - Sensors	
Size: A4	Number:	Revision: 4.0
Date: 29-Sep-14	Sheet of	
File: Z:\WiP\..\Sensors.SchDoc	Drawn By: Bruno Santos	

Appendix E

Master module v4.0 masks



(a) Top layer 1:1.



(b) Bottom layer 1:1.