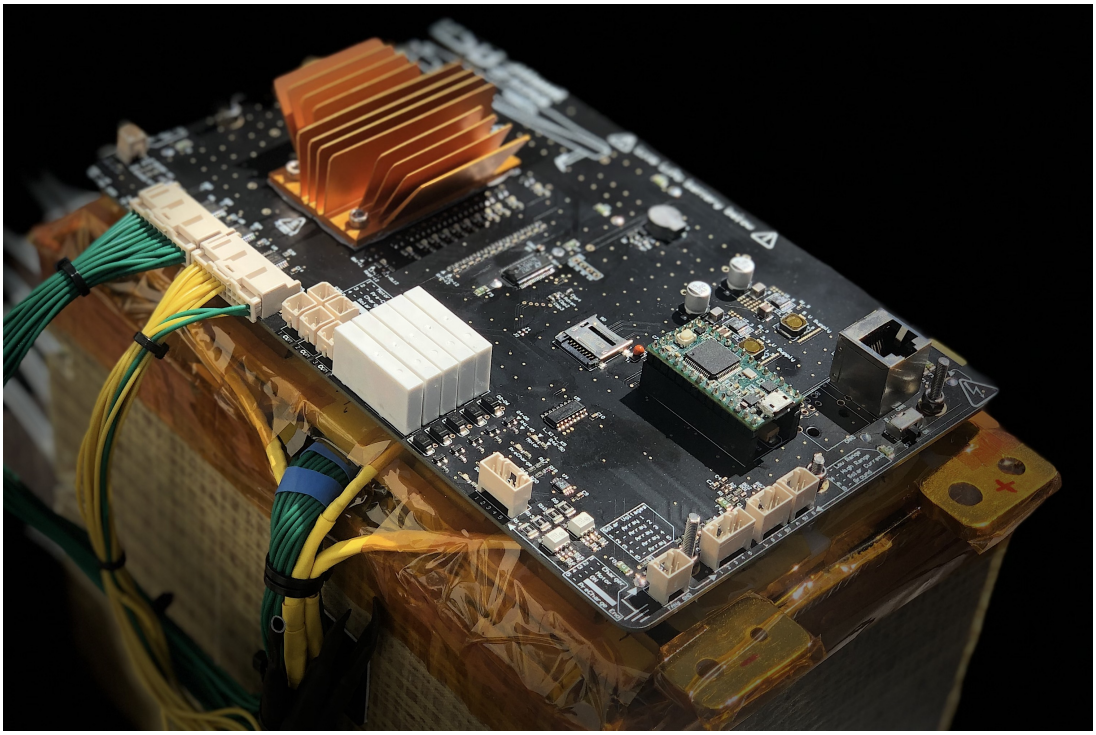
TÉCNICO LISBOA



BMSSPV - Battery Management System for Solar Powered Vehicles applied to Técnico Solar Boat prototype

Sebastião Maria Antelo de Holbeche Beirão

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisors: Prof. Rui Manuel Rodrigues Rocha
Prof. Carlos Manuel Ribeiro Almeida

Examination Committee

Chairperson: Prof. Teresa Maria Sá Ferreira Vazão Vasques
Supervisor: Prof. Rui Manuel Rodrigues Rocha
Member of the Committee: Prof. Moisés Simões Piedade

November 2021

Declaração

Declaro que o presente documento é um trabalho original da minha autoria e que cumpre todos os requisitos do Código de Conduta e Boas Práticas da Universidade de Lisboa.

Declaration

I declare that this document is an original work of my own authorship and that it fulfils all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Dedicated to the entire Técnico Solar Boat team

Acknowledgments

First and foremost, I would like to thank all of Técnico Solar Boat team. The work done in this thesis and the motivation to do it could not have been possible without your help. For the last five and a half years, I have had the opportunity and fortune to work with some of the most incredible people within Técnico Solar Boat. It was for sure a wonderful experience.

From the entire team there are a few names that I need to mention: Gilles Trincheiras, Basile Belime and Gonçalo Costa for their role as Head of the Electrical Systems Department, team management skills, and for always pushing us to innovate and improve our systems even further; João Pinto for setting the foundation for the TSB's Serial Protocol used in this work; João Cruz and David Mendes for their help in implementing the CAN Bus in our second solar prototype and in the case of David also for the development of the BMS GUI; João Martins, Vasco Figueiredo, Vasco Oliveira and Vítor Caires for their help during the assembly process of the BMS and the battery; José Pedro Figueiredo for succeeding me as the Head of Electrical Systems and all his work for the team; Robin Tomaz for leading the Mechanical Systems Department, developing incredible systems for the boat and above all piloting it; João Silva, Dinis Rodrigues and João Novo for leading this incredible team and allowing us to achieve so much in so little time. Furthermore, so that nobody is excluded, I would also like to thank every other member of the team who somehow had an important role in developing this work. I wish you all the best of success for the challenges to come.

Thank you also to everyone who has been part of this team for making this project such a great one.

I could not also fail to thank FST Lisboa for their help and suggestions during the development and testing of the first version, V0.

I would also like to express my gratitude to both my supervisors, Prof. Rui Rocha and Prof. Carlos Almeida, for helping and guiding me throughout this thesis and for their corrections and suggestions that for sure made this work a lot better.

Finally, a special thanks to my parents for all their lessons and support, for always pushing me forward throughout my student years, and for investing in my future. Another word goes to my grandfather, who studied Mechanical Engineering at IST, for his teachings and for always caring about me.

Abstract

Since 2015 Técnico Solar Boat (TSB), a student's project from Instituto Superior Técnico (IST) develops manned competition boats powered exclusively by green energies. The project started with solar-powered boats, but nowadays, hydrogen-powered boats are also developed. TSB's boats are designed to participate in international competitions that take place in Portugal, The Netherlands and Monaco. The main goal of this thesis is to design, build and test a Battery Management System (BMS) for TSB's solar-powered boats.

The boat on which the developed BMS will be used has a $44.4\,\mathrm{V}$ Lithium-Ion Polymer (LiPo) battery, $6\,\mathrm{m}^2$ of monocrystalline solar cells and a drive train composed by two BLDC motors. The low-power motor has a peak power of $5\,\mathrm{kW}$ while the high-power one has $10\,\mathrm{kW}$. This motor configuration allows the boat to be very efficient for longer races when only the low-power motor is used and at the same time have a high top speed when both motors are used simultaneously.

The system being developed is a centralized BMS that monitors the battery pack and sends the acquired information to the boat's Controller Area Network (CAN) Bus as well as to a Graphical User Interface (GUI) if a computer is connected. Besides monitoring the battery, the BMS is also responsible for controlling the different relays within the battery box as well as deciding whether the battery is safe and consequently if the boat can be used or not.

A set of requirements imposed by the competition's regulations and others imposed by the team have to be respected. Given the characteristics of the vessel in which the system will be implemented, the purposed BMS should be as reliable as possible so that the battery safety is never compromised. The system should also be lightweight and cost less than commercially available solutions.

Keywords: BMS, Battery Management System, Solar Powered Boat, Electrical Vehicle, Lithium-Ion Polymer Battery.

Resumo

Desde 2015 que o Técnico Solar Boat (TSB), um projeto de alunos do Instituto Superior Técnico (IST), desenvolve barcos de competição movidos exclusivamente a energias verdes. No início apenas eram desenvolvidos barcos solares, no entanto, hoje em dia, também já são desenvolvidos barcos movidos a hidrogénio. Os barcos do TSB são desenvolvidos para participar em competições internacionais que têm lugar em Portugal, nos Países Baixos e no Mónaco. O objetivo principal desta dissertação é projetar, contruir e testar um sistema de gestão da bateria para os barcos solares do TSB.

O barco onde o sistema de gestão da bateria irá ser usado tem uma bateria de polímero de lítio com $44.4\,\mathrm{V}$, $6\,\mathrm{m}^2$ de painéis solares monocristalinos e dois motores BLDC. O motor menos potente têm uma potência máxima de $5\,\mathrm{kW}$ enquanto que o mais potente tem $10\,\mathrm{kW}$. Esta configuração permite que o barco seja bastante eficiente em provas mais longas, onde apenas o motor de $5\,\mathrm{kW}$ é utilizado, mas ao mesmo tempo, tenha um velocidade máxima elevada quando ambos os motores são usados em simultâneo.

O sistema de gestão da bateria em desenvolvimento é um sistema centralizado que monitoriza a bateria e envia a informação recolhida para o barramento Controller Area Network (CAN) do barco bem como para a interface gráfica, caso um computador esteja conectado ao sistema. Além disso, o sistema é também responsável por controlar os diferentes relês que se encontram na caixa da bateria bem como decidir se a bateria está segura e consequentemente se o barco pode ser ou não usado.

Um conjunto de requisitos definidos pelo regulamento da competição assim como outros impostos pela equipa deverão ser respeitados. Dadas as características do barco no qual o sistema será implementado, é fundamental que o sistema proposto seja fiável, de forma a que a segurança da bateria nunca seja comprometia. O sistema deverá também ser leve e mais barato que as alternativas comercias disponíveis.

Palavras-chave: Sistema de Gestão de Bateria, Barco Solar, Veículo Elétrico, Bateria de Polímero de Lítio.

Contents

List of Figures

List of Tables

Acronyms

AC Alternate Current

ADC Analog-to-Digital Converter

BMS Battery Management System

BSM Battery Stack Monitor

BSS Battery Surveillance System

CAN Controller Area Network

CRC Cyclic Redundancy Check

DC Direct Current

DC-DC Direct Current to Direct Current

EEPROM Electrically-Erasable Programmable Read-Only Memory

EMF Electromagnetic Force

ESD Electrostatic Discharge

EV Electric powered Vehicle

GND Ground

GPIO General Purpose Input Output

GUI Graphical User Interface

HW Hardware

IC Integrated Circuit

ID Identifier

ISR Interruption Service Routine

IST Instituto Superior Técnico

JSON JavaScript Object Notation

LDO Low Dropout

LED Light Emitting Device

Li-ion Lithium-ion

LiPo Lithium-Ion Polymer

LT Linear Technology

MCU Microcontroller Unit

MOSFET Metal Oxide Semiconductor Field Effect Transistor

MPPT Maximum Power Point Tracker

NTC Negative Temperature Coefficient

OV Over-Voltage

PCB Printed Circuit Board

PWM Pulse Width Modulation

RC Resistor-Capacitor

RTC Real Time Clock

SOA Safe Operating Area

SoC State of Charge

SPI Serial Peripheral Interface

SR São Rafael

SW Software

TSB Técnico Solar Boat

USB Universal Serial Bus

UV Under-Voltage

WDT Watchdog Timer

1

Introduction

Introduction

In the past few years, not only due to environmental concerns, but also due to the instability in fuel prices, we have witnessed an increasing demand for Electric powered Vehicles (EVs). This demand is not only seen in terrestrial applications, such as cars, buses and even trucks, but also in the maritime and, more recently, aviation industries. Companies such as Silent-Yachts, Sunreef Eco and the Portuguese owned Sun Concept are already manufacturing and selling electric solar-powered boats, and many more are expected to join in the near future [1].

Regardless of the type of vehicle used, all of them share the same typical architecture composed by an energy storing unit, at least one electric motor and a power converter which converts the stored Direct Current (DC) energy in Alternate Current (AC) to power the electric motors[1]. This work will be focused mainly on the first one, the energy-storing unit, and more specifically on its BMS.

BMSs are real-time systems that control many functions vital to the safety and reliability operation of energy storing units in EVs. This includes tasks such as: monitoring temperature, individual cell voltage, current, battery health and State of Charge (SoC) and handling the communication with other system components. This system should be capable of identifying any potentially dangerous situation, such as over-voltage, and work in order to stop this faulty condition. Aside from EVs, BMSs are also commonly used in other battery applications such as uninterrupted power supply, off-grid power systems and maritime applications.

This thesis is done in partnership with TSB, a student's project from IST whose goal is to design, build and develop boats powered by sustainable energies, namely solar and hydrogen. This thesis will be focused on the development of a custom BMS for the solar-powered prototypes, São Rafael (SR). These prototypes are designed to compete in international competitions in Portugal, Monaco and The Netherlands. During these events, the boat is tested on its efficiency (in a 3 to $4\,\mathrm{h}$ endurance race, in which the boat with the most completed laps wins), its speed (in one-to-one races), and its manoeuvrability (in the slalom race).

Given that the competition's regulations limit both the battery capacity and the area of the solar panels, the battery and its BMS have to be designed in such a way that all the energy available is

[1] AC motors are normally chosen over DC ones given their higher starting torque, precise speed control, lower shaft friction and maintenance, easier cooling and above all, higher torque-to-weight ratios, which makes them much smaller in size than a comparable DC motor [2].

used but, at the same time, the boat is able to finish the races. This is especially important during the endurance race, where we need to use all the available energy while making sure that the battery remains within its Safe Operating Area (SOA)[2]. For such a race, it is crucial to have an accurate estimate of how much energy there is left on the battery, i.e. the battery SoC.

Figure 1.1: TSB's SR-02 prototype, during the Odisseia TSB event in Lisbon, September 2020.

1.1 Motivation and objectives

When TSB started, in 2015, we tried to develop our own BMS. However, problems with its development, some lack of knowledge and the limited time until our first competition forced us to search for a commercial alternative that would meet, at least partially, our needs.

Although this solution was used for competing for two years, it was far from perfect and required additional hardware to make it comply with the competition's regulations. The main problems with this option were that it only had two inputs for temperature measuring and that there was no way to control the solar panels' relays. These relays are mandatory by the competition regulation, and they should be able to isolate the solar panels from the rest of the boat's electrical system [3]. Finally, we needed a way to send the BMS's data to the rest of the boat's systems.

Given that the chosen solution did not have all the features that we needed, we ended up develop-

[2]The SOA is defined as the temperature, voltage and current conditions over which the cell or battery can be expected to operate without self-damage or degradation.

ing a custom board with the functionalities that our BMS was lacking. This board was named Battery Surveillance System (BSS) and its main functionalities were:

- Reading up to 32 temperatures from different points in the battery pack;
- Controlling the solar panels' relays;
- Telling the BMS how to control the charger and load relays;
- Communicating with the BMS to retrieve its data.

Also, we added a micro-SD card to this board so that we could log all the data for future analysis. Since the Maximum Power Point Trackers (MPPTs)[3] were nearby the BSS, we used it to measure the voltage of our five solar arrays and the total current that came from them. Finally, this board was responsible for broadcasting the BMS's and BSS's data to the rest of the boat's systems, using a Universal Serial Bus (USB) serial protocol.

This solution proved to be functional and didn't cause any major problems to the boat's performance, except for the low reliability of the implemented serial protocol, but it was far from optimized. There was much room for improvement. Learning the problems we faced during the 2018 competitions, we wanted to increase the reliability of the boat's electrical system for the 2019 season. We completely redesigned and simplified this system. At the time, the complete BMS was composed of two separate boards, and the battery assembly was very disorganized due to all the cables. We wanted to simplify this. Besides these improvements, we also wanted to have more control over the battery and all the systems that were connected to the BMS. These were the main reasons that gave us the motivation to restart the development of our own BMS.

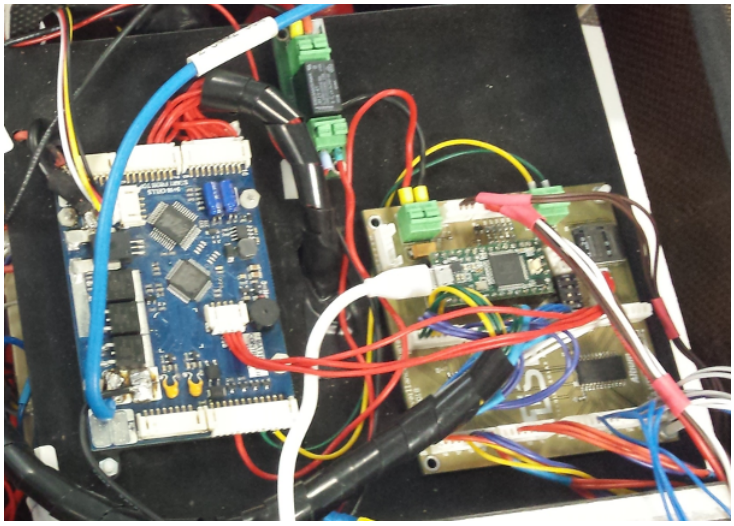


Figure 1.2: BMS (left) and BSS (right) of SR-01.

[3] A MPPT is an internally controlled Direct Current to Direct Current (DC-DC) converter with an algorithm to find and enforce a solar array voltage to its maximum power point.

The objective of this thesis is to develop an all-new BMS to replace the previously used system that was composed of a commercial BMS and our BSS. This new BMS should set a new standard in the project and be developed in such a way that allows future team members to easily make modifications to it, whether these are related to software or hardware. Hereto, special care must be taken to make sure it is well documented. Adopting a commercial solution is not an option since they rarely satisfy the team's needs, and when they do so, they are too expensive, as we will see in Section 2.2. Besides, adopting a self-developed solution will create, within the team, the know-how needed to improve this critical boat system in future project iterations.

1.2 Thesis outline

This thesis is divided into six chapters. Chapter 1 introduces this work and presents the motivation and objectives for it.

Chapter 2 presents the main functionalities of a BMS, as well as the two technologies available. The different BMS topologies will be presented, together with a list of commercially available solutions, as well as the reasons for not using those.

Chapter 3 starts by describing the battery design followed by the requirements to which the developed BMS should be constrained. The general architecture of the proposed system will also be presented in this chapter.

Chapter 4 characterizes the implementation of the proposed BMS. Aspects like the Hardware (HW) choices are all detailed here. The three versions developed are explained in detail. To conclude, a section regarding the Software (SW) is presented in which the most important SW features are detailed.

Chapter 5 outlines the deployment process of the proposed BMS, as well as some setbacks during this process. Tests performed to evaluate the performance and validate that the requirements are successfully met are also described in this chapter.

Finally, Chapter 6 compares the achieved system with the objectives that were set at the beginning, detailing all the problems and achievements experienced throughout this master thesis. A final section with some future work is also described in this chapter.

2

State of the Art

State of the Art

In the early ages of lithium battery technology development, BMSs were analog. This meant that hardware modification had to be made to modify parameters such as the cell's maximum or minimum voltage. With the technology improvements in Lithium-ion (Li-ion) cells throughout the years, the demand for more complex and feature-rich BMSs increased, and with it, the shift towards digital BMSs started. Figure 2.1 summarizes the main functions that most generic digital BMSs have nowadays. The heart of such BMSs is the Battery Stack Monitor (BSM) Integrated Circuit (IC), which is responsible for at least measuring the voltages of each cell. Some BSMs also include features such as over and under-voltage monitoring, some kind of balancing control, as well as General Purpose Input Outputs (GPIOs) ports that can be used either as digital inputs or outputs or as analog inputs. Another part is responsible for temperature and current measuring, as well as electronically controlled switches to control the load and charger. Finally, there is also a Microcontroller Unit (MCU) which communicates with the BSM, acquires its data and broadcasts all the BMS data to other systems. The MCU is where the algorithm which is responsible for making sure that the battery pack is safe runs.

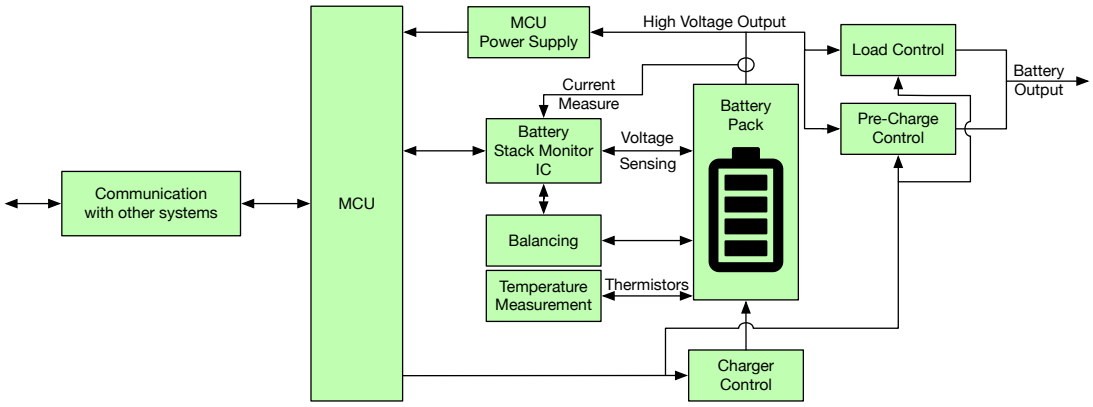


Figure 2.1: Generic BMS functional block diagram.

Nowadays, the majority of the BMSs available on the market are digital since they allow an easier (re)configuration of the critical parameters (voltage, temperature and current limits) while allowing the acquisition and transmission of valuable data, such as battery SoC, current, voltage and temperatures

to other interfaces and systems. Additionally, in case there is any problem with the battery, most digital BMSs are capable of not only reporting this faulty condition but also identifying exactly where the fault happened [4]. Another important task of this type of BMS is to report how much energy is left in the battery, which is very important in TSB's application because it allows the team to use all the available energy during the competition.

It is worth mentioning that even though digital BMSs are far superior to analog ones, the latter are still used in many standalone devices, such as single-cell chargers or standalone cells like the ones used in smartphones. Usually, these analog devices are designed for specific cell chemistries. LiPo cells must be connected to a BMS at all times, not only for charging, but also to ensure that they are operating according to the manufacture's SOA.

2.1 BMS topologies

According to [5], there are four different BMS topologies: centralized, modular, master-slave and distributed. There is no clear choice regarding which topology should be used for a given application. It depends a lot on its specific needs: safety, cost (parts, assembly and maintenance), and reliability are determinating factors.

Except for the centralized topology, which is primarily used in low voltage systems, with few cells in series (typically not more than 15), all the others are more frequently used for higher voltage battery packs, such as the ones used in EVs.

2.1.1 Centralized

A centralized BMS is implemented on a single board which is responsible for all the BMS tasks - see Figure 2.2. A bundle of wires connects the board to the cells ($N + 1$ wires for N cells in series). This approach has several advantages.

- It is the least expensive approach, for a low voltage battery;
- It is compact;
- For small battery assemblies, in case of troubleshooting or when a repair is required, it is easier to replace a single board given that on the other architectures, the faulty module needs to be identified beforehand.

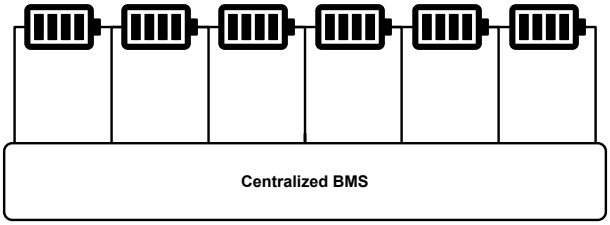


Figure 2.2: Centralized BMS.

An example of such a system is the Tiny BMS by Energus [6].

2.1.2 Modular

A modular BMS is similar to a centralized one, except that the BMS is divided into multiple, identical modules, each with its bundle of wires going to one of the batteries in the pack - see Figure 2.3. Typically, one of the modules is designated as a master, as it is the one that manages the entire pack and communicates with the rest of the systems. In contrast, the other modules act as simple remote measuring devices. A communication link transfers the readings from the other modules to the master module.

The modular topology compared to the centralized one adds the following advantages:

- The cable management is easier since each module can be placed closer to the cells it handles;
- Expansion to larger packs is straightforward: more modules are added.

However,

- The cost is slightly higher than the centralized topology since remote measuring modules have duplicated, unused functions;
- A few extra wires are required since some points need to be connected to two boards;
- The complexity is also slightly increased given that the slaves must have a way to communicate with the master module.

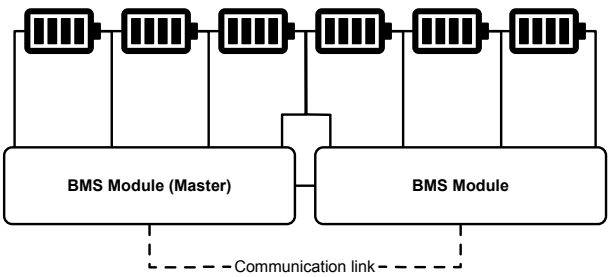


Figure 2.3: Modular BMS.

An example of such topology is the Lion Smart Modular BMS [7]. There is even a prototype version of this system that uses infrared to eliminate the communication wires. It has direct Printed Circuit Board (PCB) connection to the cells to eliminate tap wires [8]. This concept was demonstrated on a BMW i3 [9].

2.1.3 Master-slave

A master-slave BMS (Figure 2.4) is similar to a modular system in the sense that it uses multiple identical modules (the slaves), each measuring the voltage of a few cells. These slaves have a communication link to a master board, which is different from the slave boards and doesn't measure the cells' voltage directly. It only handles computations and communications.

The master-slave topology shares the same advantages and disadvantages as the modular topology. In addition, the extra wire problem is eliminated. The cost of each slave tends to be less than for the modular topology, as it is optimized just for the one job of measuring cell voltages, which reduces the overall system cost.

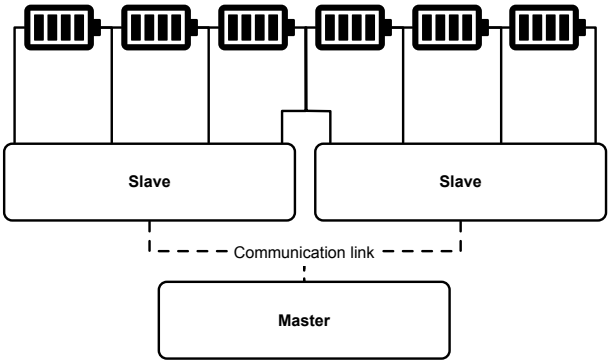


Figure 2.4: Master-Slave BMS.

An example of this approach is the REC BMS Master 9M [10].

2.1.4 Distributed

The distributed topology (Figure 2.5) is significantly different from the others. In a distributed BMS, the electronics are contained on cell boards that are placed directly on the measured cells. These boards usually measure cell voltage, temperature and take care of the balancing. This approach reduces the number of wires between cells and electronics since it only uses a few communication wires between the cell boards and a BMS controller. There are even some solutions that use wireless links to achieve this communication, thus completely eliminating the wires [11]. The BMS controller handles computations and communications.

This kind of approach has significant advantages but also disadvantages when compared to the others. The higher cost and the fact that communication cables are exposed to electrical noise, which may cause errors, are disadvantages. On the other hand, the better connection reliability, less expensive cost

of replacing parts, low noise and high-resolution voltage readings (since the board is directly connected to the cell's tabs), and expansion versatility are clear advantages. Adapting the number of cells only involves changing the number of cell boards, as long as the controller can handle the resulting amount of cell boards. For a more detailed list of pros and cons, see [11].

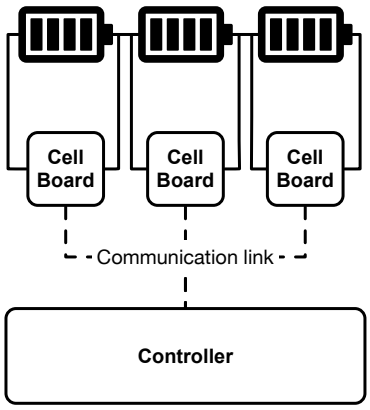


Figure 2.5: Distributed BMS.

An example of such topology is the Elithion Lithiumate HD BMS [12]. As it can be seen in Table 2.1, this is the most expensive solution.

2.2 Commercially available BMS

After reaching out to other teams to find out what they were using, we discovered that the vast majority of them were using a BMS and battery bundle produced by MG Energy Systems. After contacting the company, the price asked for such a system was absurdly high (4000 €), especially considering it was TSB's first year competing and that we had already bought a battery. Given that, we had to search for other alternatives.

Table 2.1 summarizes the different alternatives that were considered at the time and are still available as of today. For an alternative to be considered, it had to have the means of sending data to other devices and be able to handle 12 cells in series. The alternatives found had three different topologies: centralized, master-slave and distributed.

As shown in Table 2.1, the majority of the solutions found were centralized. This is primarily due to the small number of cells in series that the battery has (12). Normally distributed and master-slave solutions are only worth it for higher voltage battery packs where the number of cells in series is much higher, as mentioned earlier.

From the shown alternatives, there is one, the Roboteq's BMS1060A, that could be discarded right away given that its maximum current is $100\,\mathrm{A}$ which was below the expected maximum system current. The other options presented in Table 2.1 are all suitable for the system, and they basically differ on the

number of cells that can be monitored. While the centralized solutions can monitor up to 24 cells, the master-slave and the distributed solutions can monitor up to 255 cells. At the time, money was, and still is, a major concern in projects like TSB resulting in all solutions above 500 € not being considered, even though they may offer more features and remove the need for extra boards to account for the missing functionalities.

Taking this into account, we were left with the first two options. The alternative that we ended up using consisted of a Li-ion 12s14p[1] self-assembled battery pack composed of cylindrical 18650[2] cells in conjunction with the Tiny BMS from Energus Power Solutions [6]. Even though this solution was not perfect because it lacked sufficient thermistor inputs, a way to control two possible sources of power (AC/DC charger and solar panels), data-logging and a reliable way to send its data to other boat systems, it was the best solution that we could afford at the time, especially considering what it offers versus its cost.

In fact, if we would have had the knowledge that we have today and more available funds, choosing the Orion Jr. BMS would probably have been a better option, given that it can control all the relays that our current boat has. It also has a CAN bus, which means that it would probably be suitable for our current boat.

[1] XsYp stands for X cells in series and Y in parallel. Hereinafter this parallel will be referred to as cell aggregate.
[2] 18650 stands for a cylindrical cell with a diameter of $18\,\mathrm{mm}$ and a hight of $65\,\mathrm{mm}$.

Table 2.1: Comparison between most suitable commercial available solutions.

Model	Energus Tiny BMS 150A/750A	Orion Jr. BMS	Roboteq's BMS1060A	REC BMS Master 9M	Elithion Lithiumate HD BMS	Lithium Balance c-BMS
Type	Centralized	Centralized	Centralized	Master-Slave	Distributed	Centralized
Cells	4 - 16	4 - 16	11 - 15	1 - 225	1 - 255	4 - 24
Maximum Voltage [V]	75	60	80	60	900	100
Maximum Current [A]	750	1000	100	No Limit	600	2000
Balancing Current [A]	0.1	0.15	Undisclosed	1.3	0.2	0.2
Thermistor Inputs	2	2	3	8 per slave	1 per cell	6
Relay Outputs	2	4	None	4	3	4
Current Sensor	Hall Sensor	Shunt	Shunt	Shunt or Hall sensor	CAN or sensor	Shunt or Hall sensor
Communication	UART	CAN	CAN	RS-485, CAN	CAN, RS-232	CAN
Power Consumption [mW]	200	1100	800	390	1200	2700
BMS price for 12s battery	199 €	400 €	400 €	550 €	1 250 €	340 € [3]

[3]This price doesn't include an extra 1 250 € for the software needed to configure the BMS.

3

Architecture

Architecture

This chapter will cover the overall system architecture, mainly concerning the BMS itself, but to do so, we first need to specify the battery pack configuration. Although this configuration is essential for the BMS design, it is out of the scope of this work, and thus it will only be briefly summarized in Section 3.1. Other parts of the boat's electrical system that are considered to be relevant for designing the BMS will also be addressed.

Although the team can define most of the BMS requirements, there are a few constraints imposed by the competition regulations, like the obligation to provide a way to monitor individual cell voltages during the battery tests [13]. This requirement implies that the BMS must be digital since such requirement is impossible to satisfy with an analog BMS.

3.1 Battery design

Regarding the battery design, the only constraints imposed by the regulations besides the need to use commercially available cells are the maximum system voltage and the maximum energy stored onboard. The former should not be greater than $52\,\mathrm{V}$, which for Li-ion and LiPo cells leads to a maximum of 12 cells in series, while the latter is limited to $1.5\,\mathrm{kWh}$.

For the first year, the team's choice went for cylindrical 18650 Li-ion cells. These had an energy of $9.62\,\mathrm{Wh}$ each, and thus to have a battery close to the maximum energy allowed in the competition, we used a 12s14p configuration. The number of cells in series was given by the motor specifications, which needs a nominal voltage of $44.4\,\mathrm{V}$. This 12s14p configuration, shown in Figure 3.1 produces, in theory, a battery with slightly higher energy ($116\,\mathrm{Wh}$ above the limit). However, after performing discharge tests, we concluded that the battery had less than $1.5\,\mathrm{kWh}$, which already indicated that this assembly had some power losses.

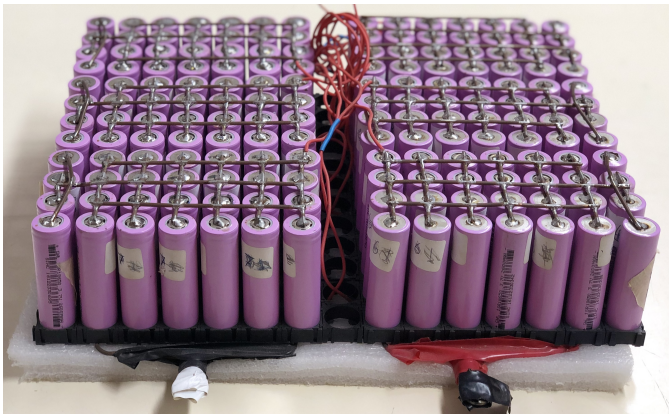


Figure 3.1: TSB's first battery pack.

This battery configuration proved unreliable, mainly due to the total number of cells and the way it was assembled. For our second year, we wanted to reduce the number of cells to have a more straightforward and reliable assembly. Moreover, for this year, the motor power was doubled (we went from $4\,\mathrm{kW}$ peak to $8\,\mathrm{kW}$ peak), which made the evolution towards the more power-dense LiPo cells happen naturally.

We ended up finding cells with a $3.7\,\mathrm{V}$ nominal voltage and capacity of $16.8\,\mathrm{Ah}$, which allowed us to move from a 12s14p battery, with 168 cells, to a much simpler 12s2p battery with only 24 cells, shown in Figure 3.2. This assembly has a theoretical nominal capacity of $1491.84\,\mathrm{Wh}$ which almost matches the competition maximum. This configuration was the closest match we were able to find. Tests carried out by the competition's technical inspection team showed that the actual capacity was slightly higher (around $1520\,\mathrm{Wh}$). Nonetheless, it was still within the allowed capacity, considering the measuring error.

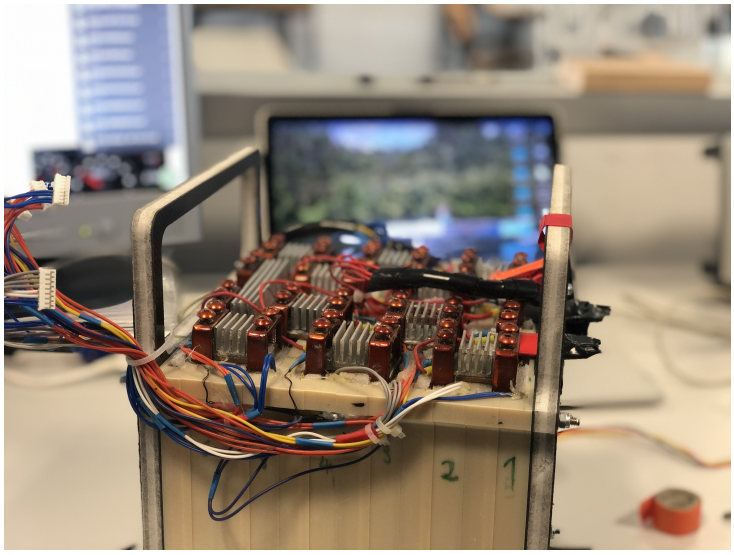


Figure 3.2: TSB's second battery pack.

This cell configuration is the one we still use today, with minor changes in what concerns the cell's C rating[1]. Nonetheless, the assembly still had another iteration. While in Figure 3.2 the cell's tabs were pressed against the bus bar, on subsequent batteries, we laser welded the cells' tabs to the bus bars,

as shown in Figure 3.3. This not only improves the connection reliability but also decreases the contact resistance and the assembly complexity. After using bare copper tabs on our third battery, for this fourth iteration, the bus bars were also nickel-plated in order to protect them against oxidation.

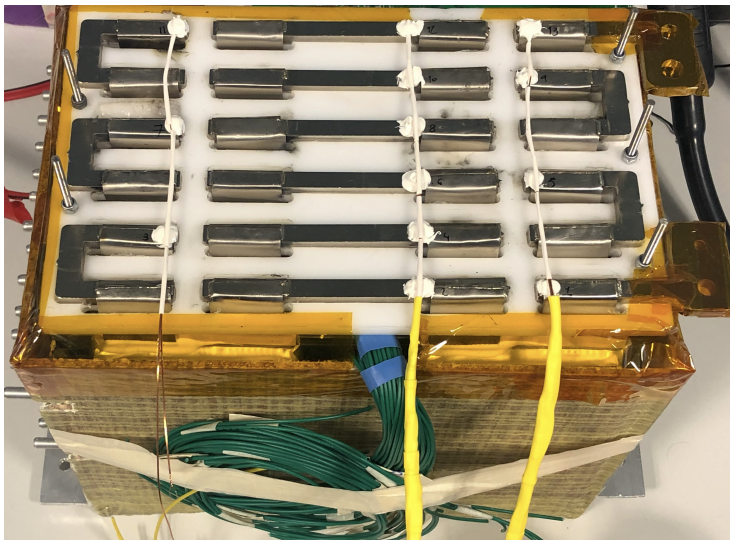


Figure 3.3: TSB's fourth and most recent battery pack.

Compared to the second and third battery packs designed, our fourth battery implements a new cooling system for the cells. Previously, the cells were only cooled on the tabs by forced air convection. Now ultra-fine water-cooling plates are added in between each cell. Doing so helps to maintain the battery temperature below $45\,^{\circ}\mathrm{C}$, which is the maximum allowed temperature for charging, ultimately improving the boat's performance.

3.2 BMS requirements

As already mentioned, the BMS must comply with the requirements imposed by the competition regulations, which for our class and according to [14] are:

- Monitor battery voltage;
- Monitor battery temperature;
- Be able to shut the high power system down when necessary;
- Monitor both charge and discharge currents;
- Control too high currents;
- Monitor the voltages from all the cell aggregates that compose the battery (12).

All the measurements of the above parameters should have a minimum refresh rate of $2\,\mathrm{Hz}$. The BMS must also be able to balance the battery since, according to [15], the battery must be balanced

[1] C rating is given by the division between the cell's maximum current (either charge or discharge) and its capacity.

when starting the capacity test. Moreover, this is a necessary task in order to use all of the battery's energy. Besides these requirements imposed by the regulations, the team also wants the following:

- High accuracy cell voltage measurements, with an error lower than $2\,\mathrm{mV}$;
- Measure at least 16 temperatures: one on each cell aggregate since they can't be charged if their temperature is above $45\,^{\circ}\mathrm{C}$; two on the balancing resistors heatsink, to make sure the resistors don't overheat while balancing; one to measure the ambient temperature inside the battery box; and finally, another one to measure the BMS main chip temperature. The measurement of these temperatures should have a refresh rate of at least $0.5\,\mathrm{Hz}$;
- Sense the battery voltage at three different points to know when the boat or the motors are turned on, or the charger is plugged in. The MCU should detect this with a pin-change interruption;
- Capability to control, at least, the motor, charger and solar relays so that the current flow to or from the battery can be stopped;
- Control the precharge of the motor controllers' capacitors;
- Calculate battery information (such as SoC and number of cycles) to give a more detailed view of the battery status to the team;
- Control the battery fans so that they can be shut down and thus save energy if they are not needed;
- Low-power consumption, and high standby life (at least years without draining the battery);
- Be capable of storing user-configurable parameters in non-volatile memory;
- Watchdog Timer (WDT) functionality.

Given that the MPPTs we currently use don't have a communication interface and that the BMS will be nearby them, we also want the BMS to be able to:

- Measure the voltage of the five solar arrays with a refresh rate of at least $0.5\,\mathrm{Hz}$;
- Measure the total current coming from the solar panels.

By having access to the aforementioned data, we can make sure that the MPPTs are doing their job (by checking the solar arrays voltages and comparing it to the open-circuit voltage) and calculate how much energy we can consume during the endurance race (by knowing how much power we are getting from the solar panels).

The proposed BMS should communicate with the rest of the boat's systems with a CAN bus, a protocol that is a standard across the vehicle and marine industries, known for its robustness, so much that nowadays it is also widely used in the aviation industry [16]. It is particularly suitable for noisy environments subject to electromagnetic interference like an electric-powered vehicle. The BMS should broadcast all the acquired data to the boat's CAN Bus so that it can be displayed to the pilot and logged on the boat's CAN logger. The bus should run at $1\,\mathrm{Mbit\,s^{-1}}$ given that this is the boat's CAN Bus baud rate.

Apart from the communication with other boat systems, the BMS should also be able to communicate with a computer that can be used to configure it (change parameters such as cut-off voltage and temperature limits, for example) and also to graphically display the BMS's data with the help of a GUI. For this task, an USB serial connection should be used.

Finally, the BMS should log relevant events to an onboard micro-SD card so that errors can be checked later, have a buzzer to signal any fault(s) and have a Real Time Clock (RTC) to keep track of time.

3.3 BMS architecture

Although all the topologies presented in Section 2.1 can be used for the proposed system, one of them must be chosen. The distributed topology is more expensive than the others, and given the relatively low number of cells in series, which is limited by the regulations that won't change until at least 2025, the advantage of adding more cells easily doesn't apply to this our case. We are left with the centralized, modular and master-slave topologies. The last two are very similar. The only difference is the fact that while on the modular topology all boards are equal, on the master-slave topology the master is physically different from the slaves, as seen in Section 2.1.

Considering that there are plenty of BSMs available on the market that support 12 cells in series, dividing the battery pack into groups to adopt either the modular or master-slave topology doesn't make much sense. Especially considering that the BSM is one of the most expensive components of the BMS and adopting the mentioned topologies would require more than one BSM which increases the overall system complexity and communication overhead. Moreover, the modular topology would require an extra MCU compared to both the centralized and the master-slave topologies, which can also be an expensive component. That's why, typically, these topologies are more often seen in systems made for larger battery packs with tens of cells in series.

Bearing in mind not only what was aforementioned, but also that our battery design is composed of a single stack, where cells are first connected in parallel and, only after that, in series in order to produce the most compact design and reduce the amount of measuring points, and the overall system complexity, the topology that makes the most sense is the centralized.

With both the requirements and the topology already defined, it is time to define the system's general architecture. The main task of the BMS is to measure the cells' voltage, the pack voltage, the pack current and the cells' temperatures. The voltage measurement task can be done in multiple different ways. However, the most common way of doing it is by using a BSM IC. This approach simplifies enormously the complexity of the final circuit given that it already implements multiple features needed in a BMS in a single chip, such as voltage measuring and a way to control the cells' balancing. Because a robust and fast protocol is needed to establish the communication between the BSM and the MCU, the chosen MCU must have a Serial Peripheral Interface (SPI) bus capable of communicating at $1\,\mathrm{Mbit\,s^{-1}}$. Moreover, the vast majority of BSM chips available on the market use SPI. This SPI bus is also needed to communicate with the micro-SD card.

In order to monitor the battery temperature, thermistors must be added to the battery assembly. These thermistors may be multiplexed to reduce the total number of analog pins needed to measure all the temperatures. A current transducer or shunt must be installed to measure the current flow to and

from the battery. To control too high currents, the BMS must be capable of opening the load relay. The requirements related to the measurement of both battery voltages and cells' voltages are satisfied by the BSM itself. Regarding the control of the precharge relay, a comparator should be used to determine when the precharge is finished.

An optocoupler should be used to provide isolation between the battery and the low voltage GPIO pins of the MCU to sense the battery voltage. The measurement of the voltage of the five solar arrays may also be multiplexed so that only one analog pin needs to be used. Another current transducer or shunt must be used to measure the solar panels' current.

Regarding MCU, apart from the already mentioned $1\,\mathrm{Mbit\,s^{-1}}$ SPI bus, we need at least 15 GPIO pins (4 relays, 1 buzzer, 1 for fans, 2 for Chip Select pins for SPI, 3 battery voltage sense pins, 3 multiplexer channel select pins and one precharge end detector). Finally, we need up to 9 analog inputs for temperature and solar panels' voltage measurements, depending on whether the ones from the BSM are used or not.

4

Implementation

Implementation

In this chapter, the components and modules necessary to achieve the previously mentioned BMS architecture are presented both in what concerns the hardware and software sides of the system.

The hardware part is divided into three subsections, one for each version of the BMS that has been developed. The first version, V0, was just a test board used to test the communication between the MCU and the BSM. The second version, V1, was developed with the boat integration already in mind, but the lack of experience led to some malfunctions which, even though they didn't prevent the battery from being safe, were not desired on a final product. Nonetheless, this version was used in and still is on TSB's SR-02 prototype. The V2 is the final version designed during this work and is the version that is still in use today in TSB's most recent solar prototype, SR-03.

Lastly, the software implementation of the project is detailed. For this, a flowchart detailing the different aspects of the software are presented, followed by a detailed explanation of the most important parts. There is no need to divide it into multiple sections, given that the work was carried over from iteration to iteration.

4.1 BMSSPV V0 hardware project

The first step required to advance with the design of the proposed BMS is to choose which BSM will be used. There are a lot of BSM ICs on the market to choose from. In [17] there is a list that is frequently updated in [18] that compiles the most common BSMs. This list divides the chips by manufacture and classifies them into three different categories: not recommended, recommended and highly recommended. The ICs that are listed as not recommended were discarded as well as chips that are not available in the main distributors. According to the requirements, the voltage measurement error needs to be lower than $2\,\mathrm{mV}$, and thus, chips with a higher error were also discarded. We are left with only six possibilities. Given that for the same manufacture, there are almost identical options, with the only difference being the maximum number of cells that the IC can monitor, we also discarded those that can monitor more than 16 cells, since they tend to be more expensive and may require extra components or connections to terminate the unused pins. This leads us to the final four candidates, which are:

- LTC6811 from Analog Devices;
- MAX14920 from Maxim Integrated;
- MC33771 from NXP;

- bq76PL455 from Texas Instruments.

To choose the final BSM IC, we first looked at the recommendation given by Davide Andrea, who marks both NXP and Maxim solutions as recommended, while the Analog and Texas solutions are marked as highly recommended and high-performance solutions. To choose between both ICs, we searched for information regarding how to communicate with them and asked for advice from our colleagues from FST Lisboa who have already developed their BMS [19]. After finding that Analog Devices provides libraries for implementing almost all the features of the IC, which will make the software part easier to implement, and that FST Lisboa uses the same chip, we decided that the LTC6811 was the right choice for us. Besides that, it is more accurate than Texas' solution.

There are two versions of this IC: the LTC6811-1 and the LTC6811-2. By reading the datasheet [20] of the IC, Analog recommends the use of the LTC6811-2 when a single device is to be used since it requires fewer external components and consumes less power, especially when the communication interface is configured as a 4-wire SPI. Given that the chosen topology is centralized and each IC can monitor a maximum of 12 cells in series, and our battery has a 12s2p configuration, the LTC6811-2, hereinafter referred to as BSM is the one that will be used.

This BSM, when configured as a 4-wire interface, which has the advantages mentioned previously, communicates with the MCU over SPI and is capable of communicating with a speed of up to $1\,\mathrm{Mbit\,s^{-1}}$ which confirms the requirement imposed in Section 3.2.

Taking into consideration the requirements previously defined, as well as the choice made in what concerns the BSM IC the next step that should be taken in order to proceed with the development of the BMS is to choose which MCU will be used. By analyzing the software complexity, a 16-bit MCU would be able to handle the task. Considering that the team may want to add new functionalities to the proposed BMS in future project iterations, a 32-bit MCU provides a more future-proof solution that prevents the need to adapt the software from 16-bit to 32-bit in the future. Besides, nowadays, 32-bit MCUs are considered the industry standard since most embedded system R&D effort is focused on 32-bit cores, and thus both architectures can achieve similar power consumptions [21]. There are two main architectures available for 32-bit MCUs: AVR and ARM. The latter is the more common and most advanced. Given that, the ARM architecture is the choice.

Different options of the ARM architecture were considered, but ultimately, the choice was made based on what the team uses on the rest of the systems. Doing so, future modifications to the software will be relatively easy for any team member who is used to program other systems and code that is already developed by the team can be reused. This will be especially useful for the CAN library, given that the same code can be used on all the boat's systems, thus making updates much easier.

Within the team, all systems are based on Teensy boards like the one shown in Figure 4.1. These are Arduino compatible boards (on the software side), but with higher processing power and a lot more features, such as integrated CAN controllers and a software watchdog timer, that make them much more attractive. Their microcontrollers are all based on the ARM architecture, manufactured by NXP, ranging from a Cortex-M0+ to a Cortex-M7. Table 4.1 outlines the most relevant differences between the Teensy boards that are considered to be suitable for this work (there are two more boards, but they are much less powerful 8-bit AVR boards and thus were not considered).

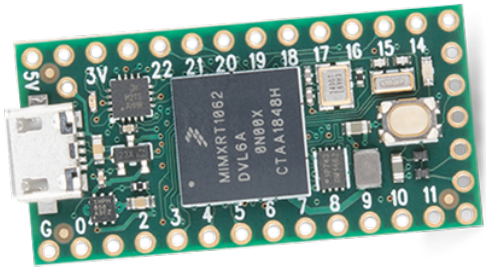


Figure 4.1: Example of a Teensy board, in this case the Teensy 4.0.

Table 4.1: Comparison between the different Teensy boards available[1].

Feature	Teensy LC	Teensy 3.2	Teensy 3.5	Teensy 3.6	Teensy 4.0	Teensy 4.1
Processor	MKL26Z64VFT4	MK20DX256VLH7	MKL26Z64VFT4	MKL26Z64VFT4	MKL26Z64VFT4	MKL26Z64VFT4
Core	Cortex-M0+	Cortex-M4	Cortex-M4F	Cortex-M4F	Cortex-M7	Cortex-M7
FPU [bits]	-	-	32	32	32 & 64	32 & 64
Rated Speed [MHz]	48	72	120	180	600	600
Overclockable [MHz]	-	96	-	240	912	912
Flash Memory [kB]	62	256	512	1024	1984	7936
Bandwidth [MB/s]	96	192	192	411	66	66
Cache [Bytes]	64	256	256	8192	65536	65536
RAM [kB]	8	64	256	256	1024	1024
EEPROM [Bytes]	128 (emulated)	2048	4096	4096	1080 (emulated)	1080 (emulated)
Pins						
Digital I/O	27	34	58	58	40	55
Analog Inputs	13	21	27	25	14	18
Usable Resolution [bits]	12	13	13	13	10	10
Timers	7 Total	12 Total	17 Total	19 Total	49 Total	49 Total
PWM, 32-bit	-	-	-	-	3	3
PWM, 16-bit	3	3	4	6	32	32
Low Power	1	1	1	1	None	None
Programmable Interval	2	4	4	4	4	4
RTC	None	1	1	1	1	1
Communication						
USB	1	1	1	1	2	2
SPI	2	1	3	3	2	2
With FIFOs	1	1	1	1	2	2
CAN Bus	-	1	1	2	3	3
With CAN-FD	-	-	-	-	1	1
SD Card	-	-	1	1	1	1
Price [€]	12,66	21,97	23,13	28,63	21,97	25,87

As it can be seen in Table 4.1, we have a few options to choose from. The Teensy LC can be discarded right away, given that it has neither a RTC nor a CAN controller, and it is not well supported by the community (libraries such as the one used for CAN communication do not support this board). Although one may be tempted to choose one of the Teensy 4.X, it is necessary to consider that the power consumption often rises as clock frequency increases, so it is necessary to ensure that all the processing power will be needed. Besides, both these boards are reasonably new and have just now been made available to the public, and thus most libraries are not yet compatible with these boards. An example of such a library is the one used for managing the different low power modes, which is essential for this work. So, at least at this time, the Teensy 4.X should not be considered.

[1] Source: Teensy Technical Specifications - Accessed on 17/10/2021.

With Teensy LC and Teensy 4.X excluded, we are left with three options: Teensy 3.2, Teensy 3.5 and Teensy 3.6. It is important to mention that although there is no native SD Card support on Teensy 3.2 (no 4-bit SDIO protocol), one can still use it by communicating over the SPI Bus.

By looking at the requirements, we can conclude that 21 digital pins will be needed, two of which need to have PWM (for the buzzer and fans), four analog pins, two pins for the CAN Bus and three more for SPI. This means that in terms of I/O, Teensy 3.2 is just enough for our needs.

The differences between Teensy 3.2 and 3.5 / 3.6 are mainly the clock frequency, the amount of memory, the form-factor, and the native SD-Card support. Considering previous projects with these boards, the Teensy 3.2 resources are more than enough for the task and given that no real-time data is to be stored on the SD Card, native support is not mandatory. Moreover, the footprint is way smaller, which will help with the board design. Thus the Teensy 3.2 will be the MCU used in this BMS.

Before committing to the full BMS project, it was decided first to develop a small test board that would allow us to test the main functions of the BMS. Although this version would not satisfy all the requirements defined, its main objective was to test the communication between the MCU and the BSM to make sure that the primary function of the BMS would work on the final version. This board, as well as its main sections, is shown in Figure 4.2.

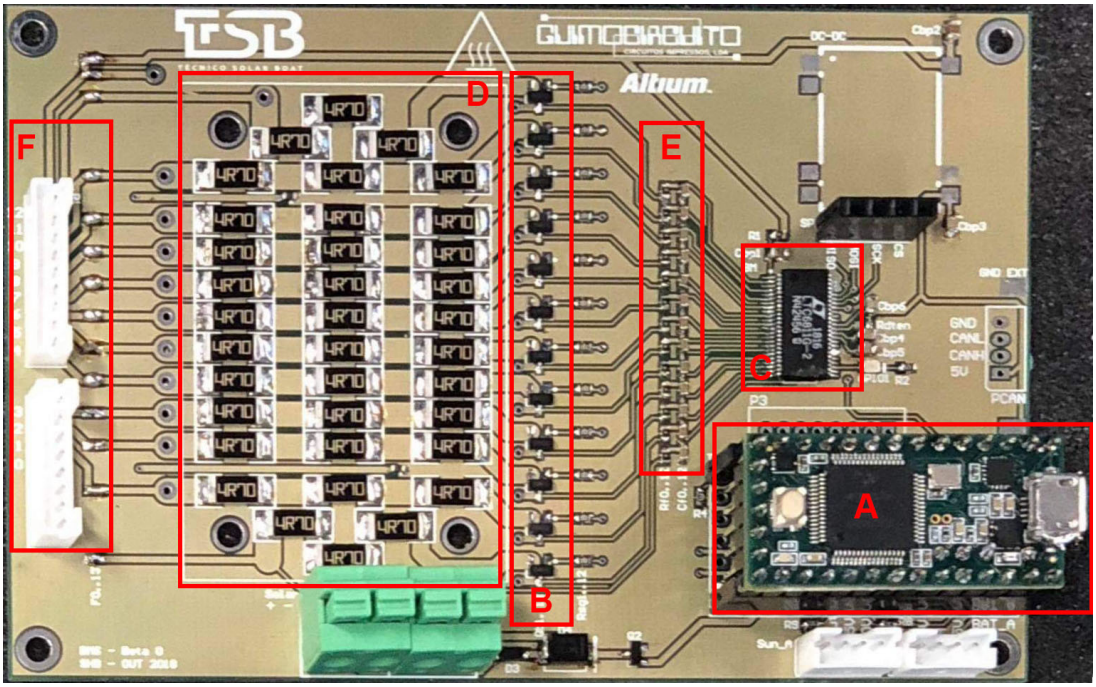


Figure 4.2: BMSSPV V0.

A: Teensy 3.2
B: Balancing MOSFETs
C: LTC6811-2
D: Balancing resistors and thermistors
E: Cells measurement filters
F: Cells connectors

This design was considered by the team a huge accomplishment. Even though not all the implemented functionalities were working as expected, its main objective was fulfilled. The communications between the Teensy and the BSM worked as expected, and the Teensy was able to acquire all the

desired data from the BSM and consequently be aware of the battery status.

Amongst the things that weren't working as expected, the most important one was the balancing Metal Oxide Semiconductor Field Effect Transistors (MOSFETs). Due to an error on the PCB schematic, their footprint was wrong, and thus, we were not able to balance the battery. This error was fixed by soldering the MOSFETs upside down as a workaround. The last thing that was not accounted for was the fact that the BSM's GPIO pins are open drain, and so they can only sink current. This made it impossible to turn on the debug led that was connected to GPIO 1 for testing purposes. With the knowledge acquired from this test board, it was decided that subsequent designs should measure the battery current transducers with the BSM GPIO pins instead of Teensy's analog pins so that this measurement can be synchronized with the cells' voltage measurement.

4.2 BMSSPV V1 hardware project

Following the successful tests performed on BMSSPV V0, it was now time to develop the first BMS that was made with the integration on the boat in mind. Inspired by our colleagues from FST Lisboa [22], we decided to build a BMS that would be on the top of the battery, thus eliminating all the cables needed to connect the BMS to each cell aggregate, the so-called tap wires. By doing so, the complexity of the system is considerably reduced. The computer aided design for this approach can be seen in Figure 4.3.

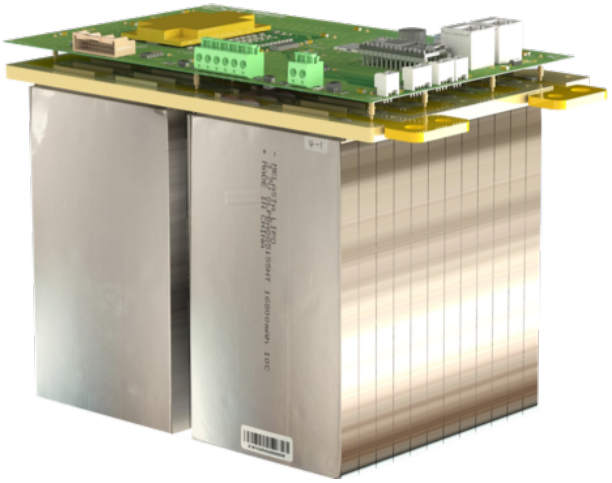


Figure 4.3: SR-02 battery assembly with spring loaded pins connecting the BMS to the cells.

Given that there were no problems identified in both the Teensy and the BSM IC chosen for the first iteration, these choices were carried over to V1. In the following subsections, each part of the BMS will be detailed and its design choices discussed. As we will see, V1 doesn't satisfy all the requirements. This is due to V1 being designed before the start of this thesis, and some of the requirements were only added after the design and tests performed on V1.

Figure 4.4 displays the most relevant blocks of the proposed BMS. Starting from left to right, we have both communication interfaces of the system to the exterior, namely the CAN bus and the Serial

interfaces. The CAN block also includes the Electrostatic Discharge (ESD) protection which is used to protect the CAN transceiver. Then there is the MCU which, in our case, is the Teensy 3.2 as we saw in Section 4.1. The MCU communicates over SPI with the BSM and also controls the contactors with GPIO pins. The MCU is powered with a DC-DC converter which provides $5\,\mathrm{V}$. The BSM is then connected to each cell aggregate of the battery pack to measure their voltage. The same connections are also used for balancing. The connection of the cells to these resistors is controlled by the balancing MOSFETs, which are controlled by the BSM. The thermistors that measure the cells' temperature are connected to multiplexers which are connected to the BSM GPIO pins which convert the analog voltage measured to a temperature value. Finally, there are two contactors for the charger and load and one relay for the precharge circuit.

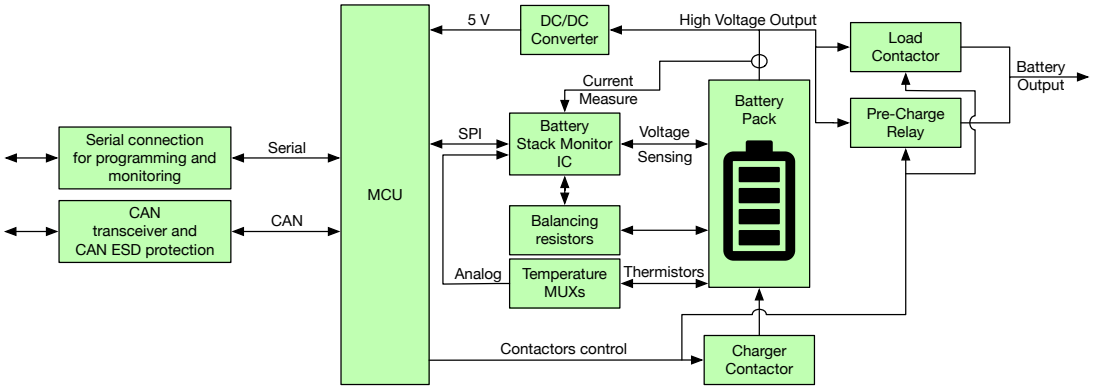


Figure 4.4: BMSSPV functional block diagram.

This version, as well as V0, were designed using Altium Designer and a two-layer stackup. Both top and bottom layers were used to route the different signals. Non used space was poured with a copper plane connected to ground. Both top and bottom ground planes are interconnected throughout the board using stitching vias. This technique is used to tie together larger copper areas on different layers, creating a strong vertical connection through the board structure, helping maintain a low impedance and short return loops.

4.2.1 Power supply

In order to continuously monitor the battery, the BMS needs to be powered by the stack. Otherwise, when the boat is turned off, the battery wouldn't be monitored. The Teensy and all the peripherals use either 5 or $3.3\,\mathrm{V}$. The $3.3\,\mathrm{V}$ can be obtained from the Teensy own $3.3\,\mathrm{V}$ regulator. So we only need to supply $5\,\mathrm{V}$ to the Teensy to obtain all the needed voltages.

There are mainly two options to do so. Either we convert the voltage of one cell aggregate $(3 - 4.2\,\mathrm{V})$ to $5\,\mathrm{V}$ or the stack voltage $(36 - 50.4\,\mathrm{V})$ to $5\,\mathrm{V}$. The latter is the most used for two main reasons: avoid battery unbalancing and the fact that a buck DC-DC tends to be easier to implement and has a higher efficiency than a boost one [23].

For this task, Traco Power TDR 3-4811WISM^2 DC-DC was chosen. It has an input range from 18 to $75\,\mathrm{V}$ and can provide a maximum of $600\,\mathrm{mA}$ at $5\,\mathrm{V}$. The battery voltage is connected to the DC-DC's

input through a fuse, and the output is distributed to all the components that need it.

4.2.2 Cell monitoring

The BSM is, without doubt, the most important HW part of the BMS. It is the part responsible for measuring the voltage of each cell aggregate, identifying unsafe conditions (like cell Over-Voltage (OV) or Under-Voltage (UV)), measuring both battery and solar currents, as well as the temperature sensors connected to the cells. At the same time, the BSM also controls the balancing of the cell aggregates to make sure the battery is kept balanced and thus allowing us to extract the maximum energy from it.

To design this part of the BMS, a thorough knowledge of the BSM and its functionalities are required to ensure not only the right connections are made but also that it is correctly configured. Some functions like the communication and discharge timers require specific connections to be made on the PCB as we will see later on.

Regarding the cells' voltage measurement task, each cell aggregate is connected, as shown on Figure 4.5, to a fuse and then connected to the BSM's C_n inputs. Between each fuse and the BSM, there is an Resistor-Capacitor (RC) low pass decoupling filter that helps to reduce the measurement noise, especially in the fast conversion modes, and to reject potentially damaging high energy transients.

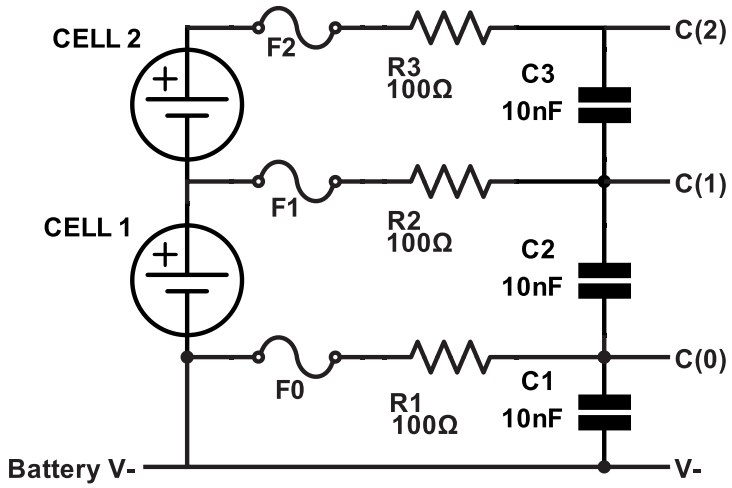


Figure 4.5: Differential capacitor filter connection scheme for the first two cells.

Adding more than about $100\,\Omega$ to the Analog-to-Digital Converter (ADC) inputs of the BSM begins to introduce a systematic error in the measurements, which can be improved by raising the filter capacitance or mathematically compensating in software with a calibration procedure. This behaviour can be seen in Figure 4.6. As the resistance increases above $100\,\Omega$, the total measurement error increases to values near $-10\,\mathrm{mV}$ for a capacitance of $100\,\mathrm{nF}$. When the capacitance is increased to $1\,\mu\mathrm{F}$, this measurement error is reduced by approximately $70\,\%$. Nevertheless, as stated on the datasheet, the configuration where the error is lower is when $R = 100\,\Omega$ and $C = 10\,\mathrm{nF}$ and thus the dimensioning of the used RC filter followed this recommendation.

[2] Datasheet for Traco Power TDR 3-4811WISM - Accessed on 17/10/2021.

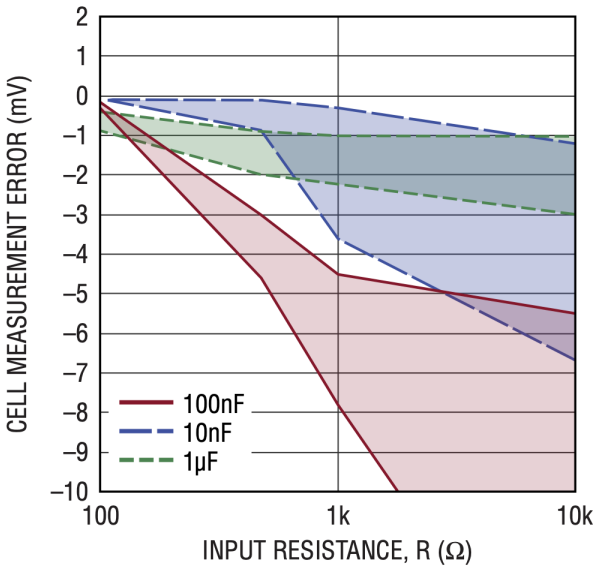


Figure 4.6: Cell measurement error range vs input RC values.

While the filter used is a differential filter, there is also another approach that uses a grounded capacitor filter. The chosen filter was the differential one, given that in this case, the capacitors only see the voltage of one cell aggregate, which allows us to use capacitors with a lower voltage rating and thus less expensive. The datasheet states that in systems where the noise is less periodic or higher oversample rates are used, a differential capacitor filter structure is adequate.

Concerning the communication part, there are two options to choose from SPI or Analog Devices' proprietary isoSPI protocol. As mentioned on Section 3.3, SPI should be used. This approach reduces the number of components, given that for isospin, a transceiver would be needed between the Teensy and the BSM as well as a pair of pulse transformers. Besides, Analog recommends that when only one BSM is needed, the LTC6811-2 should be used and configured to communicate over SPI in order to reduce power consumption. To achieve this, $ISOMOD$ and $A0$ to $A3$ pins should be connected to V^-.

4.2.3 Cell balancing

The last task of the BSM is to manage the balancing of the battery. As seen before in Section 3.2, this is not only desired, it is a mandatory feature. There are two options for balancing: passive and active. While the former is achieved by wasting the excess energy in heat, the latter redistributes the battery energy.

Although active balancing may seem attractive from a sustainability point of view, it involves a much more complex circuit, usually composed of multiple DC-DC converters. This solution increases not only the overall cost, but also the weight and volume of the final BMS. However, it is a key characteristic in certain EV applications since it allows, for example, that a parked car can have a boost of usable energy without being charged.

For this application, where there is plenty of time to charge the battery and solar panels are used, the extra weight and volume required to accommodate an active solution don't outweigh the disadvantages of passive balancing.

Moreover, the battery needs to be balanced at the start of the competition, so an unbalancing situation is not expected to happen during the competition given the very low self-discharge current of LiPo batteries. With that said, the chosen solution is passive balancing.

For this, two options are possible: internal or external MOSFETs. The BSM's internal MOSFETs only allow a balancing current of $60\,\mathrm{mA}$ or less. This means that for a cell aggregate with a energy of

$$16.8\,\mathrm{Ah} \times 3.7\,\mathrm{V} \times 2 = 124.32\,\mathrm{Wh}, \tag{4.1}$$

an unbalancing of $5\,\%$ and a balancing current of $60\,\mathrm{mA}$, the balancing time would be around

$$Balancing\ time = \frac{124.32\,\mathrm{Wh} \times 0.05}{60\,\mathrm{mA} \times 3.7\,\mathrm{V}} = 28\,\mathrm{h}. \tag{4.2}$$

The time calculated in (4.2) is just the time needed to balance one cell and assuming that the BSM would not overheat during this process. If more cells need to balance, the thermal footprint will increase and, with it, the balancing time. For this reason, the option with external MOSFETs is the choice. This way, a higher balancing current can be used without overheating the BSM.

In order to balance a cell aggregate, one PMOS transistor, Q_1, and a load, R_1 to R_3, are needed. The transistor's gate is connected to S_n while the source and drain are connected to the positive, C_n, and negative, C_{n-1}, terminals of the cell aggregate, respectively. The balancing load, in this case a resistor, should be placed between the drain and the negative side of the aggregate. The circuit for this solution is shown in Figure 4.7. This circuit is replicated 12 times, one for each series of the battery pack.

The datasheet suggests, as a rule of thumb, that a balancing circuit should be capable of correcting a $5\,\%$ SoC error within 5 hours, which according to

$$Balancing\ current = \frac{124.32\,\mathrm{Wh} \times 0.05}{5\,\mathrm{h} \times 3.7\,\mathrm{V}} = 336\,\mathrm{mA}, \tag{4.3}$$

corresponds to a balancing current of $336\,\mathrm{mA}$. However, a more conservative balancing current of around $300\,\mathrm{mA}$ was chosen, which corresponds to a balancing time of around $5.6\,\mathrm{h}$ for a $5\,\%$ SoC error, which is $80\,\%$ lower than what can be obtained while using the internal MOSFETs. Besides, multiple cell aggregates can be balanced together thus reducing the overall balancing time even further. This current is already higher than most commercial solutions, as seen on Table 2.1. To achieve such balance current, a series of three resistors with $4.7\,\Omega$ will be used.

Care must be taken while choosing these resistors to make sure their power rating is respected. The worst-case scenario is when balancing a fully charged cell. The power dissipated in each resistor is given by

$$P = R \times I^2, \tag{4.4}$$

which in this scenario translates to

$$4.7\,\Omega \times 0.3\,\mathrm{A}^2 = 423\,\mathrm{mW}, \tag{4.5}$$

so resistors with a power rating of at least $0.5\,\mathrm{W}$ must be used. It is also important to ensure a proper way for the heat to be dissipated away from the resistors to ensure that neither the PCB nor the resistors overheat. For this, a heatsink placed on top of the resistors is to be used.

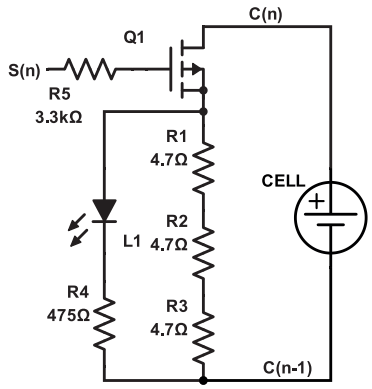


Figure 4.7: Balancing circuit generic schematic.

It was also decided to add a Light Emitting Device (LED), L_1, in parallel with the balancing resistors to have visual feedback of when a cell is balancing. The LED was only added after FST Lisboa reported to us that sometimes the BSM's discharge channels turn on unexpectedly.

The BSM has a feature called discharge timer. Such feature is controlled by the BSM's $DTEN$ pin and works as a timer for the balancing circuit so that balance can continue even after a WDT timeout. Each S_n pin can have a different timer applied to it. $DTEN$ should be pulled high to use such a feature. In this case, we will connect it to V_{REG} via a $0\,\Omega$ resistor so that the use of this feature can be decided later on.

4.2.4 Current measurement

To measure the battery and solar current, two options are possible. Either a Hall effect current transducer or a current shunt can be used. The former works by measuring the magnitude of the magnetic field, using the Hall effect, created by the current passing through the cable. This type of transducer typically doesn't require the circuit to be interrupted. In contrast, current shunts work by forcing the current to pass through a small known resistor. By applying Ohm's law to the voltage drop across this resistor, the current can be calculated. Both require signal conditioning so that the low voltage generated can be properly measured by an ADC. Hall effect transducers usually come in a ready to use form factor in which this signal condition has already been implemented, and the output signal can be directly read by an ADC. This signal conditioning also includes temperature compensation which makes Hall effect transducers more stable over a wider temperature range when compared to current shunts, given that the shunt resistance changes with temperature.

Considering that taking the current shunt approach would require an external circuit for signal conditioning and temperature compensation, which by itself would require some testing and tuning, and the fact that the team already had in stock Hall effect transducers for both of the currents to be measured, this was the chosen option. It is essential to pay attention to the current range of such transducers, given that an inappropriate range would lead to current readings with lower precision.

For the battery current, LEM's DHAB S/143[3] transducer was used. This transducer has two different current ranges: $I_{PN\ channel\ 1} = [-50\ ;\ 50]\,\mathrm{A}$ and $I_{PN\ channel\ 2} = [-450\ ;\ 300]\,\mathrm{A}$. These ranges allow us to have an acceptable accuracy independently of the current. When the battery is being charged, or the current consumption is below $50\,\mathrm{A}$, the value from channel one should be used. Otherwise, channel two

must be used. According to the datasheet, the transducer has an accuracy of $700\,\mathrm{mA}$ and $6\,\mathrm{A}$ for the first and second channel, respectively, under the range in which it will be used, which is an acceptable accuracy especially considering the technology being used.

For the solar current, LEM's HO 6-P[4] transducer was used. This transducer has a single measurement range of $I_{PM} = [-20\ ;\ 20]\,\mathrm{A}$ which is a little below what is needed. Thus, the transducer fixed offset functionality was used. To do so, an external voltage reference of $0.5\,\mathrm{V}$ was connected to the V_{ref} pin of the transducer, which shifts the measurement range to $I_{PM} = [0\ ;\ 40]\,\mathrm{A}$. Given that the sensor has an error of $1.65\,\%$ this yields an accuracy of $660\,\mathrm{mA}$.

As discussed in Section 4.1, given their voltage levels and for synchronization with cell measurements to be possible, the battery current should be measured with the BSM's GPIO pins. Namely, pins 1 and 2 should be used, given that these are the ones that can be synchronized with the twelve cell measurements. To do so, an `ADCVAX` command should be sent to the BSM. By doing so, a synchronization time of $543\,\mu\mathrm{s}$ is possible in the BSM's normal ADC mode.

The datasheet recommendations were followed in what concerns electromagnetic compatibility protection. The RC low pass filter chosen is the same used for the cells, given that this respects both the transducer and the BSM datasheets.

By having access to the current measurement, the BMS can detect an over-current situation. To satisfy the requirements, the BMS must not only detect this situation but also needs to stop it. To do so, high-current contactors are used. To stop an over-current situation while discharging, TE's EV200[5] high current contactor is used in series with motor controllers' connection. While in the case of charging over-current, Littelfuse's DCNLEV100[6] is used in series with the charging port. Even though the solar panels are not able to produce enough current to cause an over-current, the BMS also needs to be able to control them, and so one TE's KUEP-11D15-48[7] relay was used per solar array to allow the BMS to stop the battery's charging process and to comply with the regulations [3]. Fuses, with appropriated current ratings, were also used, but these are more suitable to stop a short circuit than an over-current situation. These contactors also allow the BMS to shut down the high power system as required.

4.2.5 Temperature measurement

Regarding temperature measurement, as discussed in Section 3.2, at least 16 temperatures must be monitored. Given the limited amount of analog inputs available on both the Teensy and the BSM, a solution had to be found.

Regarding cell temperature measurement, the requirements demand that there should be one thermistor per cell. Since we don't have enough analog inputs to measure 12 thermistors, there are two possible approaches: using an external ADC that can measure all the 12 temperatures or using a multiplexer, as suggested in Section 3.3, that allows us to use fewer analog inputs.

Both options have their advantages and disadvantages. For example, if an ADC is used, the temperatures can be measured simultaneously, thus allowing for a higher refresh rate when compared to the multiplexed approach. On the other hand, this would mean a higher current consumption given that all

[3] Datasheet for LEM DHAB S/143 - Accessed on 17/10/2021.
[4] Datasshet for LEM HO 6-P - Accessed on 17/10/2021.
[5] Datasheet for TE EV200 - Accessed on 17/10/2021.
[6] Datasheet for Littelfuse DCNLEV100 - Accessed on 17/10/2021.
[7] Datasheet for TE KUEP-11D15-48 - Accessed on 17/10/2021.

thermistors need to be powered simultaneously compared to the multiplexed approach where only the thermistors being measured are powered.

At an average resistance of $20\,\mathrm{k\Omega}$ at $25\,^{\circ}\mathrm{C}$ and a supply of $5\,\mathrm{V}$, choosing the ADC approach would mean, approximately, a $13\,\mathrm{mW}$ increase in power consumption, considering 12 thermistors according to

$$P = \frac{V^2}{R}. \tag{4.6}$$

In fact, for higher yet still normal temperatures, the Negative Temperature Coefficient (NTC) resistance would decrease, and the power consumption would double.

Moreover, following the ADC approach would make the PCB routing harder. Considering that the requirements allow us to have a lower refresh rate for the temperatures and that cell temperatures tend to be somehow slow to change, there is no added benefit in having a higher refresh rate. So the multiplexed approach was chosen.

Given that the multiplexed approach was chosen, it was decided to use a configuration with two eight-channel multiplexers, giving us four spare inputs for future use if needed. For this task, two Texas Instruments CD74HC4051M[8] were used. Care should be taken while choosing the multiplexers so that the channel select pin logic level is compatible with the Teensy one, especially considering that the multiplexers are powered with $5\,\mathrm{V}$, and the Teensy has a $3.3\,\mathrm{V}$ logic level. As shown on Figure 4.8, each multiplexer connects to 8 different thermistors, and then its output is connected to the BSM's leftover GPIO pins. Each multiplexer's corresponding channel select pin is connected in parallel and then to the Teensy. This approach allows us to control all the multiplexers on the board with the same three pins. As shown, this approach only requires one resistor that is shared with all the eight thermistors to form the voltage divider needed to calculate the temperature. This circuit is then replicated for the second multiplexer.

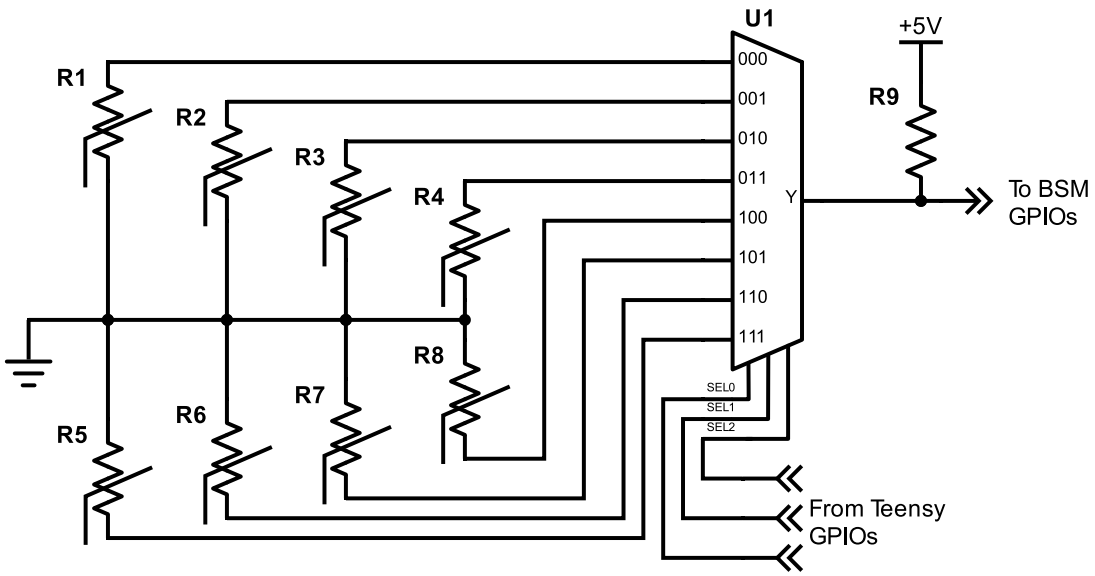


Figure 4.8: Multiplexed thermistors schematic.

[8] Datasheet for TI CD74HC4051M - Accessed on 17/10/2021.

The other temperature measurements that are required are the ambient temperature, the balancing resistors temperature (2) and the BSM temperature. For the first two, NTC thermistors were added to the PCB. Regarding the BSM die temperature, this can be obtained by sending the `ADSTAT` command to the BSM so no temperature sensor is needed. The Teensy analog pins measure the three NTC thermistors directly, given that there are no more inputs available on the BSM. The only drawback of this solution is that the thermistors should be powered with $3.3\,\mathrm{V}$ instead of $5\,\mathrm{V}$, given that this is the maximum supported voltage for analog signals applied to the Teensy.

Regardless of the device used to measure the temperatures, a voltage divider with a known resistor and the chosen thermistor must be used. Two things must be accounted for while dimensioning such components: the precision of the used resistors and the characteristic curve of the thermistor, β. The former influences the precision, while the latter the accuracy of the results that can be obtained. So resistors with a tolerance of $1\,\%$ or lower should be used, and the thermistor should be chosen so that most of the ADC range is used.

4.2.6 Precharge

A precharge circuit limits the damaging inrush current that a high voltage system with downstream capacitance can be exposed to when it is first turned on. A precharge circuit allows the current to flow in a controlled manner, slowly ramping up the voltage until it reaches a level near the source voltage. After that, the main contactor is permitted to close [24]. If the precharge is not correctly done, the contacts of the motor contactor can weld due to the high inrush current and prevent them from opening again.

For the precharge circuit, an external PCB was designed. This PCB was designed to be mounted on top of the motors' contactor, and it receives $24\,\mathrm{V}$ from the killswitch[9] and applies it to the positive side of both the motors' and the precharge relay's coils. This voltage is also applied to the BMS motor controllers sense port. The precharge relay shorts the motors' main contactor terminals with a precharge resistor and a fuse in series so that the current is limited. After a predefined delay, the precharge is completed, and the BMS closes the motors' contactor with a low-side switch, allowing all the current to flow to the motor controllers if it decides that it is safe to do so. This circuit is shown in Figure 4.9. The motor controllers were replaced by their equivalent capacitance in order to illustrate the downstream capacitance. Apart from protecting the precharge resistor, the fuse prevents that the motors spin without the main contactor being closed.

[9] Security device that is attached to the pilot and cuts the power to the motors in the event of the pilot falling over-board. It is also used by the pilot to turn on and off the motors.

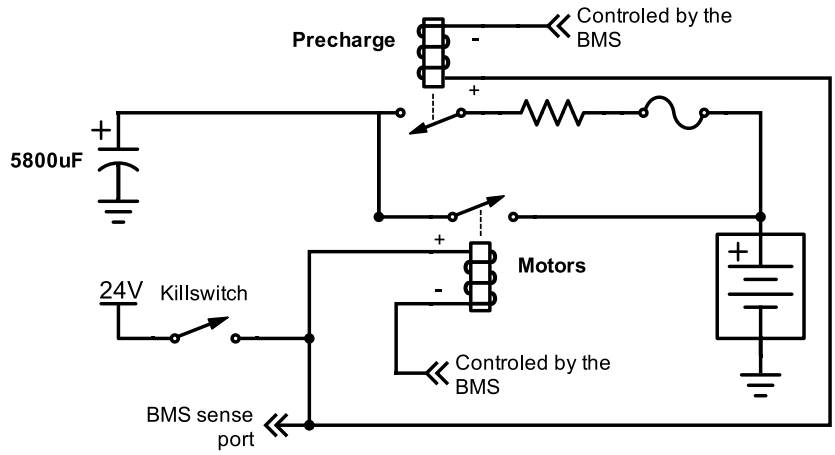


Figure 4.9: Precharge circuit schematic for V1.

Figure 4.10 shows a LTSpice simulation for the precharge of the motor controllers' capacitors considering a capacitance of $5800\,\mu\mathrm{F}$ and a precharge resistor with $47\,\Omega$. As it can be seen, the resistor can limit the initial current spike, when the precharge relay closes, to about $1\,\mathrm{A}$ while the capacitors' voltage slowly ramps up to near the input voltage. When the main contactor is closed, at around $1.8\,\mathrm{s}$, there is a current spike, but this is only of $4\,\mathrm{A}$, which is more than safe for the contactor. With this simulation, we can conclude that it is safe to close the motors' contactor $2\,\mathrm{s}$ after the precharge is started.

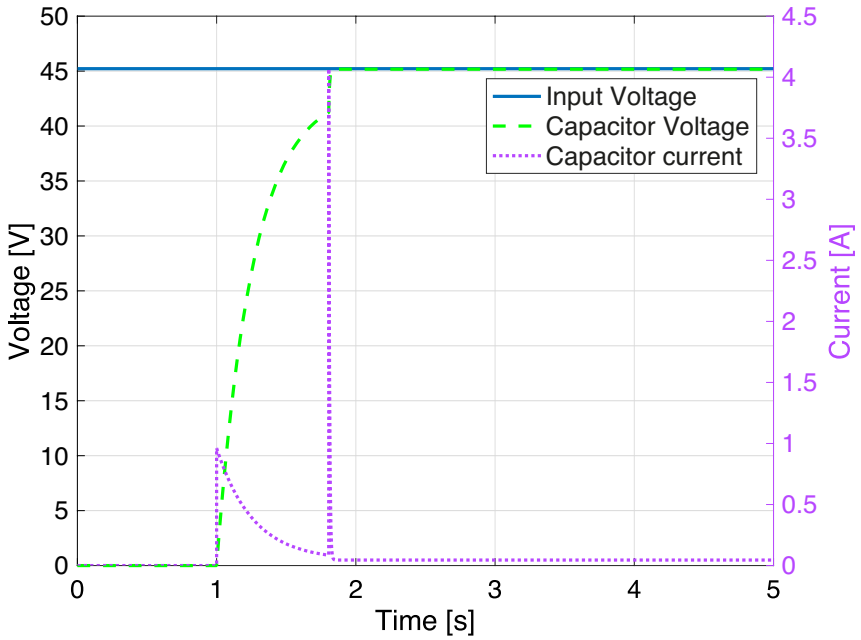


Figure 4.10: Capacitors inrush current simulation with the precharge circuit.

This simulation is also useful to determine the power rating that should be used for the precharge resistor. The highest current that passes through the resistor is when the precharge is started, and in this case, it is $0.96\,\mathrm{A}$, according to (4.4) the maximum power that needs to be dissipated on the resistor

is $43.32\,\mathrm{W}$ and given that this is just for a very small interval of time a resistor with a power rating of $50\,\mathrm{W}$ can be used.

4.2.7 Remaining subsystems

According to the requirements, there is the need to sense the battery voltage in specific points of the system, namely on the charger port and on the motor controllers input (the power supply to this system is controlled by the pilot, with the killswitch, but the BMS decides whether it is safe to turn it on or not). To achieve this, an optocoupler circuit, shown in Figure 4.11 was designed. This provides the necessary isolation between the battery voltage and the Teensy GPIO pins. Care should be taken while designing such a circuit so that the continuous forward current of the optocoupler, I_F, is not exceeded. For this resistor R_1 is used. This current can be calculated with

$$I_F = \frac{V_{in} - V_F}{R_1}, \tag{4.7}$$

where V_{in} is the maximum expected voltage to be sensed while V_F is the forward voltage of the optocoupler's emitter. Considering the dual-channel optocoupler MOCD213M[10] from onsemi (4.7) gives

$$I_F = \frac{50.4 - 1.15}{5000} = 9.85\,\mathrm{mA} \tag{4.8}$$

which is lower than $60\,\mathrm{mA}$ that is the absolute maximum I_F at $25\,^{\circ}\mathrm{C}$.

R_2 is just a pull-down resistor used to ensure that the Teensy input pin will have a defined state when there is no voltage applied to the emitter side of the optocoupler.

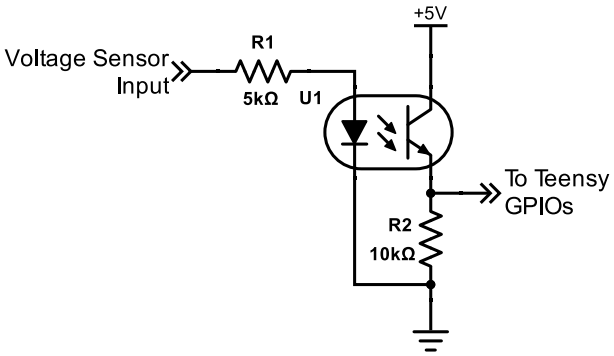


Figure 4.11: Optocoupler schematic.

To control the different relays and contactors, the BMS cuts the Ground (GND) supply to the coils with an N-channel MOSFET, as shown in Figure 4.12. The positive side of the coils is connected to the $24\,\mathrm{V}$ emergency DC-DC[11]. Alternatively, the BMS is ready to supply the battery voltage directly to the positive side of the coils, but for that, $48\,\mathrm{V}$ nominal coils should be used, but the team couldn't source these parts.

[10] Datasheet for onsemi MOCD213M - Accessed on 17/10/2021.

[11] This DC-DC is a 44.4 to $24\,\mathrm{V}$ DC-DC which turns on as soon as the boat is turned on (both emergency and electronic switches are closed) and it powers the positive side of the coils of all relays within the battery box. This part is only necessary given that the team couldn't source relays with $48\,\mathrm{V}$ nominal coils.

Flyback diodes were also added to the coils of the relays in order to protect the MOSFETs from the back Electromagnetic Force (EMF) caused by the coil's switching.

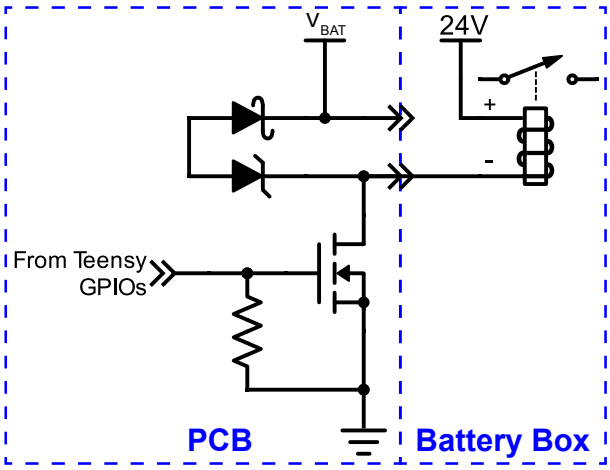


Figure 4.12: Relays schematic.

The battery fans are also controlled using an N-channel MOSFET acting as a low side switch like the relays. The only difference, in this case, is that the gate of the MOSFET is connected to a Pulse Width Modulation (PWM) capable pin so that the speed of the fans can be controlled.

To measure the voltage of the solar arrays, an approach similar to the one used for measuring temperatures was used. The measuring points are connected to a multiplexer, and then its output is connected to one of Teensy's analog pins. Given that the voltage produced by the solar arrays is way above the maximum limit both the Teensy and the multiplexer can handle, these signals had to be attenuated. For this, a voltage divider was used. To dimension such a divider, two things needed to be known: the open circuit voltage of each solar cell, V_{OC}, and the maximum number of solar cells in series. For the SR-03, these are $0.68\,\mathrm{V}$ and 80, respectively, and so

$$V_{OC_{max}} = 80 \times 0.68 = 54.4\,\mathrm{V}. \tag{4.9}$$

Given that the maximum voltage supported by the Teensy's analog pins is $3.3\,\mathrm{V}$ and considering that the top resistor of the voltage divider, R_1, has $10\,\mathrm{M\Omega}$, so that the power consumed by the voltage divider is reduced, the maximum value of the bottom resistor, R_2, is given by

$$R_{2_{max}} = -\frac{V_{out} \times R_1}{V_{out} - V_{OC_{max}}} = -\frac{10 \times 10^6 \times 3.3}{3.3 - 54.4} = 645.7\,\mathrm{k\Omega}. \tag{4.10}$$

Considering that $V_{OC_{max}}$ can increase in temperatures lower than $25\,^{\circ}\mathrm{C}$ or an irradiance higher than $1000\,\mathrm{W\,m^{-2}}$, it was preferred to have a safety margin as opposed to using the full ADC range an so R_2 will be equal to $619\,\mathrm{k\Omega}$ which gives a maximum output voltage, when $V_{OC_{max}}$ = $54.4\,\mathrm{V}$, of

$$V_{out} = \frac{V_{OC_{max}} \times R_2}{R_1 + R_2} = \frac{54.4 \times 619 \times 10^3}{10 \times 10^6 + 619 \times 10^3} = 3.17 < 3.3\,\mathrm{V}. \tag{4.11}$$

Regarding the communications aspect of the BMS, both CAN Bus and USB serial are needed. For

CAN the only thing needed is to add a suitable CAN transceiver to the board and connect it to the CAN-TX and CAN-RX pins of the Teensy. These pins are connected to the integrated CAN controller. The transceiver used was Texas Instruments SN65HVD1050DR[12], since this is the part that is already used in the rest of the boat systems and that it satisfies the requirements. On the other hand, using the USB serial doesn't require any additional hardware, given that Teensy's USB programming port can also be used as a serial port.

The only three aspects of the requirements still not covered are the RTC, the buzzer and the micro-SD card. The Teensy 3.2 already as a RTC incorporated onboard, which only needs an external $32.768\,\mathrm{kHz}$ crystal to be added to the dedicated through-hole pins. The micro-SD, as discussed in Section 4.1, is connected to the Teensy using its SPI port, which is shared with the BSM, except for the chip select line. Finally, the buzzer is driven by one of the Teensy's PWM pins through an NPN transistor.

4.2.8 BMSSPV V1 problems

Following the design, manufacture and assembly of this version, it was time to test it. The first thing to test was if there were any problems with the contact between the pogo pins and the battery bus bars. A small test board, which was manufactured in the TagusPark's PCB Laboratory, with just the pogo pins, had already been tested prior to committing to this design, and thus as expected, there were no problems with this as the BMS was always able to make good contact with the bus bars.

Concerning the malfunctions, we found that the coil suppression circuit for the relays was not correctly designed and, even though the relay should be open, there was still some voltage across the relay coil, which kept it closed. A quick fix was made by removing this circuit from the PCB and implementing it directly on the relay with different diodes. After that, we also found out that the voltage reading of the solar panels wasn't working correctly. It always gave random values, almost like it is was measuring a floating voltage.

The subsequent two malfunctions were only discovered during the competitions. The first one was related to the precharge circuit, and the second had to do with the solar panels' relays. Regarding the former, the circuit was badly designed since there was no way for the BMS to open the precharge relay. This error made it impossible for the BMS to cut the load in case there was a fault with the battery. This problem was discovered during the discharge test done by the organization to calculate the battery capacity when the first cell reached its UV threshold. The workaround that was implemented consisted in removing the precharge relay, even though doing so is not a recommended procedure. In what concerns the solar panels' relays, the BMS wasn't always able to open the solar panels' relays. This happened because the current from the solar panels was sometimes higher than the relays' break current rating, which kept the contacts closed, even though there was no voltage at the coil.

After the summer break, at the beginning of September, when one of the team members arrived at our workshop, he encountered a buzzing BMS, meaning that some fault was detected by the BMS. We ended up discovering that the BMS had discharged the battery entirely, even though when the competition ended, it was fully charged and was never used again. After further testing, we discovered that the BMS was consuming almost $1\,\mathrm{W}$ continuously. Knowing that the battery had $1522\,\mathrm{Wh}$ measured during the competition's technical inspection, the time it takes for it to be completely discharged can be

[12] Datasheet for TI SN65HVD1050DR - Accessed on 17/10/2021.

calculated with

$$\Delta T = \frac{E_{stored}}{E_{consumed}} \tag{4.12}$$

which results in

$$\Delta T = \frac{1522\,\mathrm{Wh}}{1\,\mathrm{W} \times 24\,\mathrm{h}} = 63.4\,\mathrm{d}, \tag{4.13}$$

which corresponds, approximately, to the number of days between the last competition and the day the fault was identified. This validates the theory that the battery had discharged due to the BMS's power consumption.

The last known problem is related to the onboard micro-SD card. There was an error in the schematic, and all the pins got shifted, making data logging impossible.

Apart from the previously mentioned hardware malfunctions, in what concerns the software part, three things were not implemented in this version: the computation of battery information (such as SoC and the number of cycles), the sleep function of the microcontroller and the possibility for the user to configure BMS parameters through the GUI.

4.3 BMSSPV V2 hardware project

Considering the problems previously mentioned and the fact that some of them required hardware modifications, it was decided to build a second and final version of the BMS. This version is the one that was to satisfy all the requirements, given that some requirements were only imposed after the problems that affected the first version were identified.

From all the previously mentioned problems, power consumption is the most important one that needed to be addressed. The battery is supposed to have a BMS connected at all times, and such a system should not cause the battery to be completely discharged in a matter of a couple of months. To solve such a problem, apart from a new power supply architecture, the use of Teensy's sleep function is crucial. In order to use it, we need to identify how the BMS should wake up from sleep.

V2 was also designed with Altium Designer. However, in contrast to V0 and V1, which were two-layer PCBs, V2 was made with a four-layer stackup in which the first and fourth layers are mainly signal layers, the second layer is a big ground plane and the third layer has routed power connections.

A four-layer board, even though more expensive, allows for a better ground which makes a more reliable BMS. Using only two layers on this design would not leave enough space for proper ground distribution on the entire PCB, and thus a four-layer design was preferred.

In the following sections, we will focus on the parts that differ from the previous version, and thus some parts that haven't changed will not be addressed again.

4.3.1 Power supply

By carefully analyzing the Traco Power's TDR 3-4811WISM datasheet [25], which was the DC-DC used in V1, we can see that its quiescent current is typically $13\,\mathrm{mA}$ which translates to about $577\,\mathrm{mW}$. This parameter wasn't checked when developing V1, and it turned out to be responsible for $58\,\%$ of the BMS's overall power consumption. The other components responsible for overall power consumption were the currents transducers, given that they were powered at all times and each one consumes about $100\,\mathrm{mW}$. Table 4.2 resumes the different theoretical power consumptions for each device onboard V1.

Table 4.2: BMSSPV V1 power consumption.

Device	Power Consumption V1 [mW]
Traco DC-DC	577
Current Transducers	195
MCU	160
Board ICs	53
Thermistors	8
BSM	0.252
Total	993

In order to considerably reduce the power consumption, not only a new DC-DC had to be used, but also devices that are not being used during sleep had to be switched off, and Teensy's sleep functionalities had to be implemented. The LTC6811 datasheet actually has a recommendation for an improved regulator power efficiency which is based on the Linear technology's LT3990[13] step-down regulator, which has a maximum quiescent current from $V_{IN} = 1.2\,\mu\mathrm{A}$ when in sleep mode. This sleep mode is controlled by the $EN/UVLO$ pin, which should be connected to the LTC6811's $DRIVE$ pin, automatically putting LT3990 into its lower power mode when the LTC6811 is in the $SLEEP$ state.

Putting the $5\,\mathrm{V}$ DC-DC in sleep mode causes the MCU also to lose its power supply which would ultimately mean that the system would never wake up again. This can be solved by powering the MCU directly with $3.3\,\mathrm{V}$ thus improving the Teensy's power consumption. This way the Teensy's 5 to $3.3\,\mathrm{V}$ Low Dropout (LDO) regulator, which has a quiescent current of $100\,\mu\mathrm{A}$, is bypassed.

Given that, besides from generating $5\,\mathrm{V}$, we now have to also generate $3.3\,\mathrm{V}$ to power the MCU. To do so, we can also use an LT3990 which will make the design easier given that it can be reused for both DC-DCs. There are different versions of the LT3990 available. In its normal version, the output voltage can be adjusted using a resistor network, but the other two variants, LT3990-5 and LT3990-3.3, are more attractive for our design given that they provide a fixed output voltage of 5 and $3.3\,\mathrm{V}$, respectively, which will reduce the number of external components needed, thus making the PCB design easier.

Nonetheless, given that there are still other components onboard supplied by the $3.3\,\mathrm{V}$ bus that don't need to be powered while the system is sleeping, like the micro-SD card, a P-channel MOSFET was added in between the supply voltage and these components. The components powered by the $5\,\mathrm{V}$ bus are automatically turned off when the LT3990-5 enters its sleep state.

Considering that without a functional power supply, the BMS board is useless, it was decided to first develop a small test board that could be used to validate both the DC-DC PCB design and its power consumption while sleeping. This board, shown in Figure 4.13 was manufactured in TagusPark's PCB Laboratory and followed, as much as possible, the routing that would be used in the final BMS. Due to being a single layer board this wasn't always possible.

[13] Datasheet for Linear Technoloy LT3990 - Accessed on 17/10/2021.

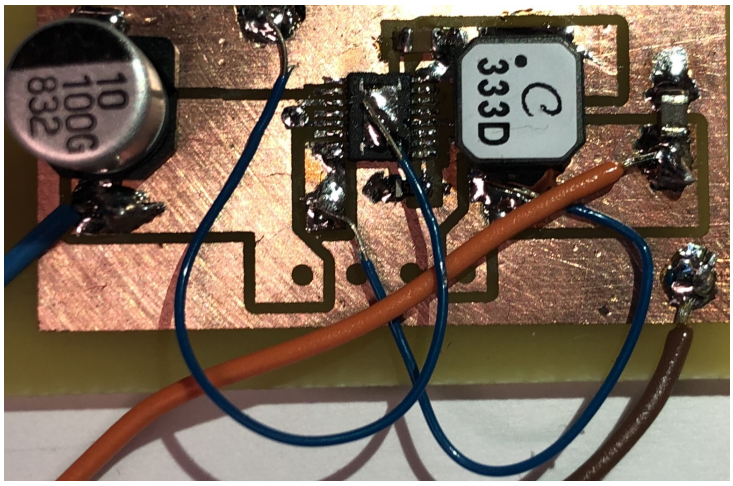


Figure 4.13: LT3990-5 test board.

Tests performed on this board showed that it was capable of performing according to the datasheet, which proves that the design was successful and could be used on the final BMS.

The schematic for this new power supply is shown in Figure 4.14. It followed the datasheet recommendation. In LT3990-3.3's case, the $EN/UVLO$ pin was connected to V_{IN} since this DC-DC should always be powered. In contrast, in the LT3990-5's case, this pin was connected to the LTC6811's $DRIVE$ pin, as suggested on the datasheet.

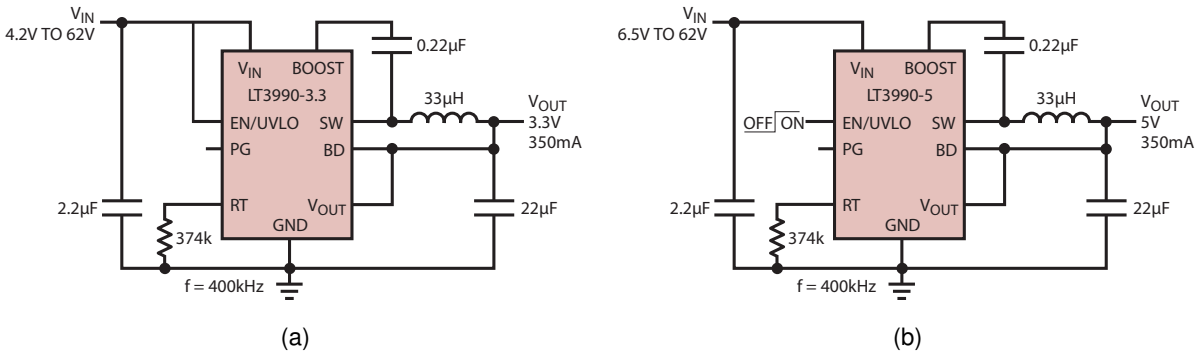


Figure 4.14: Power supply schematic for both LT3990-3.3 (a) and LT3990-5 (b), extracted from [26].

The LT3990-3.3 has a maximum quiescent current, when $EN/UVLO$ is high, of $4\,\mu\mathrm{A}$, which is much lower than the $13\,\mathrm{mA}$ of Traco Power's TDR 3-4811WISM used in V1. So having this DC-DC always on isn't a problem as long as the Teensy and the devices connected to it are turned off or put in a sleeping mode.

4.3.2 Sleep mode

There are essentially three situations that should cause the system to wake up: when the boat is turned on, when the USB cable is plugged into the Teensy, and periodically in order to check the battery status while it is not being used. To wake up the system when the boat is turned on, a new voltage

sensing port was added compared to V1. This port is connected to the emergency switch. As soon as the switch is closed, the battery voltage enters the optocoupler and subsequently in one of the MCU pins.

For the USB wake up, the Teensy has a pin called V_{USB} which is connected to the USB cable power conductor. By connecting this pin to another digital pin of the MCU, we can also sense that the USB cable has been connected and thus wake up the system.

These pins must be chosen carefully, and it is important to pay attention to the Teensy's Snooze library documentation [27]. Here, all the different ways to wake up the Teensy are explained as well as the pins that can be used for pin-based wake-ups.

To wake the Teensy up periodically, the timer or alarm drivers of the Snooze library can be used, and no additional hardware components are needed.

4.3.3 Cell balancing

Even though there were no considerable changes in this section of the BMS the balancing resistors were changed so that a faster balancing can be possible. The circuit now uses a series of three $2.7\,\Omega$ resistors which translates to a balancing current of around $520\,\mathrm{mA}$. This results in a balancing time of $3.2\,\mathrm{h}$ for a $5\,\%$ SoC error according to (4.2) which is $43\,\%$ less compared to V1.

Given that the power that needs to be dissipated in each resistor is now given by

$$P = R \times I^2 = 2.7\,\Omega \times 0.52\,\mathrm{A}^2 = 730\,\mathrm{mW}, \tag{4.14}$$

resistors with a power rating of at least $1\,\mathrm{W}$ should now be used. To accommodate for this extra generated heat a taller heatsink was also used in this version.

4.3.4 Current measurement

After a more intensive search of the different options that can be used to measure both the battery and the solar current, a new device was found. This device is an intelligent, digital voltage and current sensor developed to precisely measure DC currents.

The IVT-S from Isabellenhütte[14] is capable of not only measuring the current by using a shunt resistor, but it can also measure up to three voltages referenced to the battery ground. This device includes an onboard energy counter which will be very useful for calculating the battery SoC. Moreover, the temperature compensation mentioned in Section 4.2.4 is already internally implemented. Finally, the device is capable of sending all the acquired data to the boat's CAN Bus.

Compared to the Hall effect transducers used in V1, this sensor offers higher precision. With that in mind and the functionalities mentioned above, it was decided that V2 would use two of these devices to measure both the battery and solar panels' current. Nonetheless, V2 PCB is still prepared to use Hall effect transducers so that it can be compatible with the SR-02 electrical system.

The IVT-S is powered by the boat's $24\,\mathrm{V}$ bus, which means that when the boat is off, the sensor will automatically be off, and thus it will not consume any power, which solves one of the problems identified in V1.

[14] Datasheet for Isabellenhütte IVT-S - Accessed on 17/10/2021.

4.3.5 Precharge

Given that the precharge PCB designed for V1 was not working as supposed, a new PCB had to be designed. The circuit, which is shown in Figure 4.15, is very similar to the previous one. The only difference is the fact that the low side of the precharge relay coil is controlled by the BMS, which means that now the BMS has the last word on whether the precharge should even be started or not.

This approach is a lot safer given that in V1, the precharge was automatically started when the pilot closes the killswitch, which meant that the motor controllers could be powered even if there was an under-voltage situation, which is not desired.

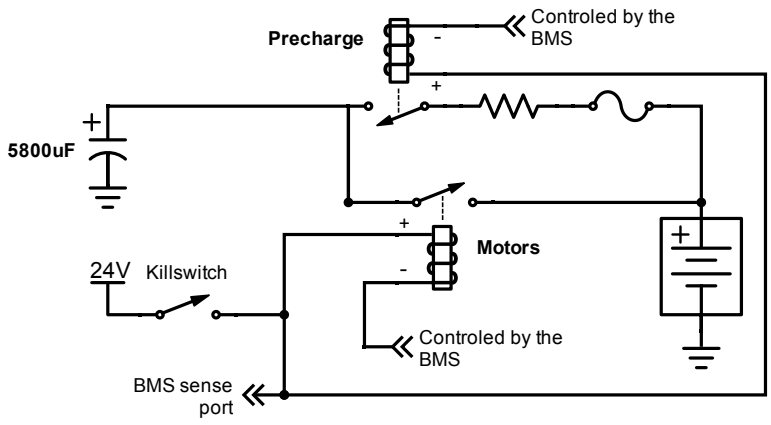


Figure 4.15: Precharge circuit schematic for V2.

Apart from the new precharge PCB, a new circuit was implemented on the BMS so that we no longer have to rely on a predefined time interval to activate the main contactor. The implemented circuit, shown in Figure 4.16 whose simulation is shown in Figure 4.17, consists of a comparator and a few resistors. The comparator compares the voltage at the motor controllers input (green line) with the battery voltage (blue line). When the former is within approximately $90\,\%$ of the latter, a logic high signal (purple line) is sent to the MCU which then closes the motors' contactor and opens the precharge relay.

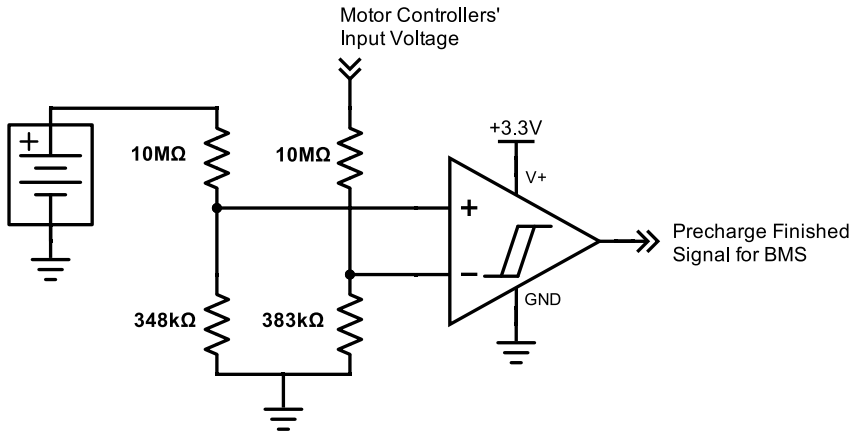


Figure 4.16: Precharge finished sensor schematic.

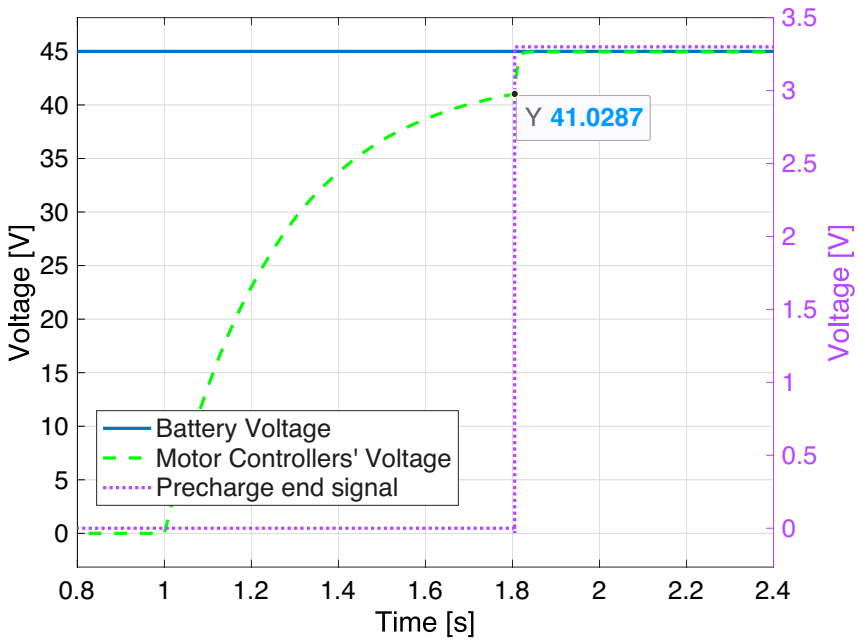


Figure 4.17: Precharge finished sensor simulation.

4.3.6 Solar MOSFETs and relays

For this version of the BMS, a new way to comply with rule [3] had to be implemented. This rule stated, when the development of this BMS began, that there should be a way to interrupt the power supply between the solar panels and the MPPT. This means that for each solar array, there must be a device that can interrupt the aforementioned power supply. As seen in Section 4.2.8, the previously used relays were not able to perform this task given that their DC current and voltage ratings were surpassed by the solar panels' characteristics.

Finding a suitable alternative to TE's KUEP-11D15-48 relays was, and still is, a problem, given that for the current and voltage ratings needed, only high current contactors like LittleFuse's DCNLEV100

are available on the market. Given that SR-03 has seven solar arrays, we would need seven contactors, which would continuously consume $42\,\mathrm{W}$ and cost around $300\,€$.

To solve this, a cheaper solution that uses MOSFETs instead of relays was designed. The problem with such a solution is that MOSFETs are controlled differently from relays, so the old circuit wasn't suitable anymore. Figure 4.18 shows the new circuit. It consists of one comparator and an AND logic gate. The comparator outputs a logic high when the boat is turned on. The AND outputs a logic high only when both the boat is on, and the BMS wants the solar panels to be on. This way, as soon as the boat is turned off, the solar panels are disconnected (the comparator produces a logic low) no matter what the BMS status is. This circuit could theoretically be implemented in software, although since it is a safety feature imposed by regulations, it was preferable to implement it in hardware.

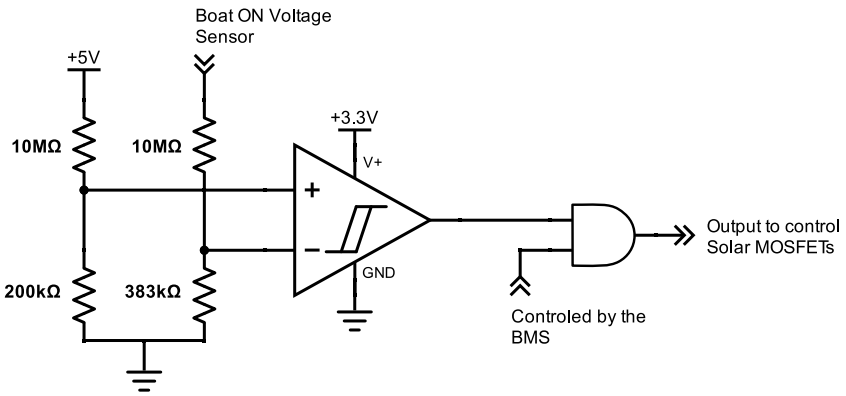


Figure 4.18: Solar MOSFETs control circuit schematic.

However, after the PCB had already been designed, new regulations were published. These now allowed to interrupt the power between the MPPTs and the battery instead of interrupting it between the solar panels and the MPPTs. This means that we can now open the circuit after connecting the seven MPPTs' outputs in parallel, thus allowing us to use a single relay to comply with the regulations. As a result, on the final electrical system, one TE's EV100[15] contactor is used to control the solar panels. Contactors are considerably safer than MOSFETs, given that they aren't so susceptible to the effect of voltage transients caused by the inductive nature of the long cables connecting the solar panels to the MPPTs.

4.3.7 Temperature measurement

Considering the new battery water cooling system presented in Section 3.1 and to make it possible to evaluate the performance of the aforementioned system, a lot more thermistor inputs were added to this version. In total, this version supports up to 40 external thermistors. The idea behind this is to allow us to have one thermistor per cell (24) and another one per bus bar (13), totalizing 37 thermistors. By placing the cell's thermistors near the bottom of its body, as shown in Figure 4.19, and another near its tab on the bus bars, we will be able to evaluate the cooling performance and decide whether more cooling is needed in future iterations and where.

[15] Datasheet for TE EV100 - Accessed on 17/10/2021.

Figure 4.19: Thermistor placement on the cell's body.

To make it possible to read 40 thermistors, three more multiplexers were added to the circuit, making a total of five dedicated to temperature measuring. Given that in this version, the BSM's GPIO 1 and 2 are no longer needed to measure the current transducers, as seen in Section 4.3.4, these pins and the GPIO 5 were used to read the output voltage of the newly added multiplexers. However, the connection to GPIO 1 and 2 can be broken by the user if Hall effect sensors are needed.

4.3.8 Remaining subsystems

Following the problems faced in V1 regarding the control of the external relays and contactors, a new approach was followed for V2. The new control circuit, shown in Figure 4.20, now uses a small onboard signal relay, directly powered by the battery voltage and actuated by a low side switch with an N-Channel MOSFET. This signal relay then has its COM pin connected to GND, by default, and its NO pin exposed to the outside to act as a low side switch to external contactors. The signal relay has a diode across its coils to protect the MOSFET against back EMF produced by the coil switching. Like before, the option to supply the battery voltage to external relays is still present.

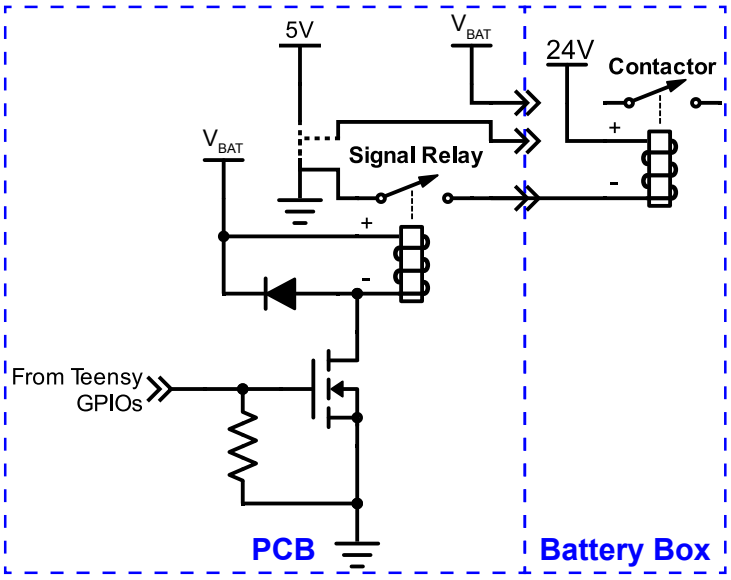


Figure 4.20: Relays schematic.

The advantage of this approach is that the described behaviour can be changed by the final user so that the relay can act as a switch or even as a high side switch, allowing for a lot more possibilities and future usages. The user can make these modifications by cutting the trace that connects the COM pin to GND. To convert it to a high side switch, the $5\,\mathrm{V}$ pads should be soldered together. Simply cutting the GND trace will make the relay act as a switch, given that the relays' COM pins are also exposed. These jumpers are illustrated in Figure 4.21.

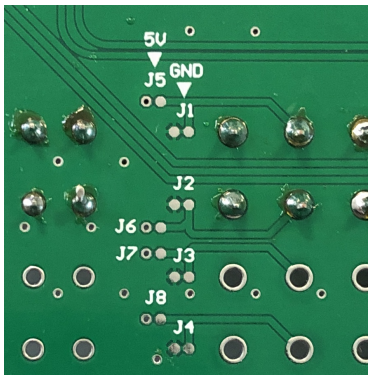


Figure 4.21: Jumpers used to change the relays' behaviour.

Even though only three relays need to be controlled according to the requirements, this circuit was replicated five times so that the BMS can control two extra relays accounting for non-planned situations and thus future-proofing it.

Other options were considered instead of the signal relay, such as a MOSFET or a reed relay. The former could not provide the flexibility of changing the switch's behaviour while the latter, even though it offers a higher mechanical life than traditional relays, there are no options that can be powered by the battery's voltage directly. Furthermore, the solutions found consume more energy than the chosen relays. With that in mind, the MOSFET and reed relay approaches were ultimately discarded in favour of the signal relay, which is completed sealed and thus not affected by the marine environment.

Apart from this change, a microprocessor supervisor circuit was added to the PCB. The function of this device is to assert a reset signal when the power supply voltage drops below the reset threshold voltage and to retain this reset for the reset timeout period once the power supply increases above the reset threshold voltage. The Microchip's MIC803[16] was used for this task. The need for this device arose with previous experiences with other PCBs where the Teensy could not boot up properly due to the DC-DC start-up time.

Finally, not only to comply with the requirements, but also due to previous problems related to damaged CAN transceivers, a CAN Bus protector was also added to this version. The chosen device was the Bourns' CDSOT23-T24CAN[17] which consists of an array of two Transient-Voltage-Suppression diodes protecting both data lines, thus providing ESD and surge protection to the transceiver. This device is particularly suitable for high-speed CAN buses, such as the one used in SR-03, because of its low capacitance and low leakage current, thus making it perfect for the task.

[16] Datasheet for Microchip MIC803 - Accessed on 17/10/2021.
[17] Datasheet for Bourns CDSOT23-T24CAN - Accessed on 17/10/2021.

4.4 Software

The main tasks the BMS's software has to accomplish are: acquire battery data (both from the BSM and other sensors), process this data (i.e. check if all parameters are within the SOA) and finally broadcast the battery's data to other boat systems through CAN and to the GUI when a computer is connected. These tasks should be run at a specific refresh rate and should thus be time-triggered. These main tasks should run in the order presented above. Otherwise, there is no new information to process or send, and so they must be run cyclically, one after the other. These tasks should respect their own refresh rate, and thus a higher priority will be attributed to their interruptions.

Other tasks of the BMS such as waking up from sleep, reacting to the voltage sensors (charger, motors, and boat on) and processing received CAN messages are less restrictive. Thus their interruptions can have a lower priority. Such tasks are event-triggered because they are triggered by a rising or falling edge on one of the MCU's digital pins.

As previously discussed, the BMS's SW runs on a Teensy 3.2. For this MCU, two programming ecosystems are available: Arduino and Mbed OS. While the former is cyclic based made for a wide variety of boards, the latter is specifically designed for ARM Cortex-M MCUs and supports deterministic, multithreaded, real-time software execution [28].

Considering the nature of the tasks presented, which don't require a deterministic, multithreaded software execution, and the fact that the Arduino ecosystem has been deeply optimized for the Teensy HW, the chosen SW architecture will be a cyclic one. Thus the Arduino ecosystem will be chosen. The Teensy programming will be done using Visual Studio Code with the PlatformIO extension in C/C++. Nevertheless, if in the future a real-time Operating System is needed, the SW can be adapted to use Mbed OS instead of the Arduino ecosystem, given that the chosen MCU already supports it.

In the following sections, we will start by presenting an overview of the SW and then describe in detail its most important parts.

4.4.1 Overview

The SW can be divided into the following main parts:

- Initialization;
- Time-based interruptions;
- Pin-change interruptions;
- Communication interruptions;
- BMS routine;
- Main loop;
- Sleep.

The general flowchart of the BMS's SW is presented in Figure 4.22. The SW starts by initializing all the subsystems and performing a few checks. If they fail, the BMS enters in a critical error state. Otherwise, the SW proceeds to the main loop where it stays unless an interruption happens or the

conditions to enter the sleep mode are satisfied. After performing the Interruption Service Routines (ISRs), it returns to the main loop. If the conditions to enter the sleep mode are assured, the SW places the Teensy in a low power state from which it can only exit with a wake-up interruption.

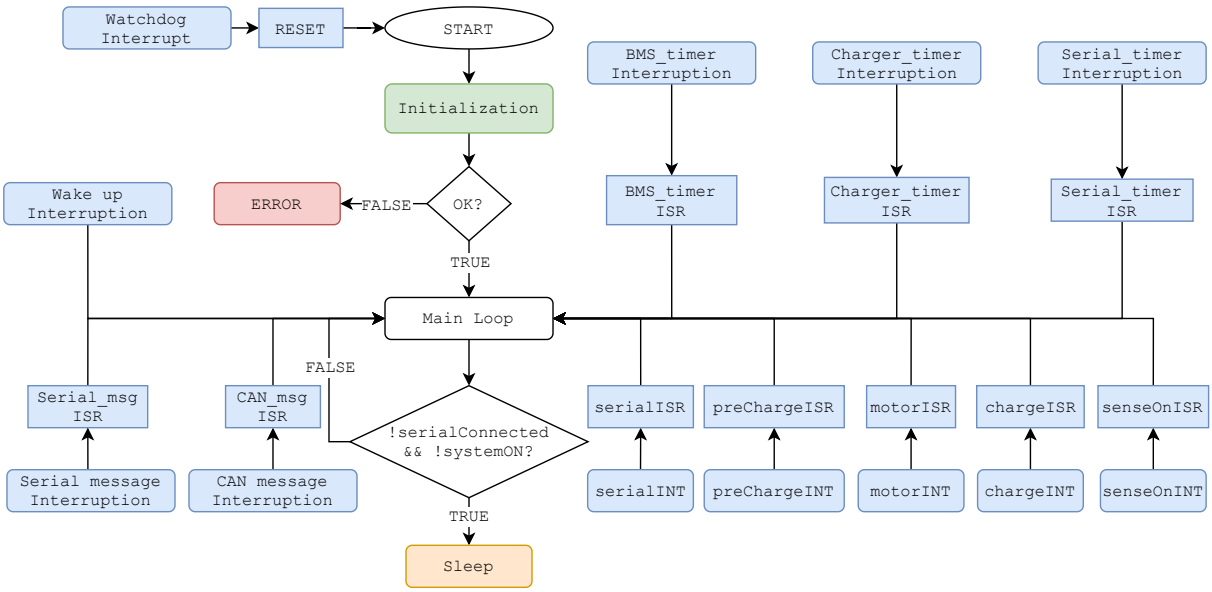


Figure 4.22: General BMS flowchart.

When the development of the BMS started, in V0, the example SW provided by Linear Technology (LT) was used to establish the communication between the BSM and the MCU. However, this SW is not fully compatible with the Teensy, and it had to be adapted. Furthermore, there were a few errors on the SW that prevented some features like the open-wire check to work and caused overall system crashes. The entire SW had to be reviewed, and fixes were implemented where necessary.

There were also problems related to the SPI mode needed to communicate with the BSM (CPHA=1 and CPOL=1), which expects the clock to idle at a logic high level and data on master out slave in line to be stable during the rising edge of the clock. The function that places the Teensy's SPI bus in this mode was not correctly configuring the bus for this. This problem was overcome by sending an empty transaction after configuring the bus, forcing it to comply with the chosen mode. Otherwise, the first message sent to the BSM was always ignored.

This LT's software example implemented a command-line interface that was capable of doing some of the BSM features. However, routines used to read OV, UV and thermal shutdown flags, as well as routines, to read overall battery voltage, BSM's internal voltages and temperature and GPIO pins 3 to 5 had to be added. On the following versions, this improved version of the library provided by LT was used as an interface to communicate with the BSM but the main code was completely re-written from a command-line interface to a full BMS SW that requires no user input to monitor the battery and is capable of sending the acquired data to the boat's CAN bus and PC based GUI.

4.4.2 Initialization

This part of the SW is responsible for initializing all the variables and peripherals needed for the BMS. It starts by initializing the serial port and the serial messages (i.e. initializing the static part of the different messages, such as size and message and data Identifiers (IDs)). Afterwards, the SW tries to load the user-configurable parameters from the Electrically-Erasable Programmable Read-Only Memory (EEPROM), initializes the data structures used to save the BSM's data, and finally initializes the communication with the BSM while also issuing the command to perform the BSM's self-checks.

Then the CAN Bus HW controller is configured for a $1\,\mathrm{Mbit\,s^{-1}}$ baudrate and, similarly to the serial messages, the CAN messages are also initialized. During this process the CAN Bus interruption is also configured (more on this on Section 4.4.6)

To finalize the start-up routine, the SW attaches the different interruptions, configures the Snooze Library drivers (used for sleep management) and configures the WDT.

The EEPROM loading routine is protected with a Cyclic Redundancy Check (CRC). If the check fails or no values are present on the EEPROM, default values are loaded and also saved to the EEPROM for future use. The occurrence is reported in the micro-SD card.

4.4.3 Time-based interruptions

Teensy provides a feature called `IntervalTimer`, which allows the user to configure a piece of code to run periodically within an ISR context. The priority of each timer can be configured so that in case of collisions, the most important one runs first. Out of the four available `IntervalTimers`, three are used on this BMS. The first, and most important one, `BMS_timer`, is responsible for signalizing to the main loop that it is time to acquire and process battery data. The second, `serial_timer`, is responsible for sending the serial messages to the GUI and the third one, `charger_timer`, is responsible for monitoring the battery current while charging.

The code called by these interruptions runs in an ISR context, and thus care should be taken not to spend too much time in their ISR. For this reason, the `BMS_timer` only sets a flag that is then processed in the main loop. In contrast, and given that they run relatively fast, `serial_timer` and `charger_timer` run their code directly within the ISR context.

By default, `BMS_timer` and `serial_timer` both run at $1\,\mathrm{Hz}$ while `charger_timer` runs once every $3\,\mathrm{s}$. However, `BMS_timer` can be configured using the GUI to have a refresh rate of up to $5\,\mathrm{Hz}$, the others have a fixed frequency that can only be changed by reprogramming the BMS.

Both `serial_timer` and `charger_timer` are disabled if the USB cable is not plugged in to the Teensy or the battery is not being charged, respectively.

4.4.4 Pin-change interruptions

Pin-change interruptions are interruption routines that run when a pin changes its state. Three types of triggers can be configured, `RISING`, `FALLING` and `CHANGE`. As the names suggest, the first two only run when the signal changes from low to high or high to low, respectively, whereas the latter runs in both cases. In this BMS a total of five pin-change interruptions were configured.

`serialINT` is responsible for keeping track of when the USB cable is plugged in so that the Teensy can start broadcasting the serial messages to the BMS GUI. To do so, a `CHANGE` trigger was attached

to the digital pin that is connected to Teensy's V_{USB} pin, as explained in Section 4.3.2. Within the ISR context, the pin state is checked if it is high, the USB has been connected, and thus the `serial_timer` is enabled. Otherwise, the timer is disabled.

`preChargeINT` is responsible for monitoring the output of the comparator mentioned in Section 4.3.5. In this case, a `RISING` trigger was used because we are only interested in catching the rising edge of the comparator's output. This ISR is responsible for turning on the motors, given that when triggered, it closes the motors' relay and opens the precharge relay, thus allowing the current to flow to the motor controllers.

Also responsible for turning on the motors is the `motorINT`. This interruption monitors the state of the killswitch with a `CHANGE` trigger attached to the motors' voltage sensor output, mentioned in Section 4.2.7. If the signal is high, the killswitch cord has been connected, and thus the pilot wants to activate the motors. In that case, the BMS checks if it is safe to turn on the motors, and if so, it closes the precharge relay, thus initializing the precharge of the motor controllers' capacitors. In contrast, if the signal is low, the killswitch cord has been removed, and thus the motors should be turned off. To do so, the BMS opens both the motors' and precharge's relays. In most situations, the precharge relay will already be open due to `preChargeINT`, but if the killswitch is removed before the precharge is completed, it will still be closed and thus, the need to open it here.

`chargeINT` is responsible for monitoring the charging port through the charger voltage sensor. This is done with a `RISING` trigger. When triggered, the BMS knows that a charger has been plugged in and checks if the battery can be charged. If so, the solar panels' relay is opened and the charger relay closed. It is crucial to open the solar panels' relay first since otherwise the maximum charge current can be exceeded, and the BMS has no way to limit neither the charger's nor the solar panels' current.

Finally, the last pin-change interruption is the `senseOnINT` which is responsible for identifying when the boat is turned off so that the BMS can enter its low power state. To do so, the voltage sensor attached to the emergency switch is used in conjunction with a `FALLING` trigger. When the interruption is triggered, a flag is set, signalizing that the boat is turned off. This flag is then processed in the main loop.

4.4.5 Serial communication

In order to communicate with the GUI, a message protocol had to be defined so that the BMS and the GUI speak the same language. This protocol is based on TSB's Serial Communication Library, which not only defines how messages should be formatted but also handles the received messages.

The structure that each message has to follow is described in Table 4.3. Each message starts with a start byte, $0\mathrm{xAC}$, followed by the device ID, which identifies who is sending the message, a message ID, which identifies the data that is being transmitted, the message length, the actual data to be transmitted and a checksum byte followed by an end byte, $0\mathrm{xAD}$. All fields are single-byte length except the data filed, which is limited to eight bytes. This is not a HW nor SW limitation but rather a team's choice so that serial and CAN messages can be compatible.

Table 4.3: TSB's Serial messages' format.

Start	Device ID	MSG ID	LEN	Data	Checksum	END
1 Byte	1 Byte	1 Byte	1 Byte	8 Bytes	1 Byte	1 Byte

The device and message IDs and the length fields are populated during the initialization phase of the code, given that they aren't changed during runtime. The start and end bytes, as well as the checksum, is populated by TSB's Serial Communication Library sending routine. The data to be sent is passed as an argument of this routine.

When a message is received in the USB serial, the Teensy has a special internal routine that automatically calls a function called `serialEvent`, if it exists. This function is defined inside TSB's Serial Library. It is responsible for receiving the message's bytes, computing the checksum (which is an XOR of all bytes except the start and end fields), verifying the message integrity, and if the computed checksum matches the one that came in the message, a flag is set, and the message is made available to the main loop for processing purposes.

The library also has a routine for sending messages. For this, a properly formatted and populated message should be passed to the function, which then computes the checksum and sends the message.

The BMS has a total of 15 serial messages that are sent, described in detail in Appendix A. These include all the information acquired by the BMS such as cell voltages, temperatures and currents. Two of these messages are special and are used to send the current state of the user-configurable parameters to the BMS GUI so that the GUI can display the currently defined values to the user. To set new values, the GUI sends a message similar to the ones sent by the BMS with the new values that should be configured.

4.4.6 CAN communication

CAN messages work differently from serial messages in the sense that they have a defined standard. The standard that we use in the PCBs that we build is the CAN 2.0A, which implies that messages have an 11-bit ID. Some of the systems that we use on the boat, such as the motor controllers, use CAN 2.0B, which uses extended 29-bit IDs, but both can coexist in the same bus. Both standards establish that the data field should be at most eight bytes long. That is why TSB's Serial Library also uses this configuration, as seen in Section 4.4.5. Figure 4.23 shows the different fields that compose a CAN 2.0A message, also known as standard CAN message. In this example, the message only has one data byte.

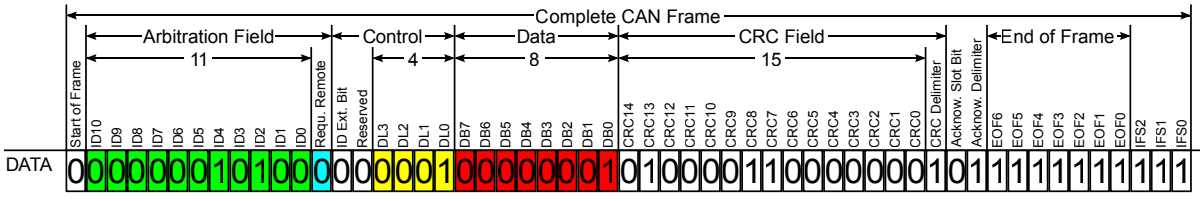


Figure 4.23: Visual representation of a standard CAN message frame, doesn't take bit stuffing into account.

From the different fields shown, three of them are most relevant for us since they are the ones

that should be populated so that a message can be sent. The **Arbitration Field**, highlighted in green, contains the message ID. The lower the ID, the higher the message's priority. The **Control Field** contains six bits, but given that we are using standard CAN messages, which is the library's default, we don't need to set the ID extended bit, which is set to zero by default. The reserved bit should always be set to zero, but the library takes care of this. Therefore, we only need to worry about the bits highlighted in yellow, which are the ones that define the Data Lenght Code. The last field we need to populate is the **Data Field**, highlighted in red, which is where the actual message data goes.

Like for the serial messages, a library was also developed for the CAN communication. This library is now used in all MCUs that go into the boat, but it was first developed for the BMS. Besides handling the communication, this library defines a few rules that should be followed by every MCU that is used. One of such rules is that the **Arbitration Field** is composed of three smaller fields that help identify who is sending the message and what data is in it. This structure is represented in Table 4.4.

Table 4.4: TSB's CAN messages' Arbitration Field.

Priority Bits	Data ID	Source ID
2 bits	5 bits	4 bits

The first two bits are reserved for setting the message priority, the following five bits are used to identify which message is being sent, and the last 4 bits define which device sent the message. The Data ID in conjunction with the Source ID identify the message being sent. The BMS's Source ID is $0x01$.

Like with serial messages, the static arbitration and control fields are all populated during the initialization phase.

Besides using TSB's CAN library, `FlexCAN_T4` library[18] was also used. This library provides an interface for receiving and transmitting CAN messages without having to worry about the underlying communication with the MCU's NXP FlexCAN controller core. TSB's CAN Library extends `FlexCAN_T4` `CANListener` class so that received messages are processed by TSB's own library.

Every time a message is received `FlexCAN_T4` raises an interruption that calls the handler defined in our library, which process each message according to its ID and decodes its content saving the data in an appropriate struct. Each device that sends messages has its own struct with fields for every data it sends. Hardware message filters can be used so that this interruption is only raised when certain messages are received, but that functionality is currently not used. It can be helpful, though, if the Teensy is not able to perform its main tasks due to receiving too many CAN messages, but that is not a problem at the moment.

TSB's CAN library also contains an array of unsigned longs with a length equal to the maximum number of possible source IDs. Each time a device sends a message, the current time is stored in this array.

Routines to check if the data contained in the devices' structs is still valid by comparing the current time with the time stored in the previously mentioned array and reset it if it isn't were also implemented in this library. A special function that converts each device's strut to a JavaScript Object Notation (JSON)

[18] FlexCAN_T4 Library - Accessed on 17/10/2021.

encoded string has also been implemented so that real-time telemetry can be sent to TSB's Message Queuing Telemetry Transport server[19], which allows the team to monitor the boat from shore.

Lastly, in order to send messages `FlexCAN_T4`'s `write` function is used. The only thing needed is to pass the message to be sent to the function, which then takes care of placing it on the `FlexCAN_T4`'s outgoing mailbox. The BMS sends a total of 13 messages, described in detail in Appendix A, which are identical to the ones sent to the serial port in what concerns its data. The configuration messages used in the serial communication don't exist here, given that the BMS's user-configurable parameters were not projected to be changed over CAN. However, this feature can easily be implemented if needed.

4.4.7 BMS routine

This is the main function of the BMS. It is responsible for communicating with the BSM, acquiring its data and acquiring data from the rest of the devices on the board. The acquired data is then processed to verify if all parameters are within the SOA. The main loop calls this function after `BMS_timer` has set the respective flag.

Figure 4.24 represents the flowchart of the BMS routine function. Firstly the BSM configuration is written to it[20]. Then the data from the BSM and the rest of the Teensy inputs is acquired. If CRC errors are detected while communicating with the BSM, the command is sent again, and the situation is reported in the micro-SD card.

If or when there are no CRC errors, the SW proceeds to compare the acquired values with the SOA thresholds. If everything is within the SOA, the system checks if the battery is full, if it is balancing, if it needs to be balanced and if charging is allowed. The rightmost part of the flowchart was replaced by the feature that lets the pilot control the solar panels through the CAN bus, but it was kept here given that it could be re-implemented. The battery box fans are also controlled here. If either any of the temperatures measured by the BMS is higher than $30\,^{\circ}\mathrm{C}$ or the modulus of the battery current is higher than $2\,\mathrm{A}$, the fans are turned on. The fans are then turned off if the temperature falls below $27\,^{\circ}\mathrm{C}$ or the current below $1.5\,\mathrm{A}$, this hysteresis prevents the fans from being constantly turned on and off.

If, on the other hand, the values are outside the SOA, depending on its severity, actions should be taken. The checked parameters are over and under voltage, charge and discharge current and temperatures. In an OV situation, both charging sources are disconnected from the battery, and a flag is set to prevent charging from happening. Besides, the balancing resistors are turned on on the cells that present an OV condition.

In the case of UV, the motors' relay is immediately open, given that this is the major load of the system. Moreover, the solar relay is closed to allow the battery to be charged. Normally, the UV condition is cleared a few seconds after the motor relay is opened due to the decrease in current being pulled from the battery. To re-enable the motors, the pilot has to disconnect the killswitch and connect it again. After an UV the boat should only move slowly, if possible only using solar power since the battery is almost depleted.

In the case of temperature, only over-temperature is currently being checked. Given the nature of our prototype, we will never be sailing with temperatures low enough to cause damage to the battery. However, this can easily be modified. If an over-temperature is detected, the pilot receives a pop-up with

[19] This functionality is used by another PCB on the boat, but the underlying function required to convert the structs' data to a JSON string was developed throughout this work.

a warning in the dashboard, and he decides what action to take based on where the over-temperature is. If the temperature continues to increase and reaches $5\,^{\circ}\mathrm{C}$ above the set threshold, the BMS will automatically open the motors' relay, given that typically high temperatures are due to drawing too much current for too long.

Regarding over-current, there are two situations to be checked: over-current while charging or while discharging. Each situation requires a different action. In the case in which the battery is being charged, both charging sources will be disconnected. In contrast, if the battery is being discharged, the motors' relay will be opened. To clear this fault and re-enable the motors, the same procedure used in the UV situation should be followed.

In all cases mentioned above, a bit is set on the warnings message to signal the respective condition to the other systems through the CAN bus and to the GUI over serial. The BMS buzzer is also activated. This way, the pilot will always be fully aware of what is going on with the battery and can, ultimately, decide what the safer procedure to follow is.

The relay that powers the boat's low power electronic is currently not controlled by the BMS but one of the spare relay control outputs can be used for this. The reason why, at the moment, the BMS doesn't do it is related to the boat's hydrofoils. The electronics relay powers them, and cutting their power mid-flight can have catastrophic consequences.

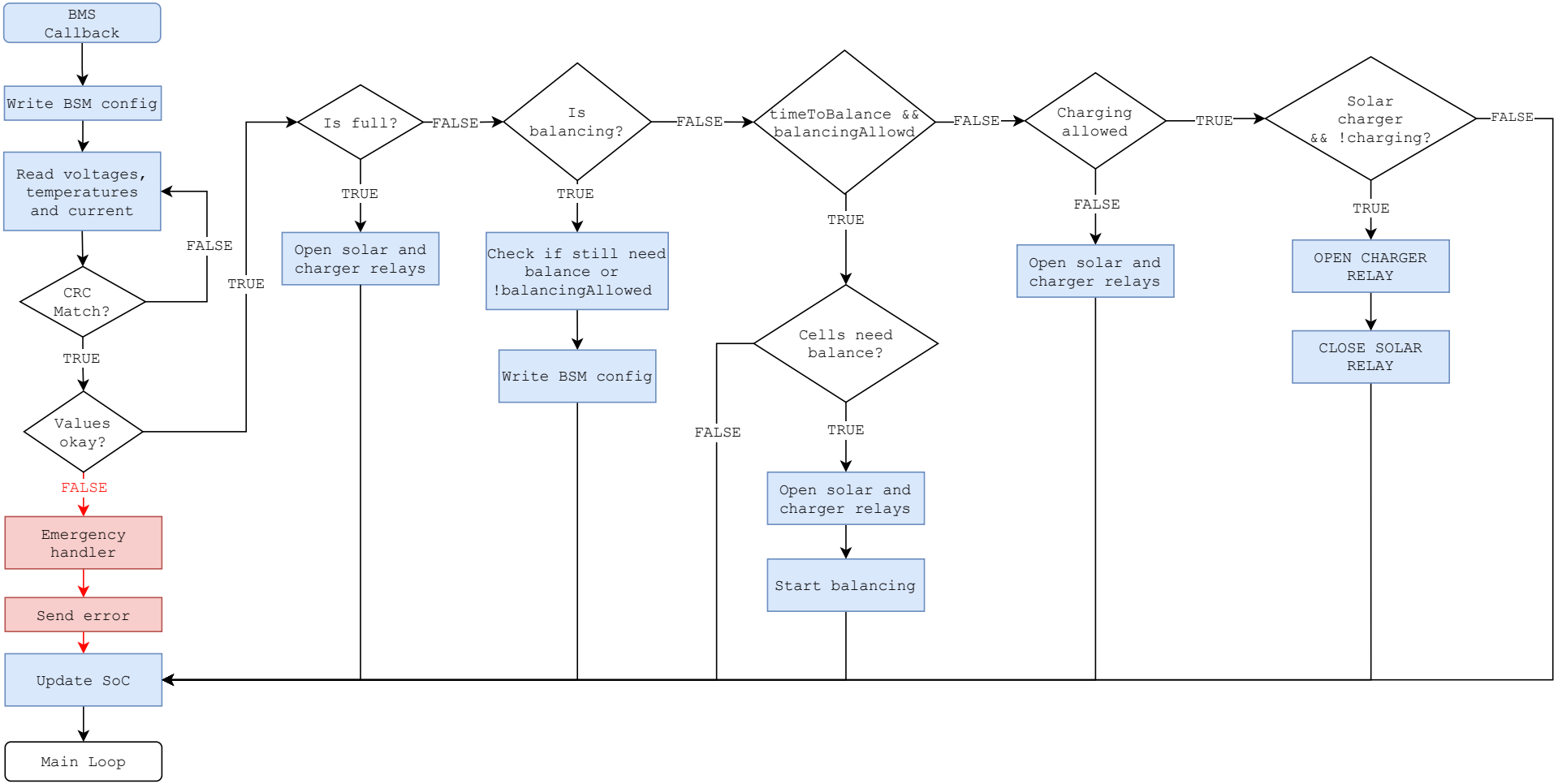


Figure 4.24: BMS routine flowchart.

4.4.8 Main loop

The main loop is where the BMS is at when it is not sleeping nor processing ISRs. Here is where valid messages received from the serial port or CAN bus that perform changes on the BMS are processed. When a valid serial message is received, it is identified by its ID, and if it corresponds to one of the two configuration messages, the user-configurable parameters are updated, and the new ones are saved to the EEPROM so that they can be loaded in future executions. At the moment, the only message received from the CAN bus that expects action from the BMS is the command used to toggle the solar panels' relay. This message is sent from the pilot dashboard, and it is mainly used when working on the boat ashore. A failsafe for this command prevents the boat from drawing too much current with the relay open. As soon as more than $5\,\mathrm{A}$ are drawn from the battery, the relay is automatically closed to prevent unintentional battery drain.

The main loop also monitors the state of the different flags that are activated by either time-based or pin-change interruptions. One of these flags is the one used to inform that it is time to acquire and process BSM data. If the flag is activated, the function `BMS_routine` is called. When this function returns, both serial and CAN messages are updated, and CAN messages are sent.

Another flag processed by the main loop is the one that tells that the boat is off and the USB cable is not connected, meaning that it is time to place the BMS in a low power state. This is done with the help of the Snooze Library.

The last task of the main loop is to call the TSB's CAN Library routine to check the validity of each device's data struct so that only valid data is used by the BMS.

4.4.9 Sleep

To make use of the different sleep modes provided by Teensy's NXP ARM core, as stated before, Snooze Library was used. This library is able to wake up the Teensy with a few different drivers like the Teensy's Kinetics touch module, digital pins or even predefined timers.

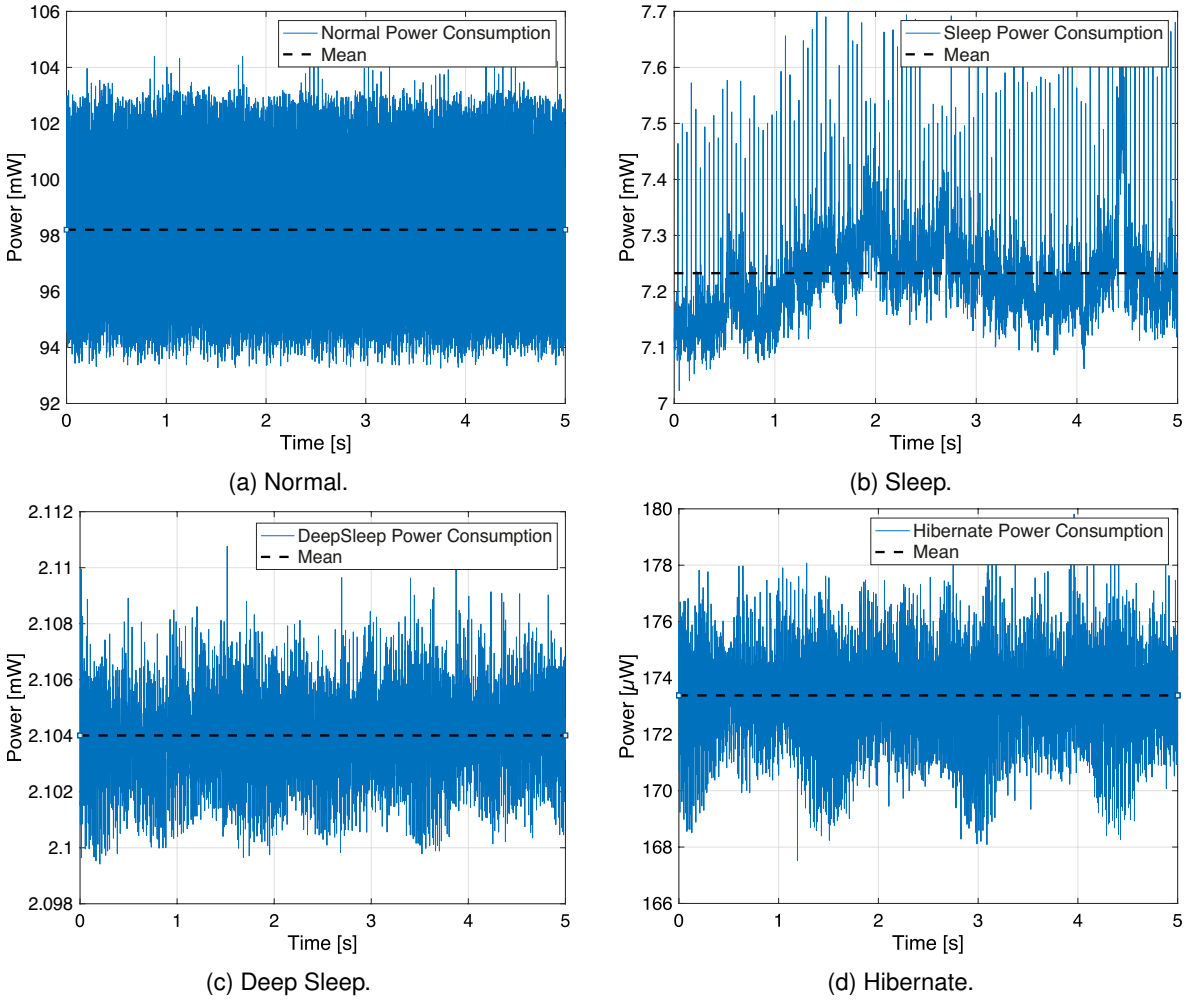
In this case, both the digital and alarm drivers are used. The digital driver is configured to wake up the Teensy if the boat is turned on or if the USB is plugged in, while the alarm driver is configured to wake up the Teensy every 10 minutes in order to check the battery. While there is a timer driver that uses a low power timer instead of the RTC, this driver is limited to one-second intervals on this Teensy, and so the alarm driver has to be used.

The library provides different power modes, but sleep, deep sleep and hibernate are the most interesting for this application. The differences between the three are not well documented, but tests performed and shown in Figure 4.25 allow us to conclude that in its normal mode, the Teensy consumes $98.2\,\mathrm{mW}$, in sleep $7.23\,\mathrm{mW}$, in deep sleep $2.10\,\mathrm{mW}$ and in hibernate $173.4\,\mu\mathrm{W}$ and so hibernate is used, given that it offers the highest power savings[21]. These tests were performed using a bare Teensy 3.2 without any external devices attached to it and directly powered through its $3.3\,\mathrm{V}$ input, thus bypassing the internal 5 to $3.3\,\mathrm{V}$ LDO regulator, replicating V2's implementation.

It is important to mention that the different power modes impose limitations to what functionalities of

[20] Writing the configuration to the BSM is a mandatory step every time a command is sent unless we are absolutely sure that the last command was sent less than two seconds ago. Such step is required because the BSM resets its configuration registers bytes `CFGR0 to CFGR5` if no valid command is received for more than two seconds and the discharge timer is not in use, which is the case, see Section 4.2.3.

the MCU are available during sleeping as well as how the MCU can be brought back to its normal state. These limitations are explained in more detail in [29].



(a) Normal. (b) Sleep.

(c) Deep Sleep. (d) Hibernate.

Figure 4.25: Teensy's power consumption during normal operation and under the different sleeping modes.

Upon waking up, the driver that wakes up the Teensy can be checked. In the case USB triggered the wake up process, the `serial_timer` is enabled. If, on the other hand, the Teensy wakes up because the boat was turned on, a flag is set to identify this condition and prevents the Teensy from entering sleep again.

4.4.10 BMS GUI

The BMS GUI was developed in parallel with the BMS development by another member of the team using the Qt framework[22]. It can run both in macOS and Windows. It is responsible for showing the BMS

[21] All power measurements were acquired with two Agilent 34410A multimeters in TagusPark's Electronics' Laboratory. One was used to measure the supply voltage, while the other measured the supply current. Both measurements were synchronized using a function generator acting as an external trigger. Five thousand samples were acquired and logged with the help of Keysight's BenchVue software. The data was then exported to a CSV and processed using Matlab.

data to the end-user and the competition's inspection team, simply and intuitively. The GUI is also used to configure the BMS's user-configurable parameters. The GUI's front page is shown in Figure 4.26.

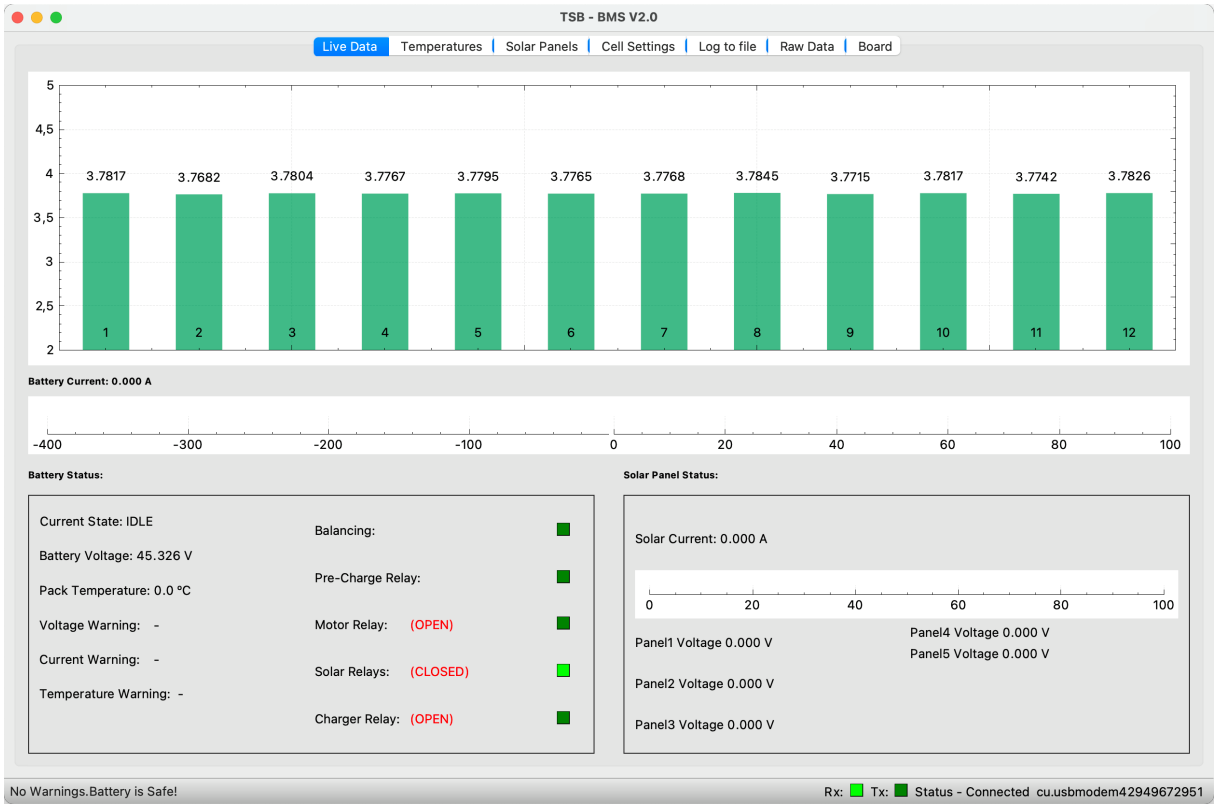


Figure 4.26: BMS's GUI front page.

Figure 4.27 shows an image of the GUI where all the temperatures measured by the BMS can be seen.

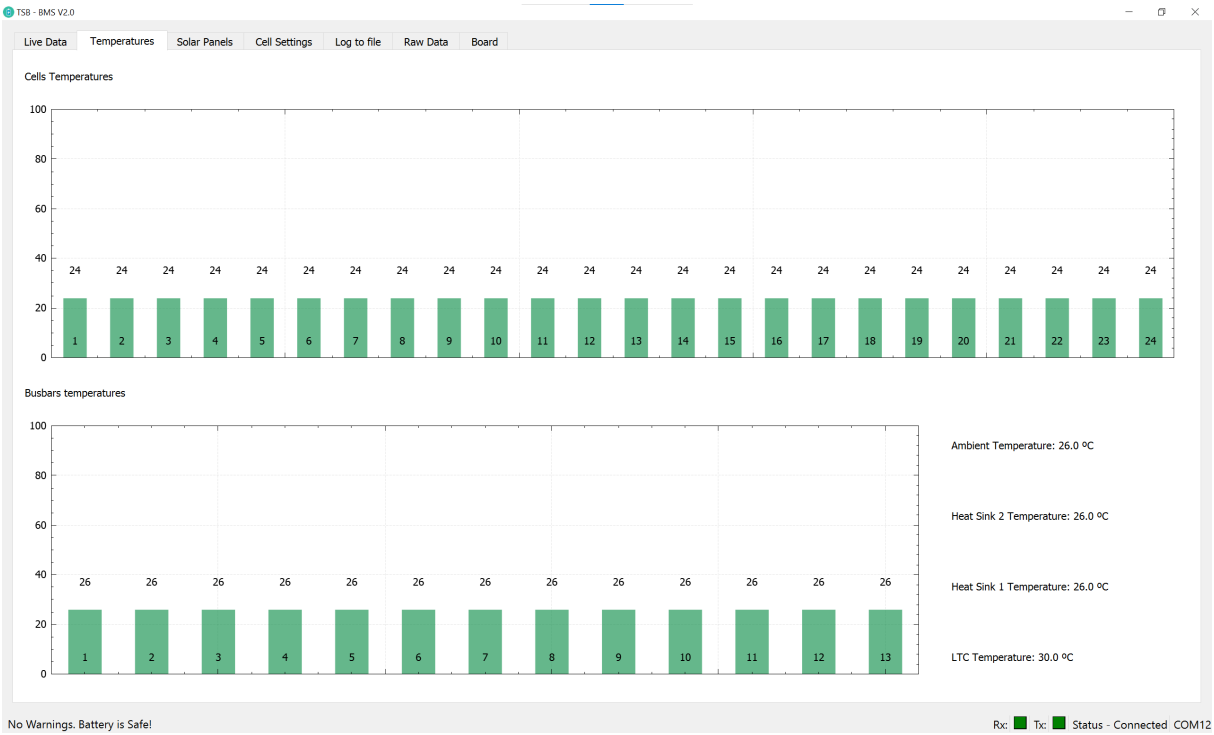


Figure 4.27: Temperatures acquired by the BMS and displayed in the GUI.

[22] Qt Framework - Accessed on 17/10/2021.

5

System Deployment and Validation

5.1 System Deployment

This section serves the purpose of describing the deployment of the system explained in the previous chapters. The first step in order to be able to test the developed BMS consists in the assembly of the battery. To do so, we first started by attaching one thermistor to each cell, like shown in Figure 4.19, and then the cells were grouped in pairs which will, later on, compose each cell aggregate. These groups were chosen with the help of the matching sheet supplied by the cells' manufacturer, which considers parameters such as the cells' internal resistance and capacity to form the best possible battery pack, both in terms of capacity and battery internal resistance.

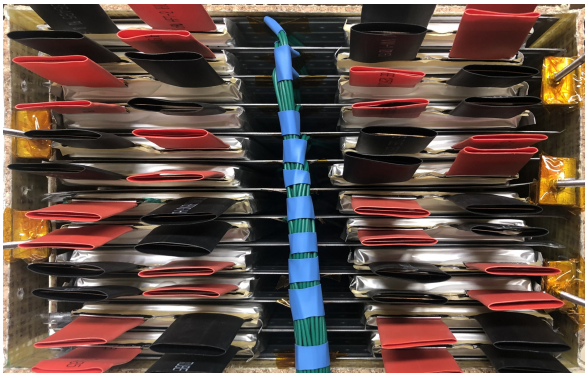
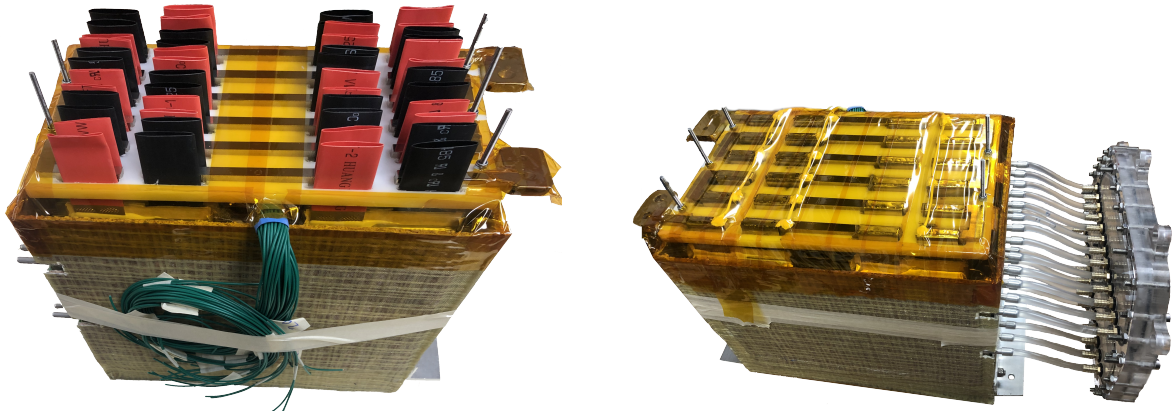


Figure 5.1: Cells, cooling plates and thermistors' cables top-view.

The cell aggregates were then sorted by their internal resistance. Cell aggregates with higher internal resistance were placed where the cooling is better. In this case, the corners of the battery enclosure were used, given that in the corners the outer cooling plate doesn't have a cell on the outside. See Figure 5.1. Afterwards, the top plate and bus bars were placed on the battery, Figure 5.2a. Figure 5.2b shows the battery after the soldering process and with the cooling distribution plate already attached to the cooling plates.



(a) (b)

Figure 5.2: Battery ready to be soldered (a) and battery after soldering with the water cooling distribution plate (b).

Regarding the BMS itself, even though the PCB manufacturing was outsourced, the assembly process was entirely done in TagusPark's PCB laboratory using the different machines available, such as the stencil positioner machine and the reflow oven. Figure 5.3 shows the PCB after the solder paste was applied and the components were positioned on the board.

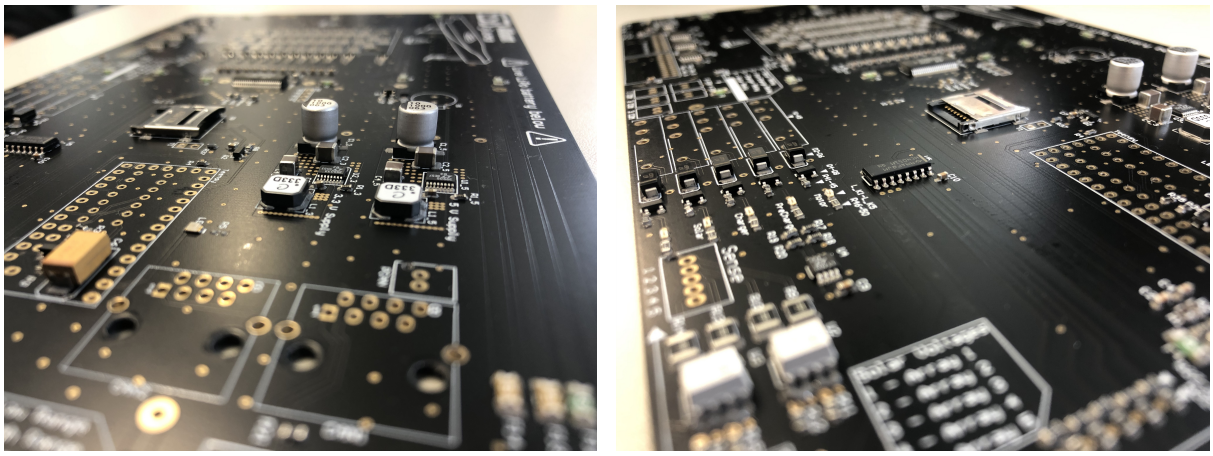
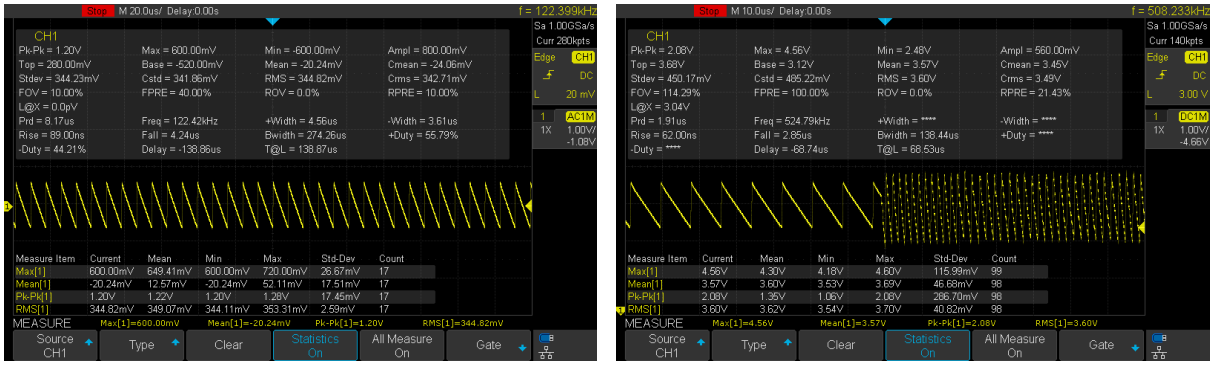


Figure 5.3: BMS PCB after applying solder paste and components, before reflowing.

After the assembly was completed, we connected the BMS to the computer to check the cells voltages with the help of the GUI, but we were surprised by abnormal cell voltage readings. After a few days of debugging, we concluded that one of the BSM's voltage references, more precisely, V_{REF1}, was not providing the correct voltage to the BSM's ADCs. Figure 5.4a shows V_{REF1} signal measured with an oscilloscope in AC, which was measuring $1.2\,\mathrm{V}$ peak-peak which is clearly not supposed to happen. Figure 5.4b shows the same signal, but this time in DC. We can see that the voltage is not stable like it is supposed to be.

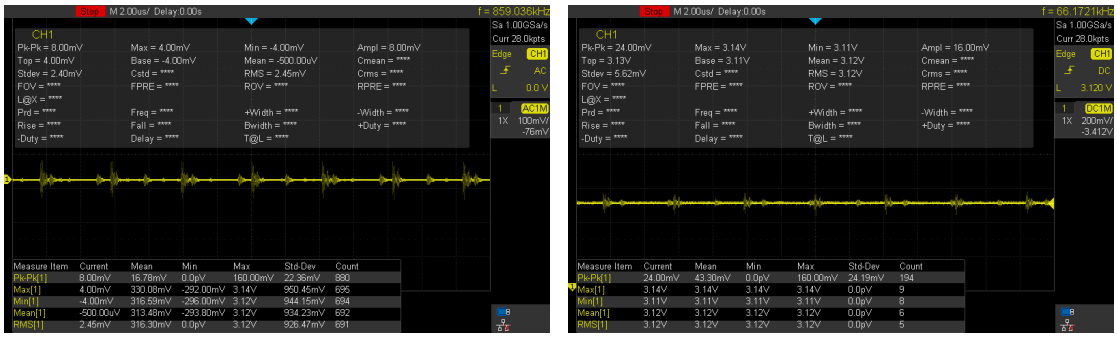


(a) AC coupling.

(b) DC coupling.

Figure 5.4: V_{REF1} signal measured with an oscilloscope.

We concluded that the problem was due to a poorly soldered bypass capacitor. After fixing this problem, V_{REF1} signal was within the specked 3.1 to $3.3\,\mathrm{V}$, like shown in Figure 5.5b and the ripple was reduced to $8\,\mathrm{mV}$ as seen in Figure 5.5a.



(a) AC coupling. (b) DC coupling.

Figure 5.5: V_{REF1} signal after correctly soldering the bypass capacitor.

A few weeks after the battery assembly was completed, we identified that one of the cell aggregates was presenting an abnormal self-discharge rate, causing the battery to become constantly unbalanced. At first, we suspected that the BMS could be the problem, but after replacing it with V1, which was not presenting such a problem, we concluded that one of the cell aggregates was faulty and had to be replaced. Due to the nature of the assembly, this was a complicated process, but after an entire day of work, we were able to replace the damaged aggregate with a new one, and the battery was ready to be soldered again. After inspecting the damaged cell, which was inflated, we concluded that during the soldering process, a spark may have penetrated the cell's aluminium foil and damaged it.

5.2 System Validation

This section serves the purpose of detailing the tests performed on the developed BMS in order to validate that it serves its purpose as well as it respects the predefined requirements. The results presented in this chapter were either obtained using the data that the BMS outputs to the boat's CAN bus or acquired in TagusPark's electronics laboratory using the different measuring equipment available.

With the exception of the power consumption section, where data from V1 is also shown so that a comparison between both versions is possible, the remaining results presented here are related to V2, given that this is the final version which all the requirements. The different tests will be presented from the most to the least important in what concern the impact the tested functionality has on the boat's performance.

5.2.1 Balancing

Having a properly balanced battery is critical in order to be able to fully utilize its energy. Otherwise, we will always be limited by the highest or lowest cell voltage depending on whether the battery is charging or discharging. In fact, in the worst-case scenario, if 11 cell aggregates are at their maximum voltage while the remaining one is at its minimum voltage, the battery cannot be used even though it has more than $90\,\%$ of its energy.

For this test, a balancing threshold of $20\,\mathrm{mV}$ was set, meaning that every cell aggregate that is more than $20\,\mathrm{mV}$ above the minimum cell aggregate voltage needs to be balanced. Figure 5.6 shows the voltage of each cell aggregate as well as the standard deviation between the 12 voltages. In this case,

cell aggregate 4 is the one with the lowest voltage, and thus, every aggregate at least $20\,\mathrm{mV}$ above its voltage should be balanced.

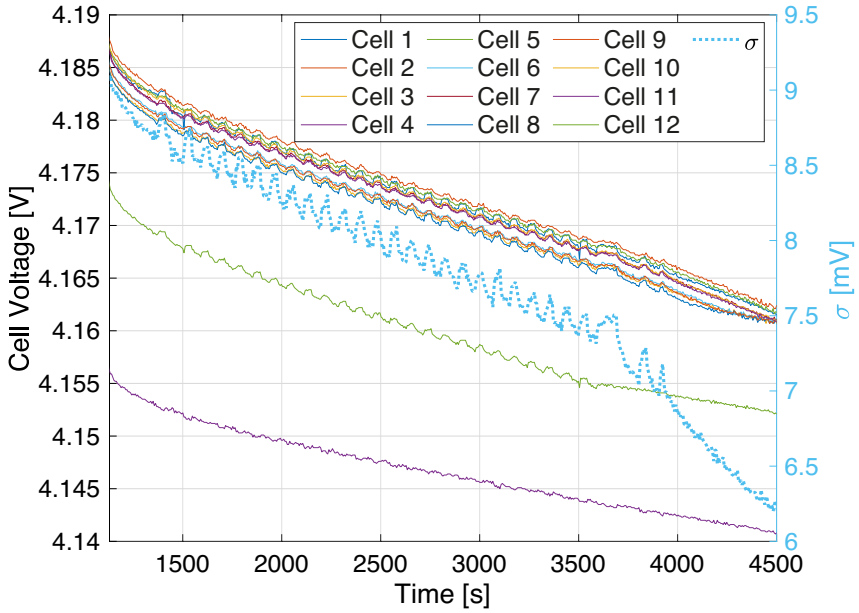


Figure 5.6: Cell voltages and their standard deviation while balancing.

Figure 5.7 shows the BMS's GUI while the battery is being balanced. In this case, the image was taken after cell aggregate 12 was already within the threshold (around second 3600 in Figure 5.6). The red colour on the bars graph denotes that the cell aggregate is being balanced.

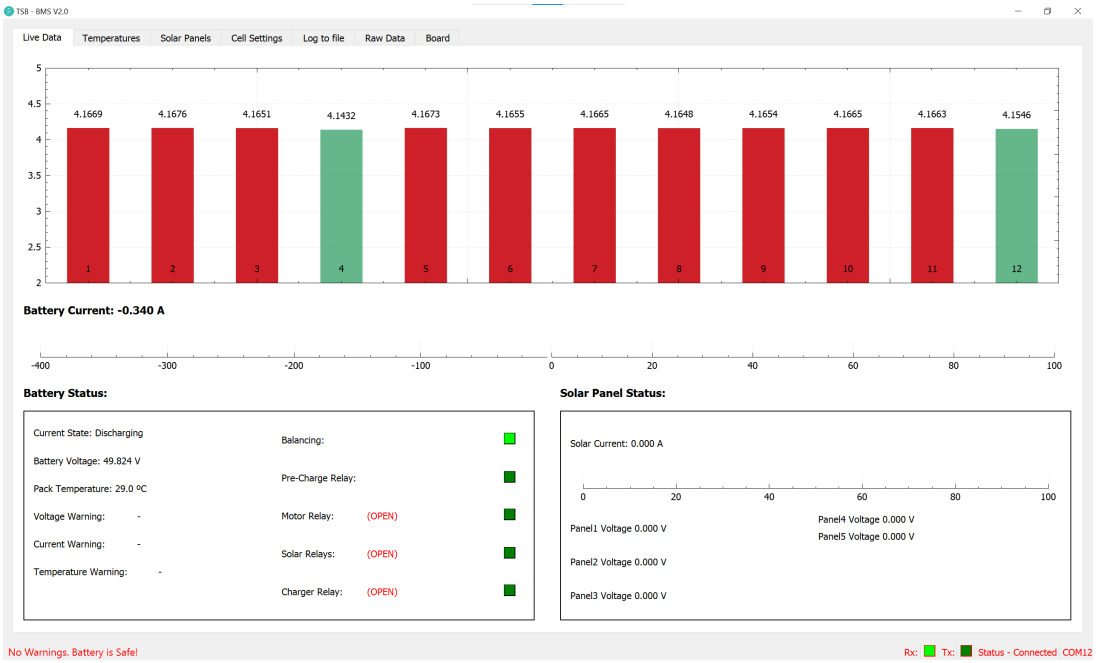


Figure 5.7: GUI while battery is being balanced.

As Figure 5.6 shows, the standard deviation between the voltage of the 12 aggregates decreased from around $9\,\mathrm{mV}$ to $6.3\,\mathrm{mV}$ which shows that, overall, the battery is more balanced at the end of the $75\,\mathrm{min}$ period. Due to the time it takes, the balancing procedure was manually stopped before all cells were within the specified threshold. Nevertheless, the acquired data already proves that balancing is working as expected. Another note should be made to explain why the voltage of the lowest cell aggregate is also decreasing during this process. This is due to the battery being in use and powering other peripherals, like the battery box fans, while balancing is taking place.

5.2.2 Battery capacity

In order to validate the battery capacity, a full discharge test was performed on the battery at a 1C discharge rate. This test was performed with the help of a Delta Elektronika $15\,\mathrm{kW}$ bi-directional power supply, which can not only source but also sink current.

Figure 5.8 shows the results of this test which took $58\,\mathrm{min}$. The blue line represents the energy discharged from the battery measured with Isabellenhütte's IVT-S smart battery shunt, while the red line shows the lowest cell aggregate voltage. The stop criteria for the test is the moment when the lowest cell aggregate reaches $3\,\mathrm{V}$. The test measured a battery capacity of $1493\,\mathrm{Wh}$ which is just $2\,\mathrm{Wh}$ above its design capacity. Given that the battery was not fully balanced and thus not fully charged, the real capacity is in practice a bit higher and more close to the $1515\,\mathrm{Wh}$ measured by the competition's technical inspection team during the 2021 Monaco Energy Boat Challenge. Nonetheless, the obtained capacity is still valid for proving that the battery does not exceed the $1500\,\mathrm{Wh}$ imposed by the regulations given that the battery, at the start of the test, can be considered fully charged and balanced according to [15].

The difference between the battery's design and real capacity is imputable to cells having more capacity than rated while they are new. Manufacturers do this so that the battery's real capacity is within $80\,\%$ of its rated capacity after the cycle life has been exceeded.

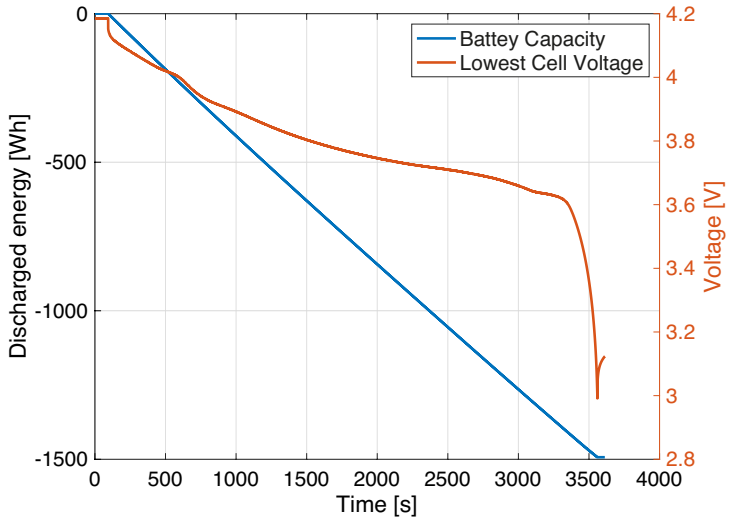


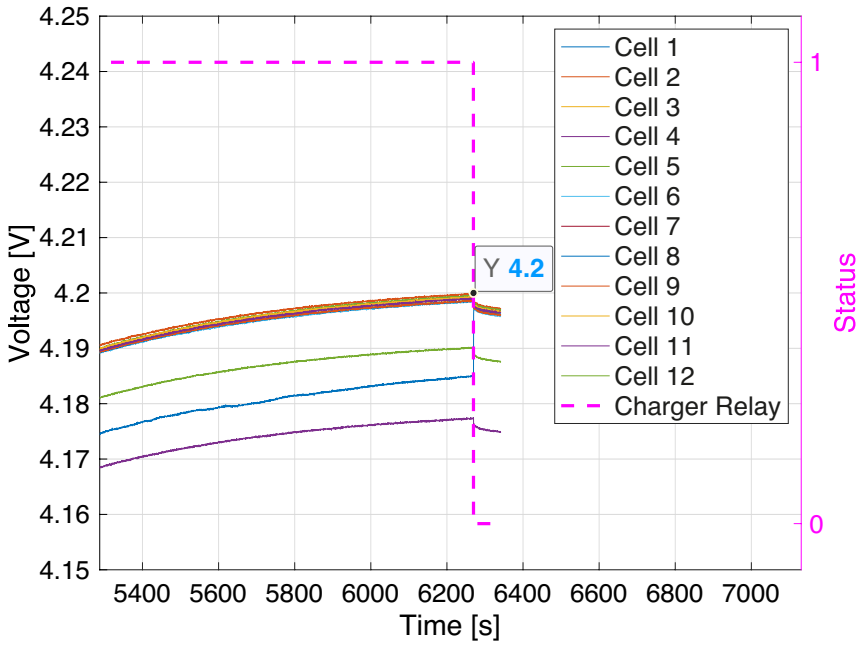
Figure 5.8: SR-03 battery capacity test.

5.2.3 Safety features

In this section, the performance of the BMS's various safety mechanisms were tested in order to validate that the BMS is able to keep the battery within its SOA. The first test performed was to verify whether the BMS is able to not only identify but also mitigate both OV and UV situations. Alongside over-current, OV is amongst the most dangerous scenarios for LiPo batteries given that it will lead to a fire if not stopped, and thus a BMS must be able to stop such condition promptly.

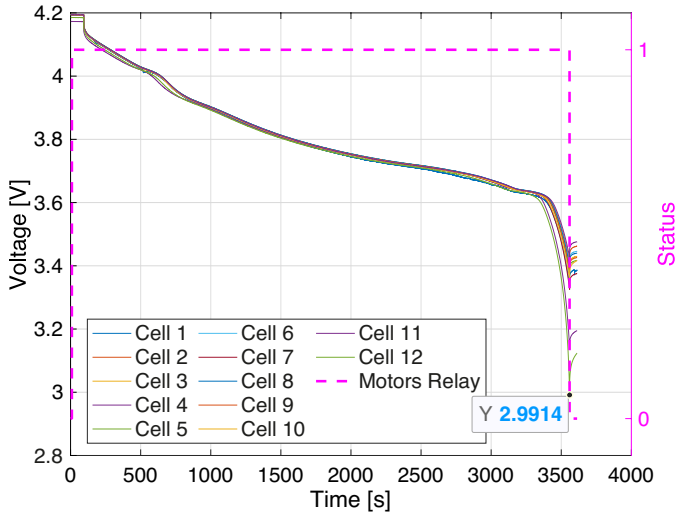
Figure 5.9 shows the BMS's action in both OV and UV situations. While in Figure 5.9a the battery is being charged and thus the voltage of each cell is increasing leading to an OV, in Figure 5.9b a battery discharge test was carried out and thus an UV was detected.

For this test the OV threshold was set to $4.2\,\mathrm{V}$ while the UV threshold was set to $3.0\,\mathrm{V}$. As it can be seen in Figure 5.9a, the BMS opens the charger relay as soon as it detects that one of the cells has reached the defined threshold. The dashed line on the graph identifies this behaviour. If the solar panels were charging the battery, the solar panels' relay would be opened instead. Similarly, in Figure 5.9b, the BMS opens the motors' relay as soon as it detects that one of the cells reaches the defined threshold. Upon opening the relay, the cell voltage increases due to the considerable decrease in current being drawn from the battery.



(a)

Figure 5.9: BMSSPV V2 over-voltage (a) and under-voltage (b) protections.

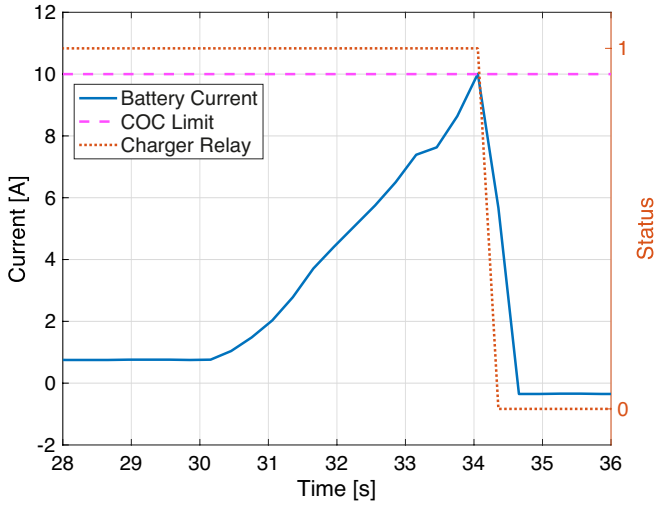


(b)

Figure 5.9: BMSSPV V2 over-voltage (a) and under-voltage (b) protections (cont.).

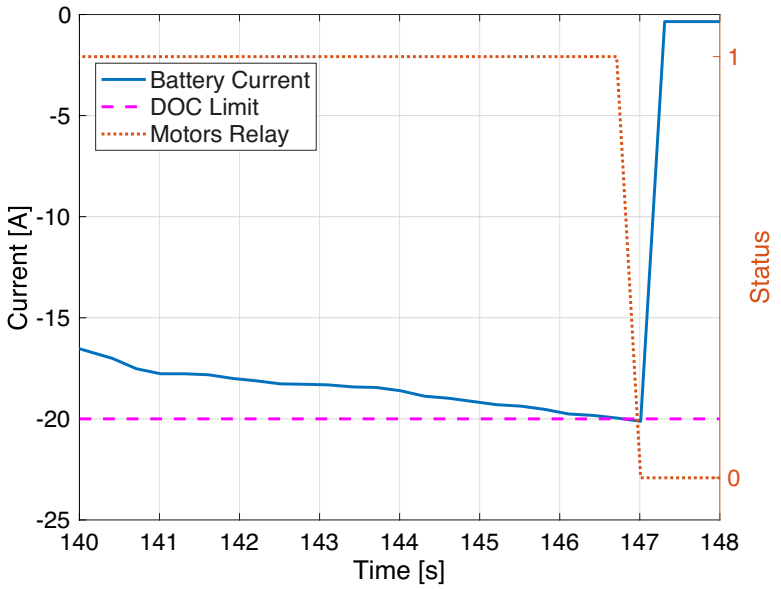
As stated above, over-current is also very dangerous in LiPo batteries. Figure 5.10 shows the protections of the developed BMS against two different types of over-current. Over-current while charging is shown in Figure 5.10a while over-current while discharging is shown in Figure 5.10b. In this test the default over-current thresholds were intentionally reduced using the BMS's GUI to 10 and $-20\,\mathrm{A}$ for charging and discharging respectively. These thresholds can be identified in the graphics by the dashed line.

As shown in Figure 5.10a, as soon as the battery current reaches $10\,\mathrm{A}$, the BMS immediately opens the charger relay, as the dotted line shows. On the other hand, Figure 5.10b, shows that once the battery current is lower than $-20\,\mathrm{A}$ the BMS opens the motors' relay. In both situations the over-current condition is stopped.



(a)

Figure 5.10: BMSSPV V2 charge (a) and discharge (b) over-current protections.



(b)

Figure 5.10: BMSSPV V2 charge (a) and discharge (b) over-current protections (cont.).

In order to test the over-temperature protection, two different types of over-temperature situations were tested. The first test consisted in drawing a high current from the battery with the cooling system turned off and a lower over-temperature threshold. The second test was an over-temperature in the balancing resistors.

As far as cell temperatures are concerned, there is nothing the BMS can do to cool them down, and so, as stated in Section 4.4.7, a warning is issued, and a pop-up is shown to the pilot. If no action is taken and the temperature reaches $5\,^{\circ}\mathrm{C}$ above the threshold, the motors' relay is opened, which decreases the current being pulled from the battery and thus the cells' temperature.

Figure 5.11 shows the current being pulled from the battery as well as the maximum temperature registered by the battery thermistors. Both bus bars and cells' body thermistors were considered. The right axis represents the status of the motors' relay as well as the BMS's over-temperature warning. For this test the battery over-temperature threshold was set to $30\,^{\circ}\mathrm{C}$ using the BMS's GUI. As it can be seen after drawing $100\,\mathrm{A}$ from the battery, it reached the over-temperature threshold, and the respective warning flag was activated. Given that the current continued to increase and with it the battery's temperature, when it reached $35\,^{\circ}\mathrm{C}$ the motors' relay was opened to prevent the battery from heating even more.

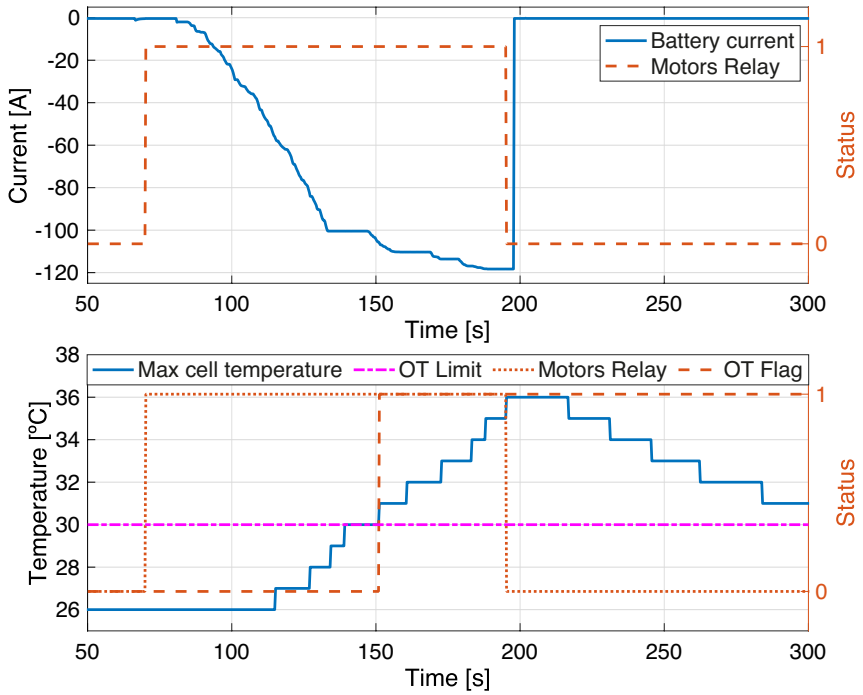


Figure 5.11: Battery over-temperature while drawing high currents without cooling.

Figure 5.12 shows the temperature measured by both thermistors placed below the balancing resistors' heatsink and also whether the balancing is active or not (dashed line). The maximum acceptable temperature for the resistors is set to $70\,^{\circ}\mathrm{C}$. As shown, if one of the heatsink's thermistors reaches $70\,^{\circ}\mathrm{C}$, the balancing process is paused until the temperature is below $60\,^{\circ}\mathrm{C}$. Then, the balancing is turned on again. This $10\,^{\circ}\mathrm{C}$ hysteresis allows the resistors to cool down properly before turning the balancing on again. Otherwise, the balancing would be turning off and on constantly.

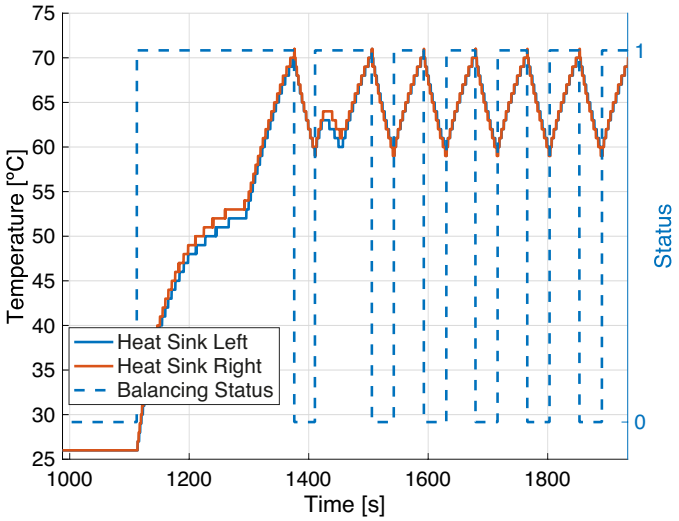


Figure 5.12: Over-temperature while balancing.

5.2.4 Precharge

Another test performed on the BMS is the one shown in Figure 5.13. This test shows the proper functioning of the implemented precharge circuit. Figure 5.13a shows the voltage at the motor controllers' input, while Figure 5.13b shows the battery current while the precharge is happening. In both figures, the precharge and motors' relays status is shown. The first marker on both figures shows the moment when the precharge relay is closed, leading to an initial current spike of $0.78\,\mathrm{A}$. After this, the current slowly decreases as the load's voltage gets closer to the source voltage. When the voltage is $90\,\%$ below the source voltage, the motors' relay is closed (second marker), and the precharge relay is opened, thus allowing all the current to flow to the load.

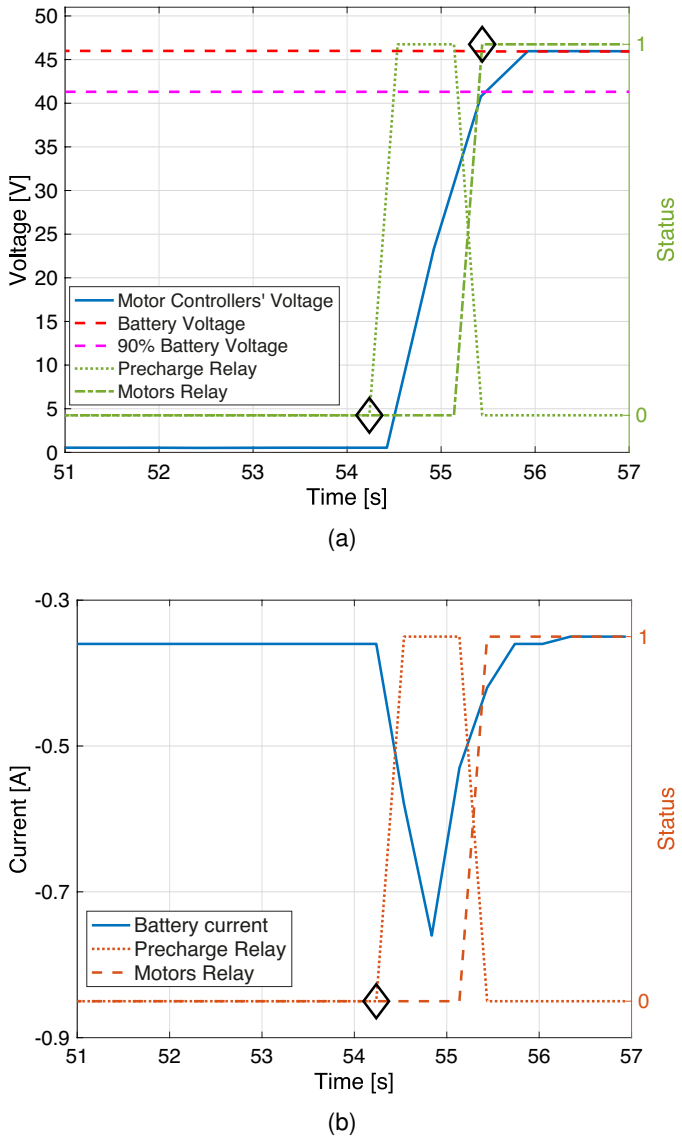


Figure 5.13: Load voltage (a) and battery current (b) during BMSSPV V2 pre-charge sequence.

5.2.5 Power consumption and sleep mode

Power consumption was by far the biggest problem that V1 faced and the one leading to the development of V2. This test serves the purpose of comparing the power consumption of both versions. Using the acquired data[1], the time each version can be connected to the battery before completely draining it can also be calculated.

Figure 5.14 shows the power consumption of V1. This version, as we have already seen, cannot have a sleeping mode due to the lack of inputs to sense when the boat is turned on, or the USB cable is connected. Thus V1 consumes, on average, $922.5\,\mathrm{mW}$ no matter if the battery is being used or not. The spikes seen in Figure 5.14 are the power consumed while the `BMS_routine` is running. In this test, the BMS's refresh rate was set to $1\,\mathrm{Hz}$ which translates to the five spikes seen.

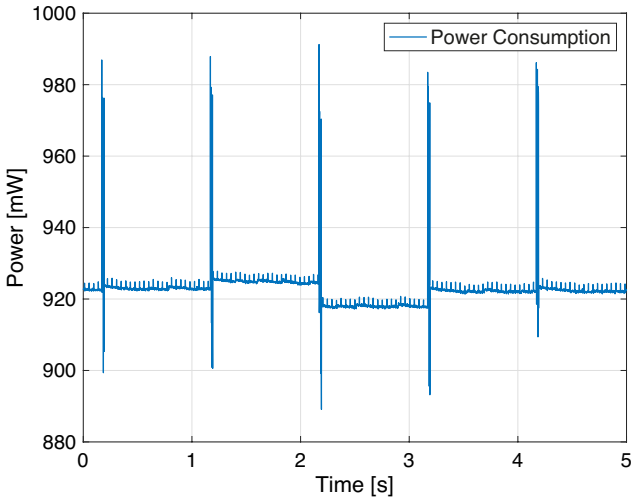


Figure 5.14: BMSSPV V1 power consumption.

Knowing the power being consumed and the battery's capacity its lifetime can be calculated with (4.12). Considering a battery capacity of $1500\,\mathrm{Wh}$, the battery lifetime with BMSSPV V1 is given by

$$\Delta T = \frac{1500\,\mathrm{Wh}}{922.5 \times 10^{-3}\,\mathrm{W} \times 24\,\mathrm{h}} = 67.75\,\mathrm{d}. \tag{5.1}$$

Both the power consumption and the battery lifetime presented here are very similar to the figures presented in Sections 4.2.8 and 4.3.1. The only difference is that these are experimental. In contrast, the ones mentioned previously were theoretical.

This test proves that V1 is not suitable for continuously monitoring a battery and thus proves that V2 was required. Figure 5.15 shows the results obtained for V2. This time we have two different graphics, Figure 5.15a shows the power consumption in normal mode, which is the same state where V1 was all the time, while Figure 5.15b shows the power consumption of V2 while sleeping (i.e. the battery is not being used).

We can see that while in normal mode, V2 consumes, on average, $241\,\mathrm{mW}$, which represents a $74\,\%$ decrease when compared with V1. Once again, the spikes shown are the power consumed while the

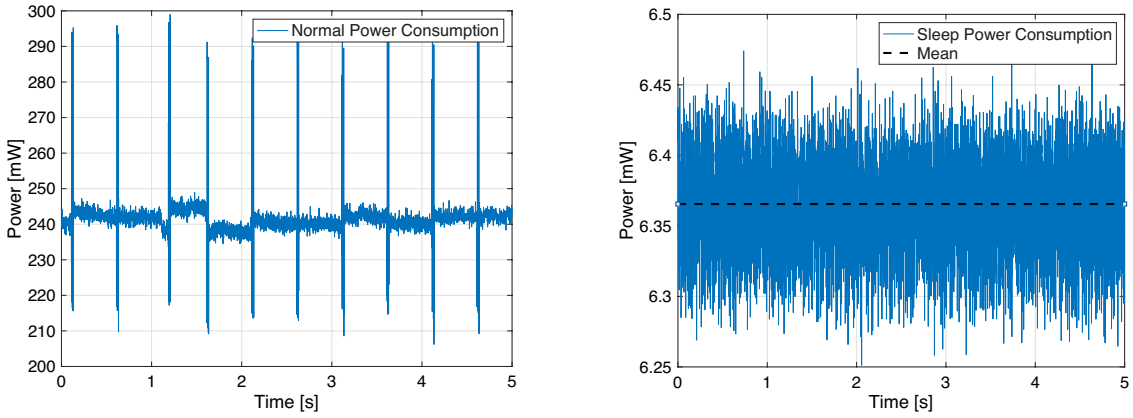
[1] The procedure used to acquire the data presented in this section is the same used in Section 4.4.9, Footnote 21.

`BMS_routine` is running, but this time the refresh rate was set to $0.5\,\mathrm{Hz}$.

By analyzing Figure 5.15b we can conclude that while sleeping V2 consumes, on average, $6.37\,\mathrm{mW}$ which represents a decrease of more than $99\,\%$ when compared to V1. This translates to a lifetime of

$$\Delta T = \frac{1500\,\mathrm{Wh}}{6.37 \times 10^{-3}\,\mathrm{W} \times 24\,\mathrm{h}} = 9811.6\,\mathrm{d} \approx 27\,\mathrm{yr}, \tag{5.2}$$

which is 114 times the lifetime of V1. This test proves that the requirement that the BMS must have a standby life of at least years is satisfied.



(a) Normal.

(b) Sleep.

Figure 5.15: BMSSPV V2 power consumption under the different modes.

The colossal power consumption decrease seen here is mainly due to the new DC-DC architecture and the MOSFET that turns off everything that is not being used while sleeping.

6

Conclusions

Conclusions

The objective of this thesis was to address the problem of battery management for the TSB project, and more broadly, the issues an electric vehicle that has multiple sources of power may face when using commercially available solutions. To solve this problem, we started by presenting the solution that was previously being used within the TSB project, followed by a description of the different BMS topologies available that could be used for this work. Commercially available alternatives were also analyzed, but they were ultimately discarded in favour of a custom made solution that would allow creating within the team the know-how needed to improve a critical boat system like the BMS in future project iterations.

The different battery designs that have already been used were then presented, followed by the definition of the requirements that the proposed BMS should comply with, as well as the architecture that the proposed system would follow.

Given that the proposed system is to be applied in a specific context and has to respect some regulations, the solution presented here consists of a centralized solution that implements all battery management in a single PCB. If, for example, the number of cells in series wasn't so constrained, the solution presented here would most probably have been different.

In Chapter 4 the implementation of the proposed system was presented, starting with the choice of both the BSM and the MCU. Then, three different versions were presented: V0 was developed to prove that the Teensy 3.2 could successfully communicate with LTC6811-2 and acquire the battery's data; Following the successful test with V0, V1 was designed already with the boat integration in mind, this version is being used in SR-02 since 2019; Lastly, with the lessons taken from V1, V2 was developed and later integrated into SR-03 in 2021.

Finally, Chapter 5 exposes the deployment of V2 as well all the tests performed to it in order to validate that it serves its purpose and respects all the predefined requirements.

Besides the laboratory tests carried out in this thesis, the implemented system has already been extensively tested on multiple occasions in different scenarios. The system has been actively used for more than six months in TSB's most recent solar-powered prototype SR-03, with which the team participated in two different international competitions, which took place in Monaco and Portugal. In Portugal, we conquered second place amongst six international competitor teams.

The system was also used for multiple consecutive hours during TSB's own event, Odisseia TSB, which took place in the Madeira Archipelago last September. During this event the boat sailed more than $40\,\mathrm{NM}$ for more than $10\,\mathrm{h}$ divided in three days. Both during the competitions and Odisseia TSB the BMS never had any problems and was always able to protect the battery regardless of the situation. These real-world tests prove that the developed system is reliable both in laboratory conditions and in the boat, where it faces a lot more stress due to vibrations, marine environment, ambient temperature

and so on.

Overall the implemented BMS can be considered a success. The objective defined by the team was to have a more straightforward BMS that could alone implement all the features needed to guarantee the battery's safety. This objective was successfully achieved. Moreover, the implemented BMS allows the team to have way more control over the way the battery behaves, which ultimately means an improvement in the overall boat's performance.

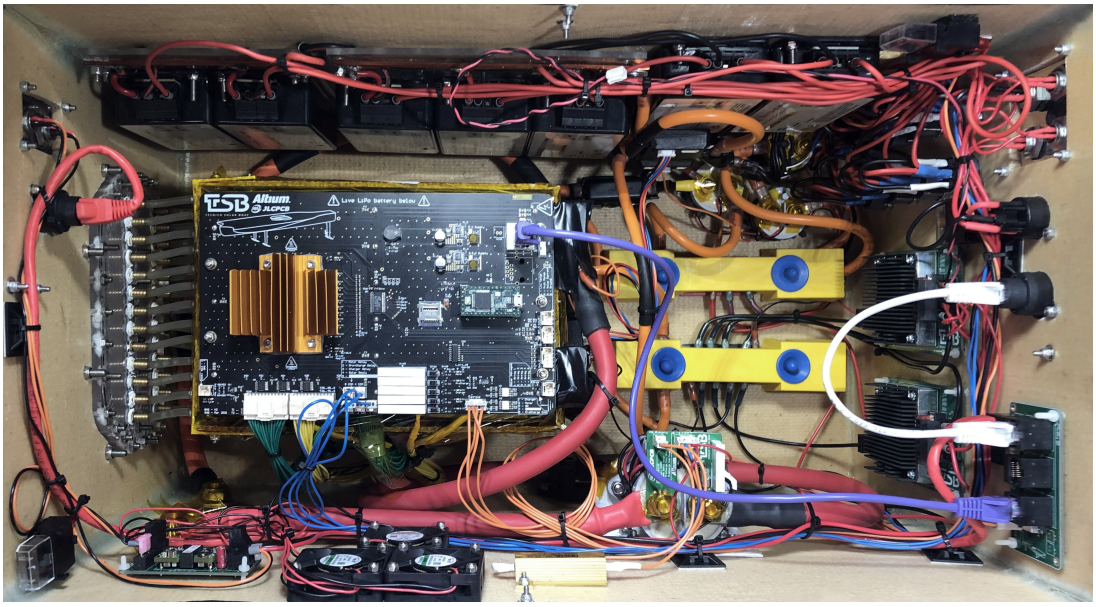


Figure 6.1: BMSSPV V2 integrated in SR-03's battery box.

6.1 Future work

In what concerns the BMS itself, the library used to communicate with the BSM can be updated given that Analog Devices released a new version where features like open wire check for multiple cells and other improvements were implemented. This updated library also fixes the errors that we had to fix in the beginning, as seen in Section 4.4.1.

Besides, digital signal processing should be implemented so that decisions are not taken based on a single measurement. Up to now, not using it has not brought any problems. Nonetheless, it should be added to further improve the developed system.

Regarding the BMS's power consumption, even though huge improvements were already achieved from V1 to V2, theoretically, there is still room for improvements. To achieve this, a good approach would be to add $0\,\Omega$ resistors in series with the power supply of each device so that the circuit can be interrupted and the power consumption measured. Such an approach would provide much insight into the overall power consumption footprint of the BMS.

Also regarding the power consumption, entering a low power mode in between measurements is something that can be thought of, but care should be taken to guarantee that the BMS can wake up with all the interrupts used.

In regards to the battery itself, some things should have been done differently. Changing a damaged

cell proved to be a challenging task, and reassembling the battery again was even more complex due to the design of the battery enclosure. Besides, the cooling of the bus bars is something that needs to be rethought. When drawing too high currents from the battery, they tend to heat quite a lot.

Capsizing is something that all boats are subject to, but that SR-03 is not ready for. Given that, and learning from previous experiences, another aspect that can be improved is the waterproofing of all systems. This can be challenging, especially for the battery, due to cooling constraints. However, it should be investigated to reduce the risk of a severe accident if water reaches the battery. Also related to this, a device that automatically opens all relays in case the boat reaches a roll above a specific limit would also improve the boat's safety, given that it will automatically be turned off even before the boat capsizes.

Figure 6.2: SR-03 sailing in Madeira during Odisseia TSB, September 2021.

Figure 6.3: SR-03 alongside São Miguel 01, the hydrogen powered boat, in Monaco, July 2021.

Bibliography

[1] E. Gelber. (2018) Solar is here to stay, but where are the boats? [Online]. Available: https://www.nationalgeographic.com/travel/features/solar-boats-green-energy-sustainability/ (Accessed on 17/10/2021).

[2] C. McGrady. (2016) Brushless vs Brushed Motor. [Online]. Available: https://www.arrow.com/en/research-and-events/articles/which-dc-motor-is-best-for-your-application (Accessed on 17/10/2021).

[3] Solar Sport One. Technical Regulations 2021. Rule 7.16. [Online]. Available: https://web.ist.utl.pt/ist182045/Thesis/Rules.pdf (Accessed on 17/10/2021).

[4] D. Andrea, *Battery Management Systems for Large Lithium Ion Battery Packs*, 1st ed. Artech House, 2010, Section 2.2.

[5] D. Andrea, *Battery Management Systems for Large Lithium Ion Battery Packs*, 1st ed. Artech House, 2010, Section 2.3.

[6] Energus. Energus Tiny BMS. [Online]. Available: https://www.energusps.com/shop/product/tiny-bms-s516-150a-750a-36?category=4 (Accessed on 17/10/2021).

[7] LION Smart. Li-BMS V4 Modular Battery-Management-System. [Online]. Available: https://web.ist.utl.pt/ist182045/Thesis/Li-BMS.pdf (Accessed on 17/10/2021).

[8] LION Smart. LIGHT Battery Concept. [Online]. Available: https://web.ist.utl.pt/ist182045/Thesis/Light_Battery.pdf (Accessed on 17/10/2021).

[9] LION E-Mobility AG. Linear Technology and LION Smart Demonstrate First Wireless Battery Management System in BMW i3 at CES Show 2017 in Las Vegas. [Online]. Available: https://lionemobility.de/en/linear-technology-and-lion-smart-demonstrate-first-wireless-battery-management-system-in-bmw-i3-at-ces-show-2017-in-las-vegas/ (Accessed on 17/10/2021).

[10] REC. REC BMSMaster 9M. [Online]. Available: https://web.ist.utl.pt/ist182045/Thesis/REC_BMS_MS.pdf (Accessed on 17/10/2021).

[11] D. Andrea, *Battery Management Systems for Large Lithium Ion Battery Packs*, 1st ed. Artech House, 2010, Section 2.3.4.

[12] Elithion. Elithion Lithiumate HD BMS. [Online]. Available: https://www.elithion.com/lithiumate.php (Accessed on 17/10/2021).

[13] Solar Sport One. Technical Regulations 2021. Rule 7.9.4. [Online]. Available: https://web.ist.utl.pt/ist182045/Thesis/Rules.pdf (Accessed on 17/10/2021).

[14] Solar Sport One. Technical Regulations 2021. Rule 7.9. [Online]. Available: https://web.ist.utl.pt/ist182045/Thesis/Rules.pdf (Accessed on 17/10/2021).

[15] Solar Sport One. Technical Regulations 2021. Rule 7.6. [Online]. Available: https://web.ist.utl.pt/ist182045/Thesis/Rules.pdf (Accessed on 17/10/2021).

[16] CAN Aviation Alliance. ARINC 825. [Online]. Available: https://www.arinc-825.com/the-arinc825-standard/ (Accessed on 17/10/2021).

[17] D. Andrea, *Battery Management Systems for Large Lithium Ion Battery Packs*, 1st ed. Artech House, 2010, Section 5.1.1.

[18] D. Andrea. (2018) Comparison of BMS ICs for Large Li-ion Battery Packs. [Online]. Available: http://liionbms.com/html/BMS_IC_table.html (Accessed on 17/10/2021).

[19] B. Santos, "Battery Management System applied to Projecto FST's EV prototype," Master's thesis, Instituto Superior Técnico, 2014.

[20] Analog Devices. LTC6811's Datasheet. [Online]. Available: https://web.ist.utl.pt/ist182045/Thesis/LTC6811.pdf (Accessed on 17/10/2021).

[21] A. Samiappan and J. Kathuria. (2017) Upgrading 8- and 16-bit MCU designs: 32-bit MCU architectures. [Online]. Available: https://www.embedded.com/upgrading-8-and-16-bit-mcu-designs-32-bit-mcu-architectures/ (Accessed on 17/10/2021).

[22] FST Lisboa. FST Lisboa Cell Stack Configuration. [Online]. Available: https://web.ist.utl.pt/ist182045/Thesis/FST-Cell-Configuration.pdf (Accessed on 17/10/2021).

[23] R. Rossetti. Buck vs. Boost Converters: How Do They Compare? [Online]. Available: https://www.maximintegrated.com/en/design/blog/buck-vs-boost-converters-how-do-they-compare.html (Accessed on 17/10/2021).

[24] B. Munari and A. Schneer. (2020) How to design a precharge circuit for hybrid and electric vehicle applications. [Online]. Available: https://web.ist.utl.pt/ist182045/Thesis/Precharge_Circuit.pdf (Accessed on 17/10/2021).

[25] Traco Power. TDR 3-4811WISM's Datasheet. [Online]. Available: https://web.ist.utl.pt/ist182045/Thesis/TDR3WISM.pdf (Accessed on 17/10/2021).

[26] Analog Devices. LTC3990's Datasheet. [Online]. Available: https://web.ist.utl.pt/ist182045/Thesis/LT3990.pdf (Accessed on 17/10/2021).

[27] C. Duffy. Teensy Snooze Library. [Online]. Available: https://github.com/duff2013/Snooze/tree/master (Accessed on 17/10/2021).

[28] Arm Limited. Mbed OS Overview. [Online]. Available: https://os.mbed.com/mbed-os/ (Accessed on 17/10/2021).

[29] Freescale Semiconductor, Inc. K20 Sub-Family Reference Manual. Chapter 7. [Online]. Available: https://web.ist.utl.pt/ist182045/Thesis/T3_2_MCU.pdf (Accessed on 17/10/2021).

A BMSSPV CAN and Serial messages

Table A.1: status message, CAN ID: $0x601$, Serial ID: $0x01$, LEN = 6.

Offset	Type	Name	Scale	Description
0	uint8_t	Status	-	Status of the battery
1	uint16_t	LTC Status	-	Status of the BSM
3	uint8_t	Cells Balancing0	-	Which cells are balancing [1:8]
4	uint8_t	Cells Balancing1	-	Which cells are balancing [9:12]
5	uint8_t	Warnings	-	Warnings

Table A.2 describes the what each bit of the Status variable means. 1 means `TRUE` 0 means `FALSE`.

Table A.2: Status bits description.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Pre charge Relay	Charge Relay	Solar Relay	Motor Relay	Balancing	Idle	Discharging	Charging

Table A.4 and Table A.3 describes the what each bit of the LTC Status variable means. 1 means `TRUE` 0 means `FALSE`.

Table A.3: LTC Status bits [0:7] description.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Open Wire Err.	STAT Meas. Test	AUX Meas. Test	ADC Overlap Test	MUX Test	STAT Memory Err.	AUX Mem. Err.	CELL Mem. Err.

Table A.4: LTC Status bits [8:15] description.

Bit 15:12	Bit 11	Bit 10	Bit 9	Bit 8
Cell that is open	-	-	PEC Error	Thermal shutdown

Table A.5 describes the what each bit of the Warnings variable means. 1 means `TRUE` 0 means `FALSE`.

Table A.5: Warnings bits description.

Bit 3	Bit 2	Bit 1	Bit 0
Under-voltage	Over-voltage	Over-temperature	Over-current

Table A.6: SOC_Voltage_Currents message, CAN ID: $0x611$, Serial ID: $0x11$, LEN = 8.

Offset	Type	Name	Scale	Description
0	float16	State of Charge	1e2	Battery State of Charge
2	float16	Voltage	5e2	Battery pack voltage [V]
4	float16	Current	1e2	Battery pack current [A]
6	float16	Solar Current	1e2	Solar current [A]

Table A.7: cellvoltages0 message, CAN ID: $0x621$, Serial ID: $0x21$, LEN = 8.

Offset	Type	Name	Scale	Description
0	float16	voltage1	1e4	Cell 1 voltage [V]
2	float16	voltage2	1e4	Cell 2 voltage [V]
4	float16	voltage3	1e4	Cell 3 voltage [V]
6	float16	voltage4	1e4	Cell 4 voltage [V]

Table A.8: cellvoltages1 message, CAN ID: $0x631$, Serial ID: $0x31$, LEN = 8.

Offset	Type	Name	Scale	Description
0	float16	voltage5	1e4	Cell 5 voltage [V]
2	float16	voltage6	1e4	Cell 6 voltage [V]
4	float16	voltage7	1e4	Cell 7 voltage [V]
6	float16	voltage8	1e4	Cell 8 voltage [V]

Table A.9: cellvoltages2 message, CAN ID: $0x641$, Serial ID: $0x41$, LEN = 8.

Offset	Type	Name	Scale	Description
0	float16	voltage9	1e4	Cell 9 voltage [V]
2	float16	voltage10	1e4	Cell 10 voltage [V]
4	float16	voltage11	1e4	Cell 11 voltage [V]
6	float16	voltage12	1e4	Cell 12 voltage [V]

Table A.10: SolarPanelVoltages0 message, CAN ID: $0x651$, Serial ID: $0x51$, LEN = 8.

Offset	Type	Name	Scale	Description
0	float16	SP1	1e3	Solar Array 1 voltage [V]
2	float16	SP2	1e3	Solar Array 2 voltage [V]
4	float16	SP3	1e3	Solar Array 3 voltage [V]
6	float16	SP4	1e3	Solar Array 4 voltage [V]

Table A.11: SolarPanelVoltages1 message, CAN ID: $0x661$, Serial ID: $0x61$, LEN = 2.

Offset	Type	Name	Scale	Description
0	float16	SP5	1e3	Solar Array 5 voltage [V]

Table A.12: temperatures0 message, CAN ID: $0x671$, Serial ID: $0x71$, LEN = 8.

Offset	Type	Name	Scale	Description
0	uint8_t	temperature1	-	Bus bar 1 temperature [°C]
1	uint8_t	temperature2	-	Bus bar 2 temperature [°C]
2	uint8_t	temperature3	-	Bus bar 3 temperature [°C]
3	uint8_t	temperature4	-	Bus bar 4 temperature [°C]
4	uint8_t	temperature5	-	Bus bar 5 temperature [°C]
5	uint8_t	temperature6	-	Bus bar 6 temperature [°C]
6	uint8_t	temperature7	-	Bus bar 7 temperature [°C]
7	uint8_t	temperature8	-	Bus bar 8 temperature [°C]

Table A.13: temperatures1 message, CAN ID: $0x681$, Serial ID: $0x81$, LEN = 8.

Offset	Type	Name	Scale	Description
0	uint8_t	temperature9	-	Bus bar 9 temperature [°C]
1	uint8_t	temperature10	-	Bus bar 10 temperature [°C]
2	uint8_t	temperature11	-	Bus bar 11 temperature [°C]
3	uint8_t	temperature12	-	Bus bar 12 temperature [°C]
4	uint8_t	temperature13	-	Bus bar 13 temperature [°C]
5	uint8_t	temperature14	-	Cell 1 temperature [°C]
6	uint8_t	temperature15	-	Cell 2 temperature [°C]
7	uint8_t	temperature16	-	Cell 3 temperature [°C]

Table A.14: temperatures2 message, CAN ID: $0x691$, Serial ID: $0x91$, LEN = 8.

Offset	Type	Name	Scale	Description
0	uint8_t	temperature17	-	Cell 4 temperature [°C]
1	uint8_t	temperature18	-	Cell 5 temperature [°C]
2	uint8_t	temperature19	-	Cell 6 temperature [°C]
3	uint8_t	temperature20	-	Cell 7 temperature [°C]
4	uint8_t	temperature21	-	Cell 8 temperature [°C]
5	uint8_t	temperature22	-	Cell 9 temperature [°C]
6	uint8_t	temperature23	-	Cell 10 temperature [°C]
7	uint8_t	temperature24	-	Cell 11 temperature [°C]

Table A.15: temperatures3 message, CAN ID: $0\text{x}6\text{A}1$, Serial ID: $0\text{x}\text{A}1$, LEN = 8.

Offset	Type	Name	Scale	Description
0	uint8_t	temperature25	-	Cell 12 temperature [$^\circ\text{C}$]
1	uint8_t	temperature26	-	Cell 13 temperature [$^\circ\text{C}$]
2	uint8_t	temperature27	-	Cell 14 temperature [$^\circ\text{C}$]
3	uint8_t	temperature28	-	Cell 15 temperature [$^\circ\text{C}$]
4	uint8_t	temperature29	-	Cell 16 temperature [$^\circ\text{C}$]
5	uint8_t	temperature30	-	Cell 17 temperature [$^\circ\text{C}$]
6	uint8_t	temperature31	-	Cell 18 temperature [$^\circ\text{C}$]
7	uint8_t	temperature32	-	Cell 19 temperature [$^\circ\text{C}$]

Table A.16: temperatures4 message, CAN ID: $0\text{x}6\text{B}1$, Serial ID: $0\text{x}\text{B}1$, LEN = 6.

Offset	Type	Name	Scale	Description
0	uint8_t	temperature33	-	Cell 20 temperature [$^\circ\text{C}$]
1	uint8_t	temperature34	-	Cell 21 temperature [$^\circ\text{C}$]
2	uint8_t	temperature35	-	Cell 22 temperature [$^\circ\text{C}$]
3	uint8_t	temperature36	-	Cell 23 temperature [$^\circ\text{C}$]
4	uint8_t	temperature37	-	Cell 24 temperature [$^\circ\text{C}$]
5	uint8_t	temperature38	-	Cell / bus bar max temperature [$^\circ\text{C}$]

Table A.17: temperatures5 message, CAN ID: $0\text{x}6\text{C}1$, Serial ID: $0\text{x}\text{C}1$, LEN = 4.

Offset	Type	Name	Scale	Description
0	uint8_t	temperature41	-	Ambient temperature [$^\circ\text{C}$]
1	uint8_t	temperature42	-	Heatsink 1 temperature [$^\circ\text{C}$]
2	uint8_t	temperature43	-	Heatsink 2 temperature [$^\circ\text{C}$]
3	uint8_t	temperature44	-	BSM temperature [$^\circ\text{C}$]

Serial messages for user-configurable parameters:

Table A.18: Parameters0, Serial ID: $0\text{x}0\text{C}$, LEN = 8.

Offset	Type	Name	Scale	Description
0	uint16_t	OV_THRESH	-	Over-voltage [V] threshold [mV]
2	uint16_t	UV_THRESH	-	Under-voltage [V] threshold [mV]
4	uint16_t	BAL_THRESH	-	Balancing threshold[1] [mV]
6	uint16_t	INBAL_THRESH	-	Unbalancing threshold[2] [mV]

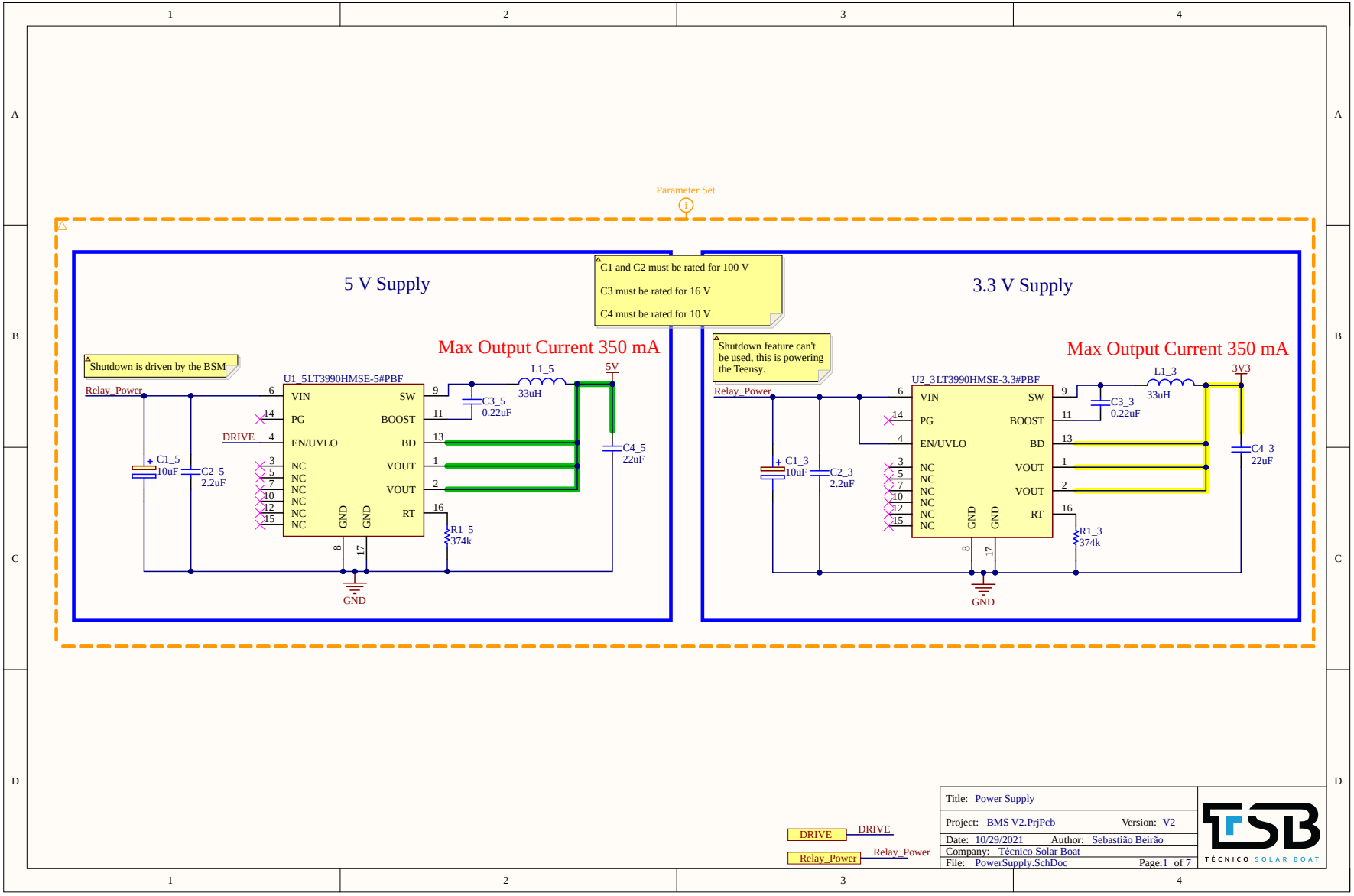
[1] Voltage bellow which balancing is not allowed.
[2] Maximum allowed voltage difference between the cell aggregates voltages.

Table A.19: Parameters1, Serial ID: $0\text{x}0\text{D}$, LEN = 8.

Offset	Type	Name	Scale	Description
0	uint16_t	DOC_THRESH	1e1	Discharge over-current threshold [A]
2	uint16_t	COC_THRESH	1e1	Charge over-current threshold [A]
4	uint8_t	TEMP_THRESH	-	Cells over-temperature threshold [$^{\circ}\text{C}$]
5	uint8_t	BAL_ALLOW	-	Balancing allowed 0 or 1
6	uint16_t	FREQ	-	BMS routine refresh rate [ms]

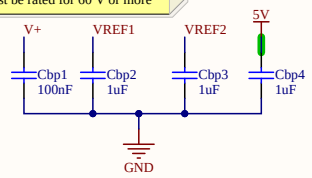
B

BMSSPV V2 Schematic


Parameter Set
5 V Supply
C1 and C2 must be rated for 100 V
C3 must be rated for 16 V
C4 must be rated for 10 V
3.3 V Supply
Max Output Current 350 mA
Shutdown is driven by the BSM
Shutdown feature can't be used, this is powering the Teensy.
Relay_Power
DRIVE
U1_5LT3990HMSE-5#PBF
U2_3LT3990HMSE-3.3#PBF
VIN
SW
PG
BOOST
EN/UVLO
BD
NC
VOUT
GND
RT
L1_5
33uH
5V
C3_5
0.22uF
C4_5
22uF
C1_5
10uF
C2_5
2.2uF
R1_5
374k
L1_3
3V3
C3_3
C4_3
C1_3
C2_3
R1_3
Title: Power Supply
Project: BMS V2.PrjPcb
Version: V2
Date: 10/29/2021
Author: Sebastião Beirão
Company: Técnico Solar Boat
File: PowerSupply.SchDoc
Page:1 of 7
TÉCNICO SOLAR BOAT

Bypass Capacitors

Cbp1 must be rated for 60 V or more

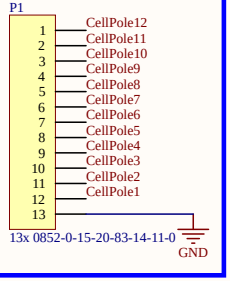


Cell Connections

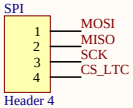
Pogo Pins that can be used:

0852-0-15-20-83-14-11-0

0852-1-15-20-83-14-11-0



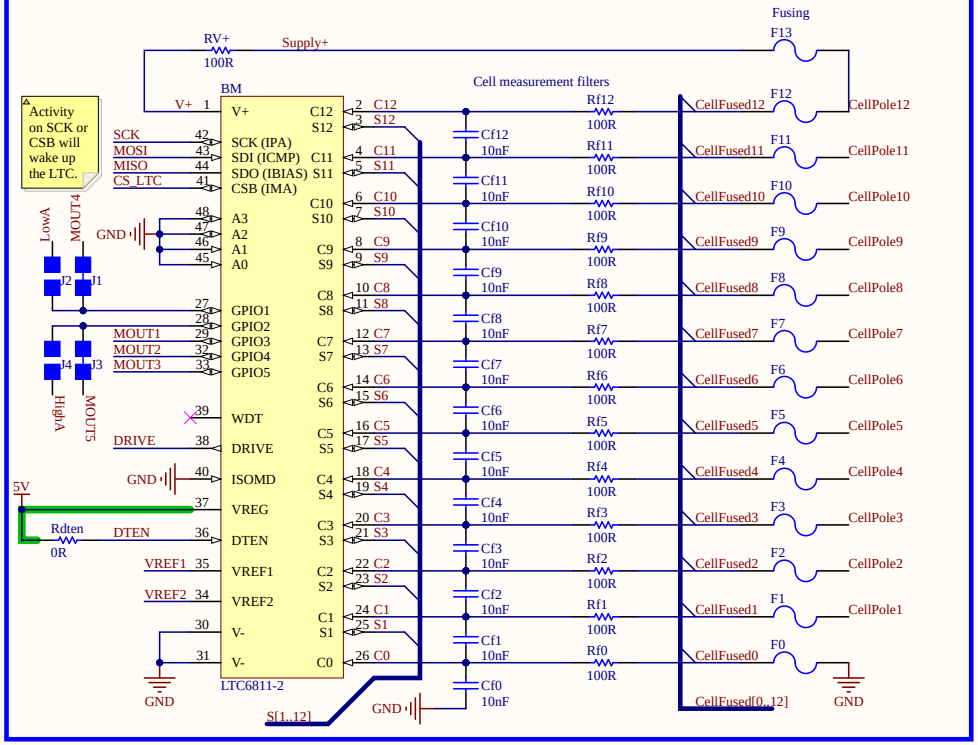
SPI Debug



Connect DRIVE pin to the 5 V regulator Enable Pin to shut it down while sleeping.

- DTEN is connected to VREG through a 0 R resistor. If the discahrge timer is used this resistor should be soldered, otherwise leave it unsoldered.
- ISOMD and A0-A3 connected to Ground for 4-Wire SPI mode.
- If Analog current sensor is used break J1 and J3 and short J2 and J4

LTC6811-2



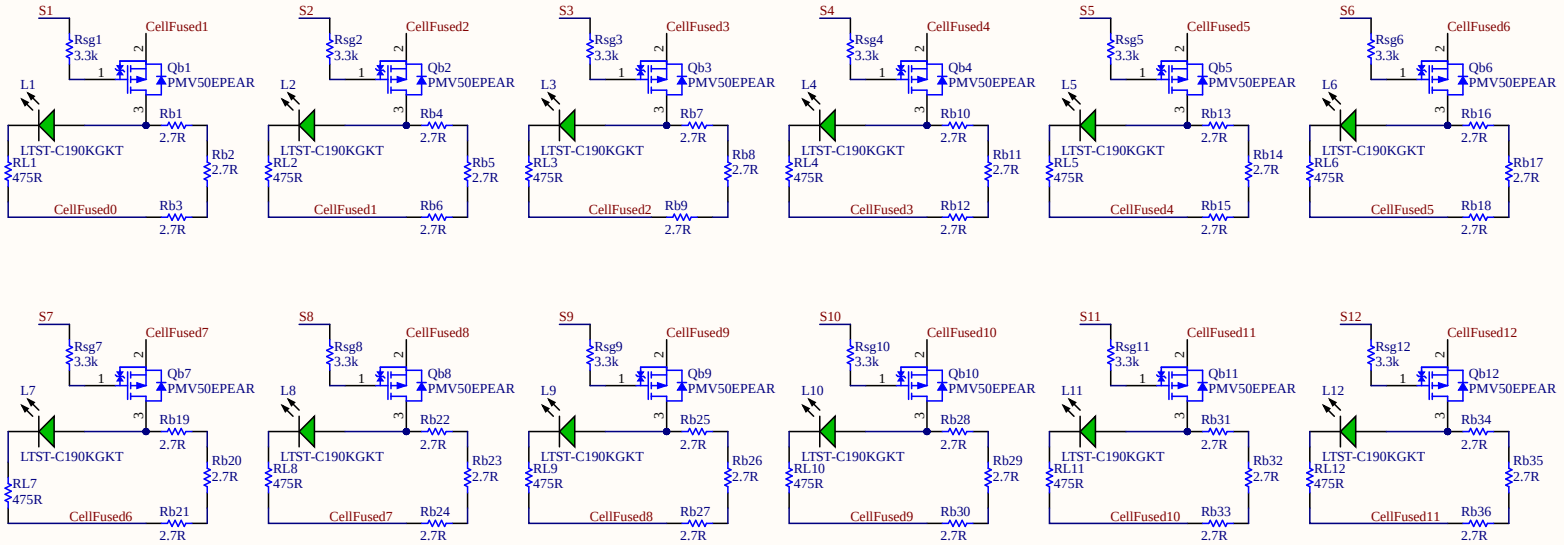
Title:	Battery Monitor
Project:	BMS V2.PrjPcb — Version: V2
Date:	10/29/2021 — Author: Sebastião Beirão
Company:	Técnico Solar Boat
File:	Battery Monitor.SchDoc — Page:2 of 7

TÉCNICO SOLAR BOAT

Balancing Circuit and heatshink

Balancing Current is 0,52 A @ 4.2 V

H
1/4 Brick Heat Sink
1/4 brick heat sink

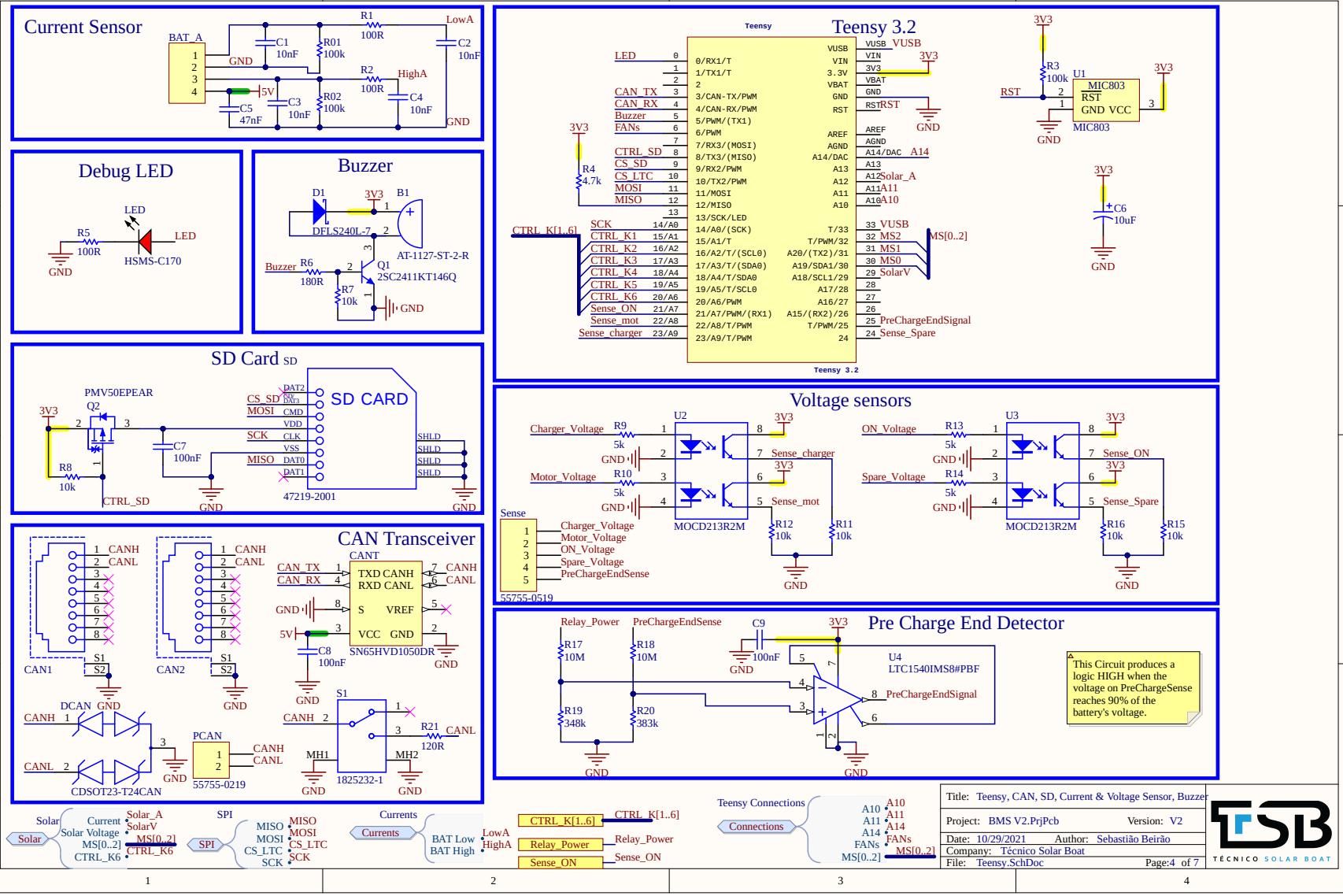


CellsFused[0..12] → CellFused[0..12]

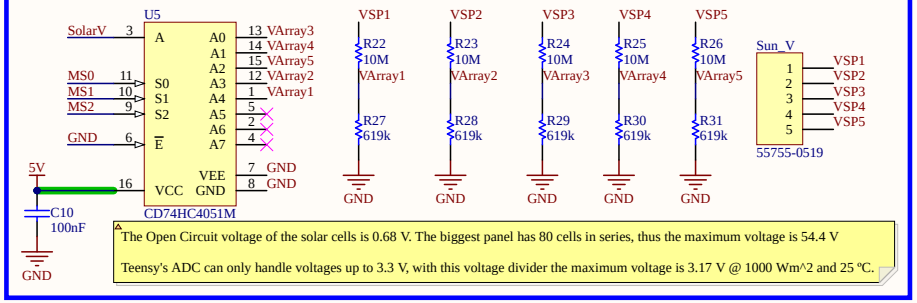
S[1..12] → S[1..12]

Title:	Balancing
Project:	BMS V2.PrjPcb — Version: V2
Date:	10/29/2021 — Author: Sebastião Beirão
Company:	Técnico Solar Boat
File:	Balancing.SchDoc — Page:3 of 7

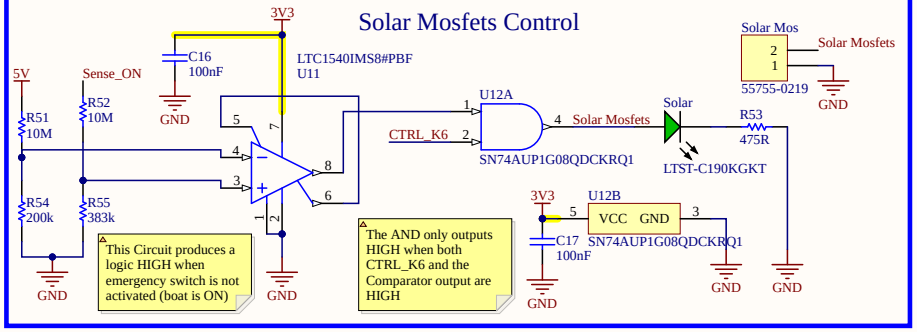
TÉCNICO SOLAR BOAT


Current Sensor
BAT_A
GND
C1 10nF
R01 100k
R1 100R
LowA
C2 10nF
5V
C3 10nF
R02 100k
R2 100R
HighA
C4 10nF
C5 47nF
GND
Debug LED
LED
R5 100R
HSMS-C170
GND
Buzzer
D1
DFLS240L-7
3V3
B1
AT-1127-ST-2-R
Buzzer
R6 180R
Q1 2SC2411KT146Q
R7 10k
GND
SD Card SD
PMV50EPEAR
Q2
3V3
R8 10k
CTRL_SD
C7 100nF
CS_SD
MOSI
SCK
MISO
SD CARD
47219-2001
SHLD
CAN Transceiver
CANH
CANL
CAN1
CAN2
S1
S2
DCAN
PCAN
CDSOT23-T24CAN
55755-0219
CANT
CAN_TX
CAN_RX
TXD CANH
RXD CANL
S
VREF
VCC GND
SN65HVD1050DR
C8 100nF
5V
MH1
MH2
1825232-1
R21 120R
Teensy 3.2
Teensy
LED
CAN_TX
CAN_RX
Buzzer
FANs
CTRL_SD
CS_SD
CS_LTC
MOSI
MISO
SCK
CTRL_K1
CTRL_K2
CTRL_K3
CTRL_K4
CTRL_K5
CTRL_K6
Sense_ON
Sense_mot
Sense_charger
0/RX1/T
1/TX1/T
3/CAN-TX/PWM
4/CAN-RX/PWM
5/PWM/(TX1)
6/PWM
7/RX3/(MOSI)
8/TX3/(MISO)
9/RX2/PWM
10/TX2/PWM
11/MOSI
12/MISO
13/SCK/LED
14/A0/(SCK)
15/A1/T
16/A2/T/(SCL0)
17/A3/T/(SDA0)
18/A4/T/SDA0
19/A5/T/SCL0
20/A6/PWM
21/A7/PWM/(RX1)
22/A8/T/PWM
23/A9/T/PWM
VUSB
VIN
3.3V
VBAT
GND
RST
AREF
AGND
A14/DAC
A13
A12
A11
A10
T/33
T/PWM/32
A20/(TX2)/31
A19/SDA1/30
A18/SCL1/29
A17/28
A16/27
A15/(RX2)/26
T/PWM/25
24
Solar_A
MS1
MS0
SolarV
PreChargeEndSignal
Sense_Spare
Teensy 3.2
R4 4.7k
MS[0..2]
R3 100k
U1
MIC803
C6 10uF
Voltage sensors
Charger_Voltage
Motor_Voltage
ON_Voltage
Spare_Voltage
R9 5k
R10 5k
R13 5k
R14 5k
U2
U3
MOCD213R2M
Sense_charger
Sense_mot
Sense_ON
Sense_Spare
R11 10k
R12 10k
R15 10k
R16 10k
Sense
55755-0519
Pre Charge End Detector
Relay_Power
PreChargeEndSense
R17 10M
R18 10M
R19 348k
R20 383k
C9 100nF
U4
LTC1540IMS8#PBF
This Circuit produces a logic HIGH when the voltage on PreChargeSense reaches 90% of the battery's voltage.
Solar
Current
Solar Voltage
MS[0..2]
CTRL_K6
Solar_A
SolarV
SPI
MISO
MOSI
CS_LTC
SCK
Currents
BAT Low
BAT High
LowA
HighA
CTRL_K[1..6]
Relay_Power
Sense_ON
Teensy Connections
Connections
A10
A11
A14
FANs
MS[0..2]
Title: Teensy, CAN, SD, Current & Voltage Sensor, Buzzer
Project: BMS V2.PrjPcb
Version: V2
Date: 10/29/2021
Author: Sebastião Beirão
Company: Técnico Solar Boat
File: Teensy.SchDoc
Page:4 of 7
TÉCNICO SOLAR BOAT

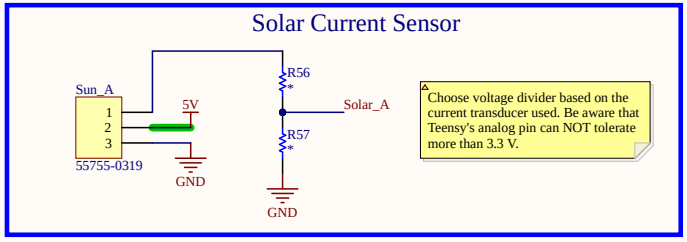
Solar Panels Voltage

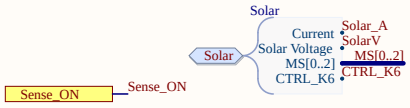


Solar Mosfets Control



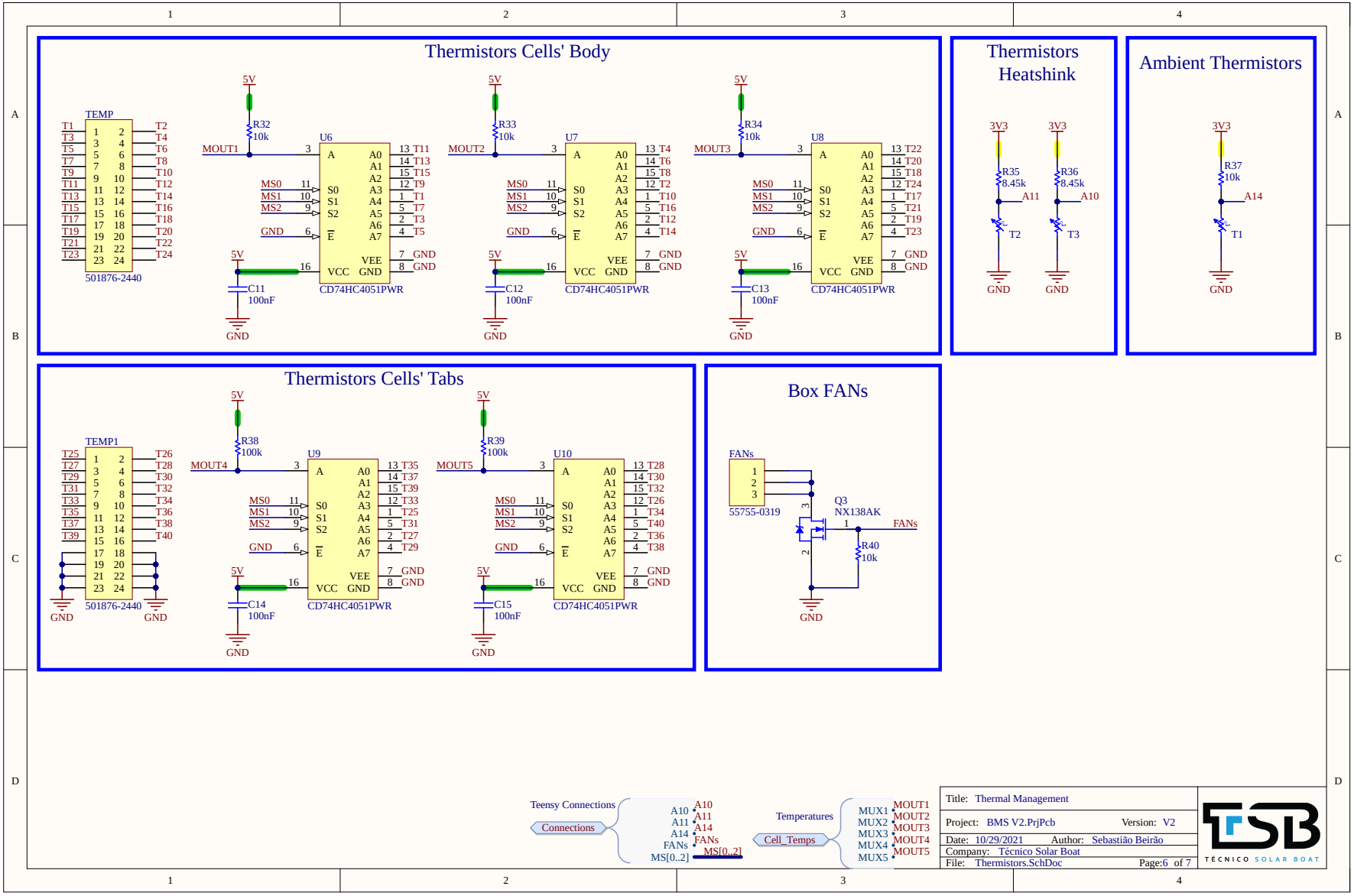
Solar Current Sensor

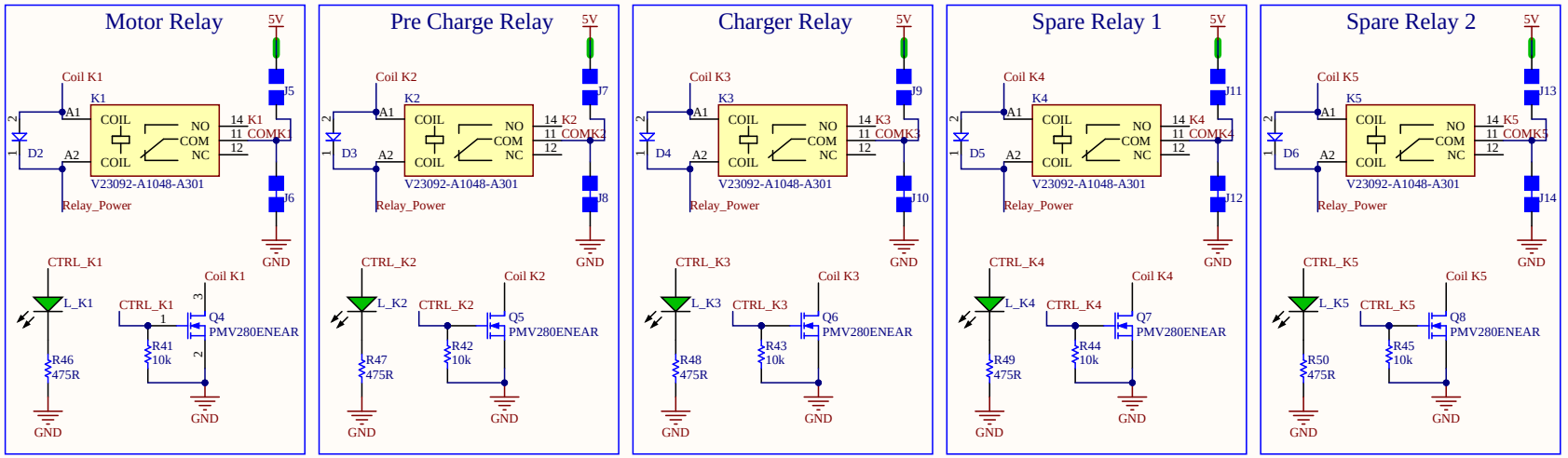


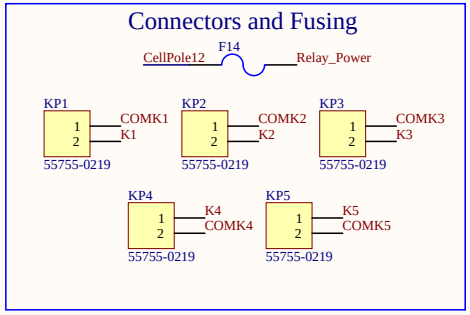


Title:	Solar Voltage, Current and Cut-off
Project:	BMS V2.PrjPcb — Version: V2
Date:	10/29/2021 — Author: Sebastião Beirão
Company:	Técnico Solar Boat
File:	Solar.SchDoc — Page:5 of 7

TÉCNICO SOLAR BOAT


Thermistors Cells' Body
TEMP
501876-2440
R32
10k
R33
10k
R34
10k
U6
U7
U8
MOUT1
MOUT2
MOUT3
MS0
MS1
MS2
GND
5V
C11
100nF
C12
100nF
C13
100nF
CD74HC4051PWR
Thermistors Heatshink
3V3
R35
8.45k
R36
8.45k
A11
A10
T2
T3
Ambient Thermistors
R37
10k
A14
T1
Thermistors Cells' Tabs
TEMP1
R38
100k
R39
100k
U9
U10
MOUT4
MOUT5
C14
100nF
C15
100nF
Box FANs
FANs
55755-0319
Q3
NX138AK
R40
10k
Teensy Connections
Connections
Temperatures
Cell_Temps
MS[0..2]
MUX1
MUX2
MUX3
MUX4
MUX5
Title: Thermal Management
Project: BMS V2.PrjPcb
Version: V2
Date: 10/29/2021
Author: Sebastião Beirão
Company: Técnico Solar Boat
File: Thermistors.SchDoc
Page:6 of 7
TÉCNICO SOLAR BOAT





Relay_Power — Relay_Power
CellPole12 — CellPole12
CTRL_K[1..6] — CTRL_K[1..6]

Title:	Relays and Solar Mosfets Control
Project:	BMS V2.PrjPcb — Version: V2
Date:	10/29/2021 — Author: Sebastião Beirão
Company:	Técnico Solar Boat
File:	Relays.SchDoc — Page:7 of 7

TÉCNICO SOLAR BOAT

C BMSSPV V2 PCB masks

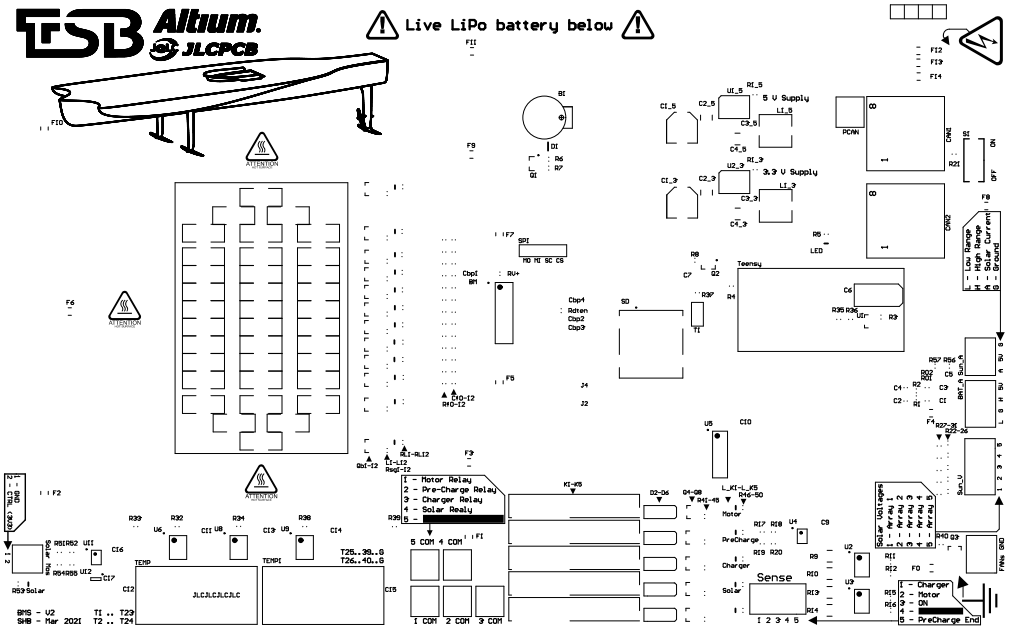


Figure C.1: BMSSPV V2 Top Overlay.

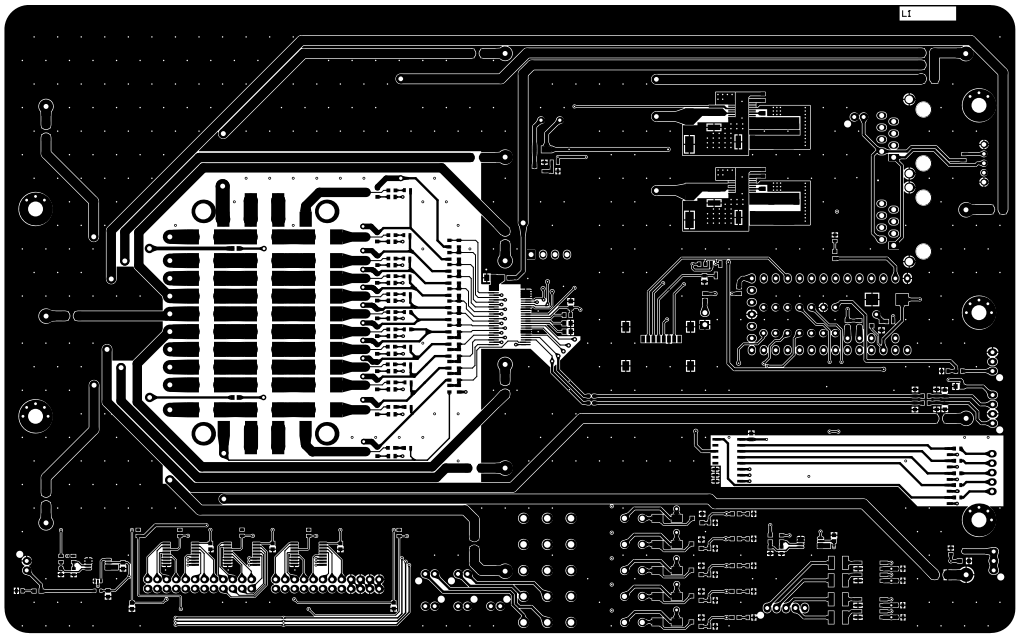


Figure C.2: BMSSPV V2 Top Layer.

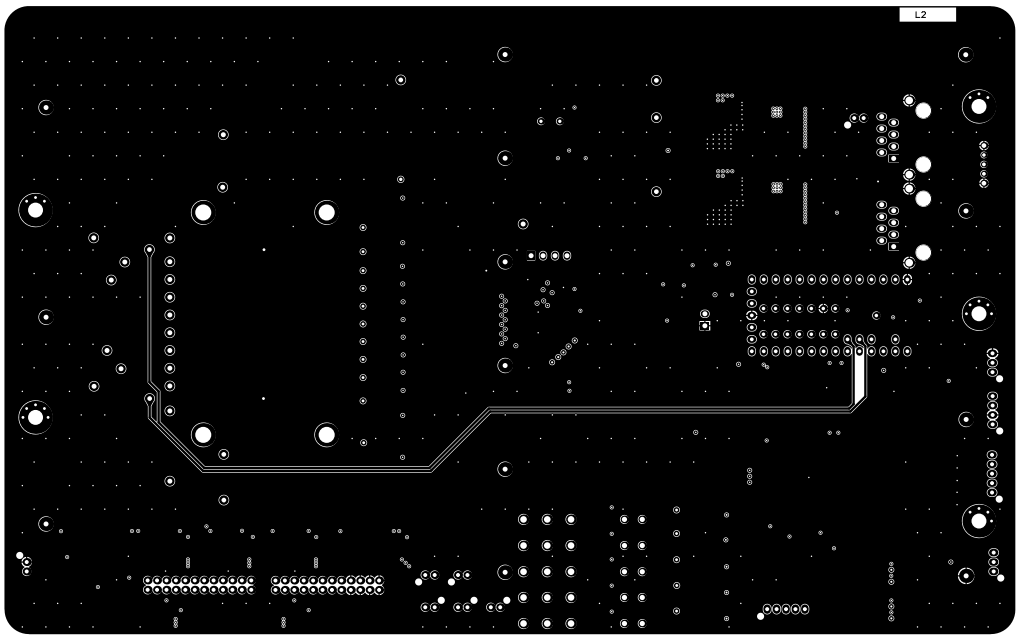


Figure C.3: BMSSPV V2 Mid 1 Layer.

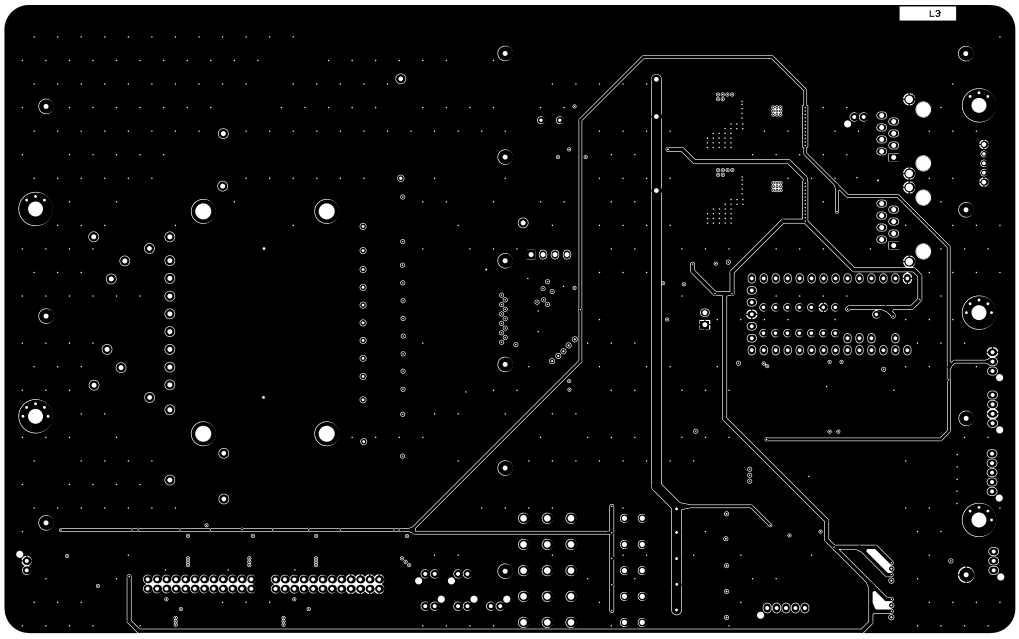


Figure C.4: BMSSPV V2 Mid 2 Layer.

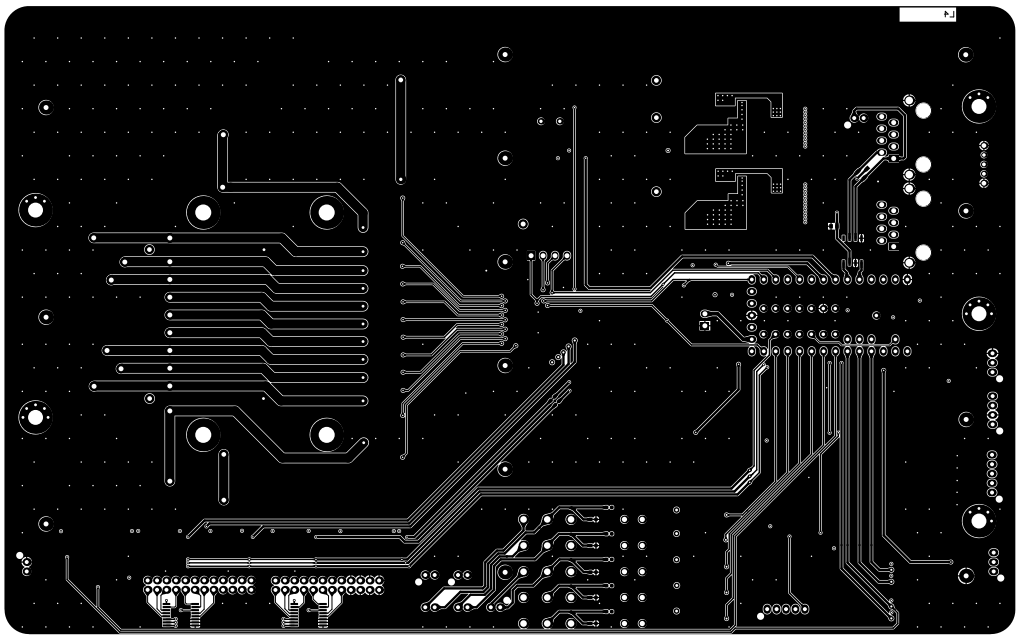


Figure C.5: BMSSPV V2 Bottom Layer.

D

Cell's Datasheet

1. 序言 PREFACE

此规格书适用于深圳市风云电池有限公司的锂聚合物可充电电池产品

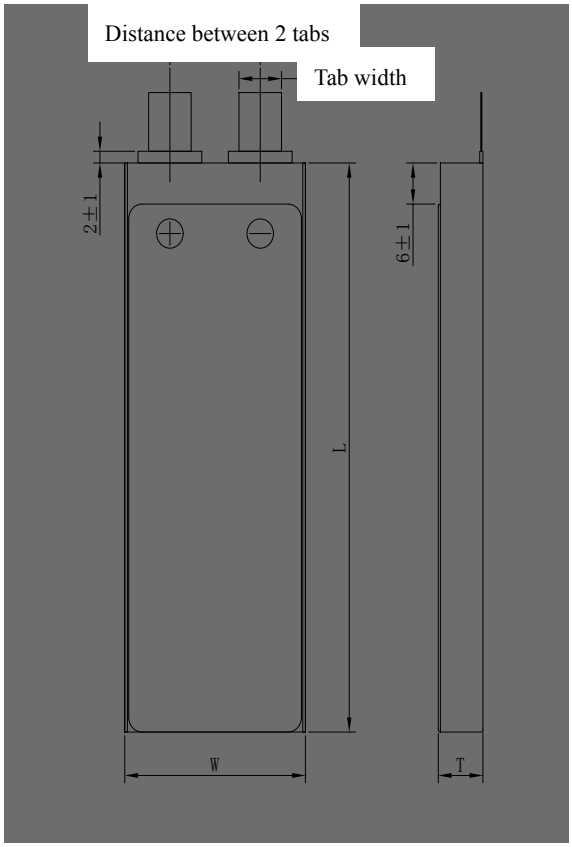
The specification is suitable for the performance of Lithium-Polymer (LIP) rechargeable battery produced by the SHENZHEN MELASTA BATTERY CO., LTD.

2. 型号 MODEL

SLPBA875175 16800mAh 15C 3.7V

3. 产品规格 SPECIFICATION

单颗电池规格 Specifications of single cell

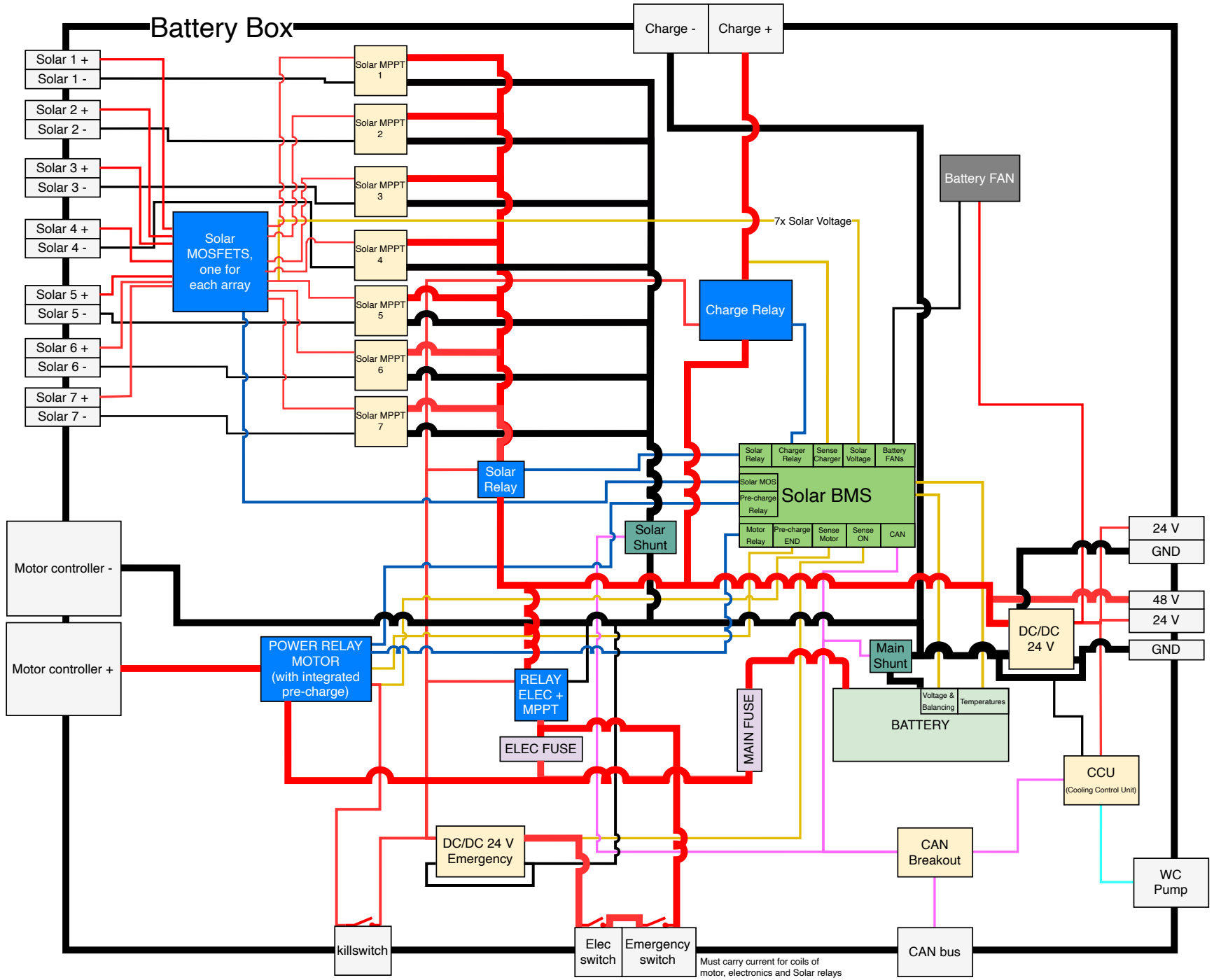


◆电芯正极材料 Cell Cathode Materials		$LiCoO_2$
◆标称容量 Typical Capacity①		16.8Ah
◆标称电压 Nominal Voltage		3.7V
◆ 充电条件 Charge Condition	最大可持续充电电流 Max. Continuous Charge Current	16.8A
	峰值充电电流 Peak Charge Current	33.6A (≤1sec)
	充电截止电压 Charging Cut-off Voltage	4.2V±0.03V
◆ 放电条件 Discharge Condition	Max Continuous Discharge Current	200A
	Peak Discharge Current	336A(≤0.1sec)
	Discharging Cut-off Voltage	3.0V
◆交流内阻 AC Impedance(mOHM)		0.6±0.2
◆循环寿命【充电:1.0C,放电:15C】 Cycle Life【CHA:1.0C,DCH:15C】		>100cycles
◆使用温度 Operating Temp.	充电 Charge	0℃~45℃
	放电 Discharge	-20℃~60℃
◆ 电芯尺寸 Cell Dimensions	厚度 Thickness(T)	10.5±0.3mm
	宽度 Width(W)	74.0±1mm
	长度 Length(L)	174±1mm
	极耳间距 Distance Between Tabs	40±1mm
◆ 极耳尺寸 Dimensions of Cell tabs	极耳宽度 Tab Width	25mm
	极耳厚度 Tab Thickness	0.2mm
	极耳长度 Tab Length	Max 30mm
◆重量 Weight(g)		304±5.0
①标称容量：0.5CmA,4.2V~3.0V@23℃±2℃ Typical Capacity:0.5CmA,4.2V~3.0V@23℃±2℃		

E

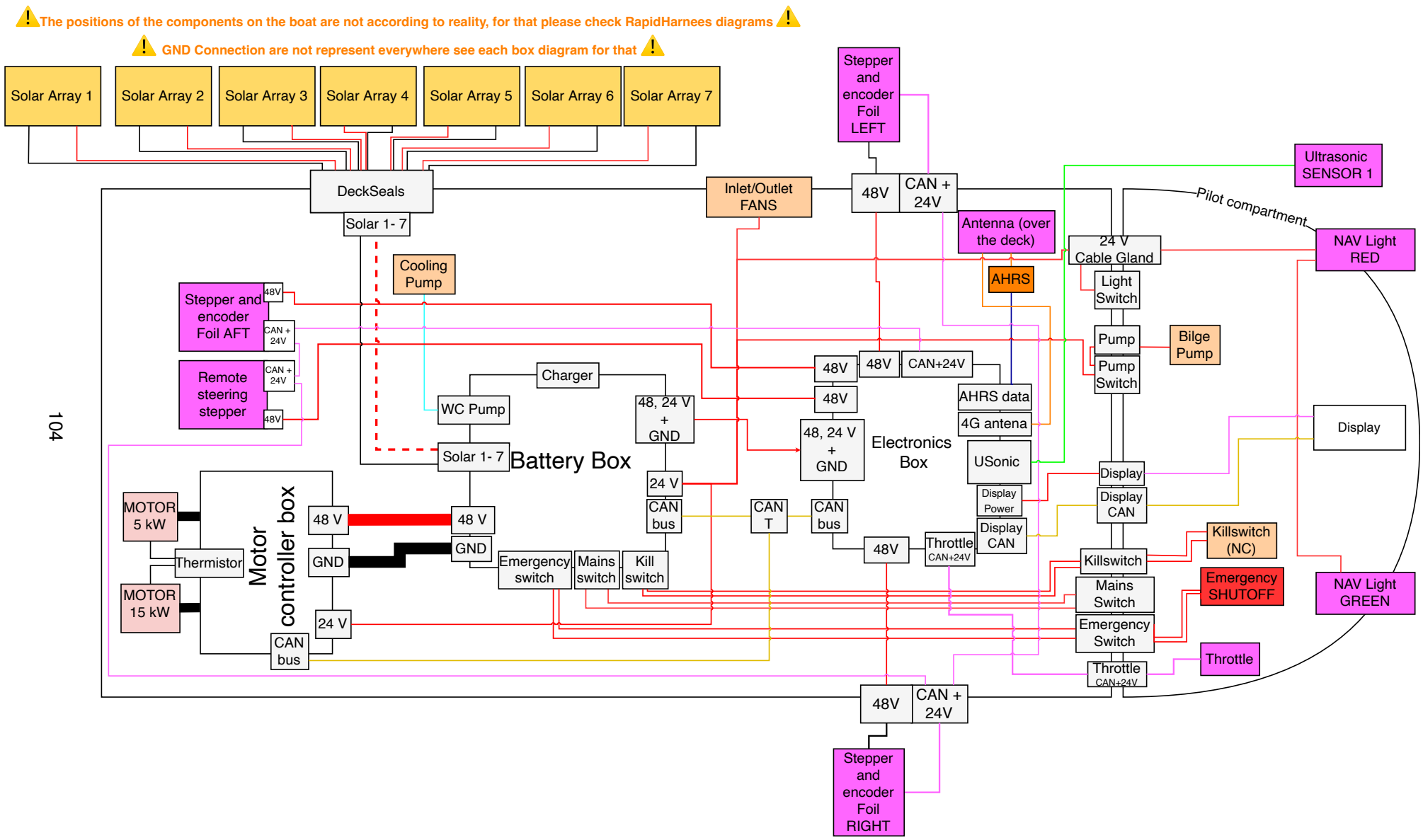
Battery box’s schematic

⚠ Shunts should be connected to 24V DC/DC ⚠
⚠ MPPTs should be connected to the CAN Breakout ⚠
⚠ BMS actuates the Relay on the low side ⚠



F

Boat Overview


⚠ The positions of the components on the boat are not according to reality, for that please check RapidHarnees diagrams ⚠
⚠ GND Connection are not represent everywhere see each box diagram for that ⚠
Solar Array 1
Solar Array 2
Solar Array 3
Solar Array 4
Solar Array 5
Solar Array 6
Solar Array 7
Stepper and encoder Foil LEFT
Ultrasonic SENSOR 1
DeckSeals
Solar 1- 7
Inlet/Outlet FANS
48V
CAN + 24V
Antenna (over the deck)
Pilot compartment
24 V Cable Gland
NAV Light RED
Cooling Pump
AHRS
Light Switch
Stepper and encoder Foil AFT
48V
CAN + 24V
Pump
Bilge Pump
Remote steering stepper
CAN + 24V
48V
Pump Switch
Charger
48V
48V
CAN+24V
AHRS data
WC Pump
48, 24 V + GND
48V
4G antena
Display
Solar 1- 7
Battery Box
48, 24 V + GND
Electronics Box
USonic
Display
24 V
Display Power
Display CAN
MOTOR 5 kW
Motor controller box
48 V
48 V
CAN bus
CAN T
CAN bus
Killswitch (NC)
Thermistor
GND
GND
48V
Throttle CAN+24V
Display CAN
Killswitch
Emergency switch
Mains switch
Kill switch
Mains Switch
Emergency SHUTOFF
NAV Light GREEN
MOTOR 15 kW
24 V
Emergency Switch
CAN bus
Throttle
Throttle CAN+24V
48V
CAN + 24V
Stepper and encoder Foil RIGHT