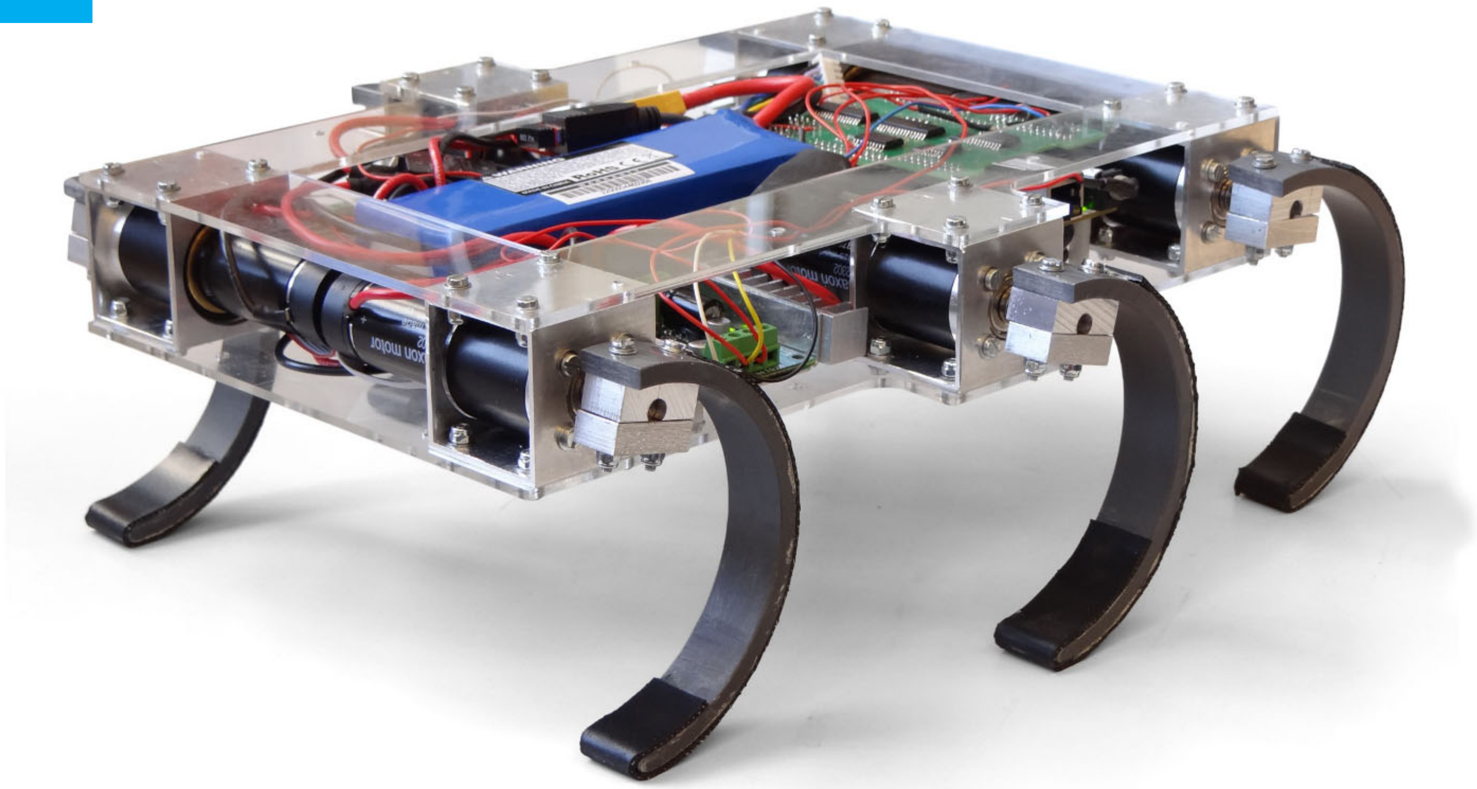
Battery Management System

Bachelor thesis

Pieter van der Kamp (4295889)
Bob Dullaart (4290631)



Abstract

In this thesis we will walk you through the design choices we made in designing a battery management system (BMS) for the Zebro robot. This specific BMS is designed as a subcomponent of an autonomous charging system. The main goal of this BMS is to safely charge and discharge the systems battery, while providing the system with relevant data on the battery status. The BMS consists of multiple subsystems, the most important being a charger, a cell balancer, a power management system and a main control unit tying it all togheter.

Contents

1

Autonomous charging module

The Battery management system discussed in this thesis is part of a bigger autonomous charging system, split up and developed in individual parts. This autonomous charging system is being designed for a specific robot, the ZeBro (”ZEs Benige RObot”, six legged robot), a walking robot with six legs designed as a swarm robot[1]. To get an idea of the bigger picture, we will briefly walk through the general idea of the system before we start on our specific part.

The main concept of the Zebro robot is to work together in a group, or ’swarm’. To efficiently let the robots perform the task they are assigned to together, it is important to be as autonomous as possible. This also applies to charging. Autonomous charging becomes more crucial when a bigger group of robots is used. In an ideal situation autonomous charging enables the robots to continuously perform their task without human intervention.

In our group we chose to divide the system into three parts. The first group has developed a modular charging station where multiple Zebros can be charged at the same time, including the power transfer and communication to the different docks. The second group developed a wireless charging module, taking care of converting the power and transferring it wireless. The final group developed a battery management system, consisting of a charger and managing modules for the robots battery, which will be discussed in detail throughout this thesis.

The part of the autonomous charger localized in the robot itself, is expected to communicate with the other internal subsystems of the Zebro by a pre-specified central communication bus, also known as the ZebroBus.

An overview of the complete autonomous charging system can be found in figure 1.1.

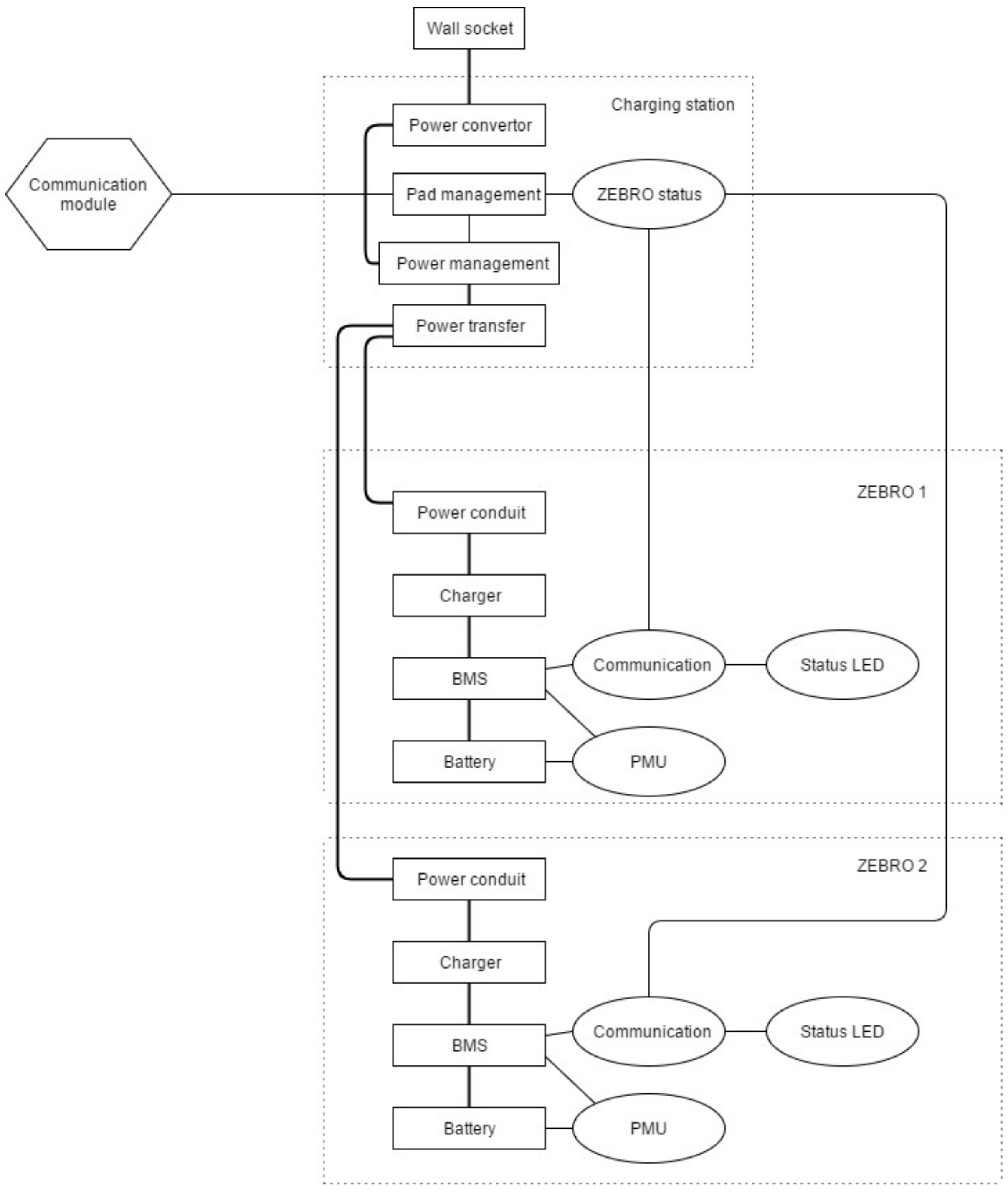


Figure 1.1: An overview of the complete autonomous charging module.

2 Introduction

Over the past two centuries, automation has played an increasing role in our society. Starting with the industrial revolution, nowadays still big steps are made in automation in many different sectors. The development of robotics also takes part in this trend. To meet the new demands of our society, all kind of robots need to become more and more autonomous, to achieve an even higher level of automation. The Zebro robot developed at TU Delft is developed around two key concepts. One is, as inspired by simple insects performing heavy jobs when working together, working with multiple robots in a group, a so called swarm. The other key concept is autonomous operation. In an ideal world, this swarm robots could perform their task for an infinite long time without any human intervention.

A next step in eliminating human intervention in whatever kind of a job a robot is doing, is autonomous charging. At the moment of writing, not much can be found in literature about autonomous, and especially wireless, charging for robots. Wireless chargers are widely available and several transfer standards are developed, but only for low-power devices, mostly smart phones. We have designed an autonomous charger specifically for the Zebro robot, but with slight adaptions also applicable for many other purposes. As stated in chapter 1, we divided this system into three subsystems (see figure 1.1), where our group designed the battery management system.

A lithium battery, like the one used in our autonomous charging system, is a very sensitive device. If it is charged too much, or discharged too much, it could easily get damaged or even explode. Also whilst using the battery, the battery cells can become unequally charged which can result in the same undesired events. Also, in any kind of robot it is important for the control system to know the state of the battery. Al this features should be taken care of in our battery management system.

This thesis will start by walking you trough our program of requirements. Based on our assignment from the Zebro project group, with some additional requirements we developed during our process, we made a clear list of what the system in the end should be able to do. Following, we will take a look at the system breakdown decisions we have made to create sub-components to satisfy these requirements.

After having specified the sub-systems and their requirements, we will consecutively discuss the most important sub-systems, starting with the charger, followed by the cell balancer, the power management system and concluding with our main control unit. In all of these sections we will briefly eloborate on what the subsystem is supposed to do, and how we chose to design it to meet this specifications. Concluding this thesis we will in the evaluation look back on our process and verify if we met our program of requirements is.

3 Program of requirements

To get a clear view of the goals that need to be achieved throughout our design process a list of requirements for the system was determined. For our BAP assignment, the autonomous charger module, a number of requirements are set by the Zebro team, which need to satisfied. Listed below are the requirements relevant for our subsystem, the Battery Management System (BMS).

- The Charger Module should be capable of measuring its current and power flow, its voltages and its internal temperature.
- The Charger Module should implement an Autonomous Module Damage Protection System (AMDPS), be capable of determining short circuits and overvoltage, preventing the module from destroying itself.
- Have very well-defined control and status interface;
  - high-level commands and status needed for the central nervous system.
  - low-level commands and debugging information.
- The system should determine and control power flow.

Complementary to this requirements from the assignment, there are requirements that are fundamental to the function of any BMS. Some more requirements arose during the meetings with the Zebro team, which were not listed in the assignment requirements documents. The resulting program of requirements is listed below, from most important to least important.

Functional requirements

1. Fit into the Zebro casing
2. Safely charge the battery with power from the wireless charging module
3. Safely discharge the battery to deliver power to the Zebro system
   - Unregulated power bus for Zebro legs
   - Regulated power line for Zebro controllers (2x Raspberry Pi Zero)
4. Estimate battery State of Charge
5. Minimize battery charging time
6. Optimize battery capacity and lifetime
7. Charge the battery with power from a wide range of sources (e.g. solar panels)

8. Estimate battery State of Health
9. Communicate with other Zebro modules

During the design process some more requirements came forward. Also, one requirement arose from the interfaces between the BMS and the wireless charging module, and had to be met to ensure successful system integration.

System requirements

1. Charge the battery from an input voltage between 8V and 32V
2. Charge the battery with a current of 6A
3. Balance battery cells
4. Deliver >12V, 100 W to the main system.
5. Deliver 5V, 3A to the Zebro controllers
6. Measure relevant system parameters
    - Charger and discharge current
    - Cell voltages
    - Battery and module temperature
7. Detect temperature, overvoltage, undervoltage and overcurrent faults and recover from these faults if possible
8. Expose system parameters to the main communication bus
    - Charge and discharge currents
    - Cell voltages
    - Battery and module temperature
    - Battery rated capacity
    - Battery SoC
    - Battery SoH
    - BMS state (charging, balancing, active, fault, etc.)
9. Allow Zebro controller to set parameters via the main communication bus
    - Bus load switch
    - Maximum charge current
    - Maximum discharge current
    - Maximum charge voltage
10. Having a debug facility

4

Top-Level System Overview

4.1. Modules

The BMS has many tasks, which can be distilled from the requirements. To ease the design of the BMS, the system is divided into modules. Each module has its own task to fulfill and requirements to meet. All modules together should also meet requirement 1:

- Fit into the Zebro casing

4.1.1. Battery

The central element in the system, of course, is the battery. While discussing with the Zebro team it became clear that for optimal functioning of the leg modules a bus voltage of >12V is needed. It was settled to use a 4-cell battery that fits in the Zebro, with the capacity as high as possible. A 4-cell 3000mAh LiPo battery that fitted neatly in the central space of the Zebro is used.

4.1.2. Main control unit

The main control unit (MCU) is responsible for controlling all the other modules. This module also communicates with the rest of the Zebro. The MCU measures system parameters and makes decisions accordingly.

This module must meet system requirements 6, 7, 8, 9 and 10:

- Measure relevant system parameters
  - Charger and discharge current
  - Cell voltages
  - Battery and module temperature
- Detect temperature, overvoltage, undervoltage and overcurrent faults and recover from these faults if possible
- Expose system parameters to the main communication bus
  - Charge and discharge currents
  - Cell voltages
  - Battery and module temperature

  - Battery rated capacity
  - Battery SoC
  - Battery SoH
  - BMS state (charging, balancing, active, fault, etc.)
- Allow Zebro controller to set parameters via the main communication bus
  - Maximum charge current
  - Maximum discharge current
  - Maximum charge voltage
- Having a debug facility

Meeting these system requirements, the system also meets functional requirements 4, 8 and 9:

- Estimate battery State of Charge
- Estimate battery State of Health
- Communicate with other Zebro modules

4.1.3. Charger module

The charger module is responsible for charging the battery from an external power supply. This module must meet system requirement 1 and 2:

- Charge the battery from an input voltage between 8V and 32V
- Charge the battery with a current of 6A

Meeting these system requirements, the system also meets functional requirements 2, 5 and 7:

- Safely charge the battery with power from the wireless charging module
- Charge the battery with power from a wide range of sources (e.g. solar panels)

4.1.4. Balancer module

The balancer module is responsible for balancing the battery cells. This module must meet system requirement 3:

- Charge the battery from an input voltage between 8V and 32V

Meeting this system requirement, the system also meets functional requirement 6:

- Optimize battery capacity and lifetime

4.1.5. Power management module

The power management module (PMU) is responsible for managing the output power of the BMS. It has to meet the overcurrent fault requirements, because the software on the MCU can not reasonably be made fast enough to prevent damage in case of a short circuit. This module must meet system requirements 4, 5 and 7:

- Deliver >12V, 100 W to the main system.
- Deliver 5V, 3A to the Zebro controllers
- Detect overcurrent faults and recover from this fault if possible

Meeting these system requirements, the system also meets functional requirement 3:

- Safely discharge the battery to deliver power to the Zebro system
  - Unregulated power bus for Zebro legs
  - Regulated power line for Zebro controllers (2x Raspberry Pi Zero)

4.2. Overview

The top-level system overview gives a clear view of the subsystems of the BMS and can be seen in figure 4.1.

Power enters the system through the wireless system or any other power source. With this power the battery is charged by the charger (chapter 5). The power is distributed to the Zebro by the Power Management Unit (chapter 7). The balancer (chapter 6) keeps the battery cells balanced to prevent damage to the battery. The system parameters are monitored by MCU (chapter 8), the 'brain' of the system. The MCU also sends control signals to all of the other subsystems and communicates with the other Zebro components.

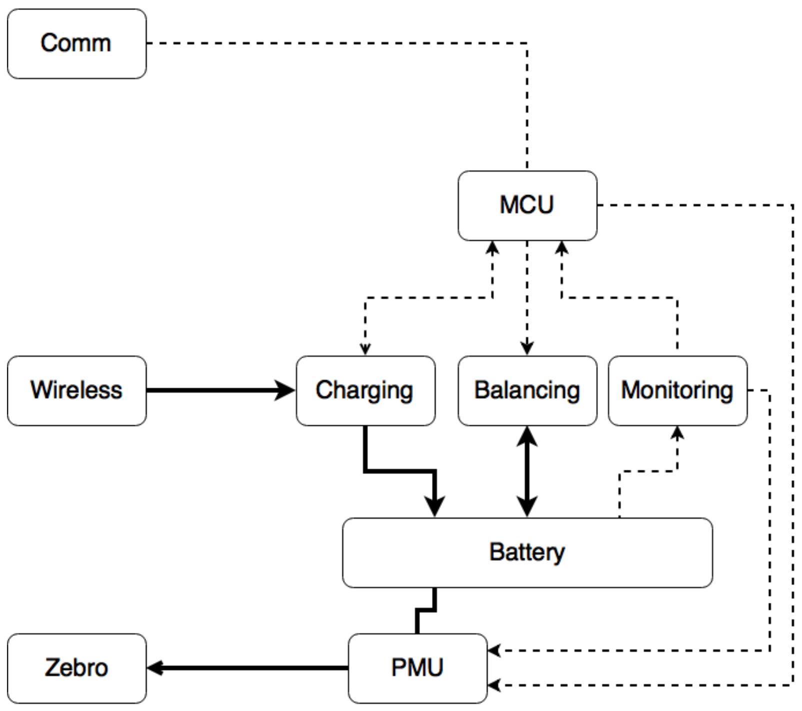


Figure 4.1: Overview of the Battery Management System. MCU is our Main Control Unit, PMU is our Power Management Unit. Bold arrows indicate power flow, where dotted arrows indicate signal flow.

5 Charger

5.1. Requirements

One of the most important functions of the BMS is charging the battery. This is the task of the charger module. The module draws power from the inductor with rectifier that is part of the power transfer system. To accommodate other charging sources, the charging module should work with a wide range of input voltages. The output current and voltage should also be adjustable, so that the charger module works with different battery packs.

The charger module has to meet these system requirements:

- Input voltage must be between 8V and 32V
- Minimize battery charging time by charging with maximum charging rate
- Charging current and voltage must be software controllable

By meeting these requirements it also meets these funtional requirements:

- Safely charge the battery with power from the wireless charging module
- Charge the battery with power from a wide range of sources (e.g. solar panels)

The charger should be able to charge a 4S 3000 mAh LiPo battery pack at 2C charging rate. This means the maximum output voltage of the charger should at least be $4.2\text{V} \cdot 4 = 16.8\text{V}$. The maximum output current should be at least $2 \cdot C = 2 \cdot 3\text{A} = 6\text{A}$. A charging rate of 2C means by definition that the battery pack is charged in half an hour, if the charging current is constant over the voltage range.

The charger must accept a voltage range of 8 V to 32 V. This way the charger can work with various low voltage sources, with an acceptable input current. The charger output voltage and current must be software controllable and measurable, to allow for end-of-charge termination and precharging in case the battery is deeply discharged. This can also be used to limit the charging rate if the input source can not supply enough energy.

5.2. Design

The design requirements state that the charger must charge as fast as possible, while also being safe. There are several algorithms to charge a battery. This list is adopted from Shen[2], with the addition of a relatively new charging algorithm: Sinusoidal ripple current charging[3].

- CC/CV (constant current/constant voltage)
- Double loop CC/CV
- Boost charger CC/CV
- Fuzzy logic CC/CV
- Grey predicated CC/CV
- (Improved) phase-locked loop CC/CV
- Multi-stage constant current
- Frequency controlled constant voltage pulse charging
- Duty controlled constant voltage pulse charging
- Constant current pulse charging
- Sinusoidal ripple current charging

Most commercially available chargers use some method of CC/CV charging, because it is relatively easy to implement and has been used for a long time. Double loop CC/CV is not a true CC/CV method, because it mimics the behaviour of a constant current source by a PI controller, reducing component costs. It is not useful to use an algorithm in this design, because the charging current must be measured to estimate the SoC of the battery during charging. The other flavours of CC/CV are attempts to decrease charging time, with varying success. Pulse charging promises faster charging times and longer battery life, but is harder to control since the battery voltage must be measured accurately by software. Sinusoidial ripple charging could improve charging times even more, but also requires even more control to charge with the right frequency.

For this design the CC/CV charging algorithm was chosen. This algorithm generally performs well is used in many designs and is easy to control. Because the charging current and voltage must be software controllable, other charging methods can be implemented within the same charger later on, if room for improvement is found. However, it is beyond the scope of this project to evaluate the optimum charging strategy.

5.2.1. Topology

The simplest possible design for a charger is a linear regulator that limits the voltage and current from a power source, charging the battery. However, this design is not efficient at all. Switch mode DC/DC converters are much more efficient. In principle, any DC/DC converter topology can be used to build a CC/CV charger. If the charger input voltage is constant and higher than the battery voltage, a buck converter is most often used, because this type of converter is relatively easy to design and stabilize.

Because one of the specifications dictates that the charger must work with input voltages that can be higher or lower than the output voltage, these DC/DC converter topologies could be used:

- Buck-Boost converter
- Full bridge converter
- SEPIC converter
- Ćuk converter

Both the Buck-Boost converter and the Ćuk converter are inverting topologies: the output voltage is negative referenced to the input. This makes these topologies harder to use for the battery charger, because there are measurements to be done at the input voltage. The SEPIC converter has the advantage over the four-switch converter that it uses only one MOSFET switch in stead of four, so it saves costs and space. Also, the input and output are isolated by a capacitor, so when the converter is not switching no current can flow from the input to the battery or vice versa. A disadvantage of the SEPIC converter is that the output current is pulsed. SEPIC converters are also generally less efficient than four-switch converters.

The SEPIC topology is chosen for the battery charger, because the space and cost restrictions are more important than efficiency. The pulsed output current is no problem because there is a battery at the converter output. In order to save even more space, a coupled inductor is used. A schematic of the SEPIC topology is shown in figure 5.1.

The controller for the SEPIC converter is the LM3481 from Texas Instruments.[4]

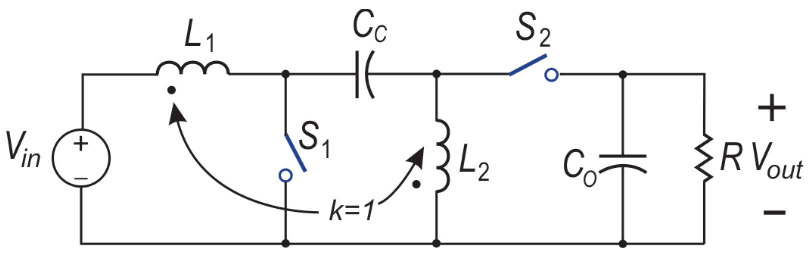


Figure 5.1: SEPIC converter topology with coupled inductor. Adopted from [5]

5.2.2. Power components

In order to select the power components, first the conditions these components must comply with are calculated. This was done using the datasheet of the converter[4] and a technical report on designing SEPIC converters[6]. The procedure laid out in these documents was followed to obtain minimum values and ratings for the components.

First, the input and output voltages and currents are selected. The output voltages are derived from a cell voltage ranging from 3 V to 4.3 V per cell. The maximum output current is the charging current.

$$\begin{aligned} V_{IN}^{min} &= 8\text{V} \\ V_{IN}^{max} &= 32\text{V} \\ V_{OUT}^{min} &= 12\text{V} \\ V_{OUT}^{max} &= 16.8\text{V} \\ I_{OUT}^{max} &= 6\text{A} \end{aligned} \tag{5.1}$$

The switching frequency is also selected. In the final prototype, the frequency will be fixed. The frequency can be altered easily by changing a resistor. By doing this, the charger can be optimized for efficiency while still staying in CCM.

$$f_s^{min} = 300\text{kHz} f_s^{max} = 1000\text{kHz} \tag{5.2}$$

From these values maximum and minimum duty cycle are calculated. The diode voltage is estimated to be 0.5V, a typical voltage drop for a schottky diode. A margin of 5% is added to allow for a bit of headroom.

$$
\begin{aligned}
D^{min} &= \frac{V_{OUT}^{min} + V_D}{V_{IN}^{max} + V_{OUT}^{min} + V_D} = \frac{12\text{V} + 0.5\text{V}}{32\text{V} + 12\text{V} + 0.5\text{V}} = 0.2669 \\
D^{max} &= \frac{V_{OUT}^{max} + V_D}{V_{IN}^{min} + V_{OUT}^{max} + V_D} = \frac{18\text{V} + 0.5\text{V}}{8\text{V} + 16.8\text{V} + 0.5\text{V}} = 0.7180
\end{aligned}
\tag{5.3}
$$

Inductor

The inductor currents. The ripple current is 40% of the maximum input current, following the rule from[6].

$$
\begin{aligned}
\Delta I_L &= 0.4 I_{OUT}^{max} \frac{V_{OUT}^{max}}{V_{IN}^{min}} = 0.46\text{A}\frac{16.8\text{V}}{8\text{V}} = 5.04\text{A} \\
I_{L,avg}^{max} &= I_{OUT} \frac{D^{max}}{1 - D^{max}} = 6\text{A}\frac{0.7180}{1 - 0.7180} = 15.23\text{A} \\
I_{L,pk}^{max} &= I_{L,avg}^{max} + \frac{\Delta I_L}{2} = 15.23\text{A} + \frac{5.04\text{A}}{2} = 17.51\text{A}
\end{aligned}
\tag{5.4}
$$

The inductor must be large enough to guarantee continuous conduction mode in all circumstances.

$$
L \geq \frac{max(V_{IN}^{min} D^{max}, V_{IN}^{max} D^{min}}{2 I^{out} f_s^{min}} = \frac{max(8\text{V} \cdot 0.7180, max(32\text{V} \cdot 0.2669}{26\text{A}300\text{kHz}} = 2.8\mu\text{H}
\tag{5.5}
$$

With these values, this coupled inductor is chosen: WURTH 74485540350. $L = 3.5\mu\text{H}$ under the average current: $L = 3.1\mu\text{H}$

Diode and MOSFET

The diode and MOSFET must each be able to handle the peak switch current, and must withstand the sum of the input and output voltages.

$$
\begin{aligned}
I_{SW}^{max} &= I_{OUT} + I_{L,pk}^{max} = 6\text{A} + 17.51\text{A} = 23.51\text{A} \\
V_{SW}^{max} &= v_{IN}^{max} + v_{OUT}^{max} = 32\text{V} + 16.8\text{V} = 48.8\text{V}
\end{aligned}
\tag{5.6}
$$

The diode chosen with these parameters is VISHAY V30DL50C-M3/I. For the choice of MOSFET it is also important to minimize both gate charge and on resistance, to minimize switching and conduction losses. The MOSFET chosen is ON SEMICONDUCTOR NTD5865NLT4G.

SEPIC capacitor

The equation is adopted from the LM3481 datasheet [4].

$$
C_S \geq= L\left(\frac{I_{OUT}}{V_I N^{min}}\right)^2 = 3.1\mu\text{H}\left(\frac{6\text{A}}{8\text{V}}\right)^2 = 1.74\mu\text{F}
\tag{5.7}
$$

The SEPIC capacitor must be rated for at least the input voltage. It must also be able to withstand large RMS currents. Therefore, a ceramic capacitor is preferred.

Output capacitor

The output capacitor converts the current ripple into a voltage ripple. The ESR and ESL of the capacitor directly control the output voltage ripple. Voltage ripple is not very important in this case, because the load is a battery which can handle large voltage ripples. To minimize board space, SMD ceramic capacitors are used which can easily handle the large ripple currents. Equation is adopted from [4].

$$I_{RMS}^{max} = \sqrt{((I_{SW}^{max})^2 - I_{SW}^{max} 2\Delta I_L + \frac{4}{3}\Delta I_L^2)(1 - D_{min}) - I_{OUT}^2} = 14.8985\text{A} \tag{5.8}$$

The converter has three 10μF capacitors on its output, which derate to 4μF when under DC bias. Total output capacitance is:

$$C_0 = 3 \cdot 4\mu\text{F} = 12\mu\text{F} \tag{5.9}$$

5.2.3. Control

Feedback compensation

The SEPIC converter contains four dynamic circuit elements contributing to its transfer function: two inductors, the SEPIC capacitor and the output capacitor. If voltage-mode feedback control would be used, this would result in a fourth order transfer function. Current mode control was invented to overcome this problem. Also, tight coupling of the inductors reduces the transfer function to second order [5].

The small signal model of the SEPIC converter is obtained by moving the second inductor and swapping the positions of the diode switch with the output capacitor. The resulting circuit has the exact same function as the original SEPIC circuit. This can be seen in figure 5.2.

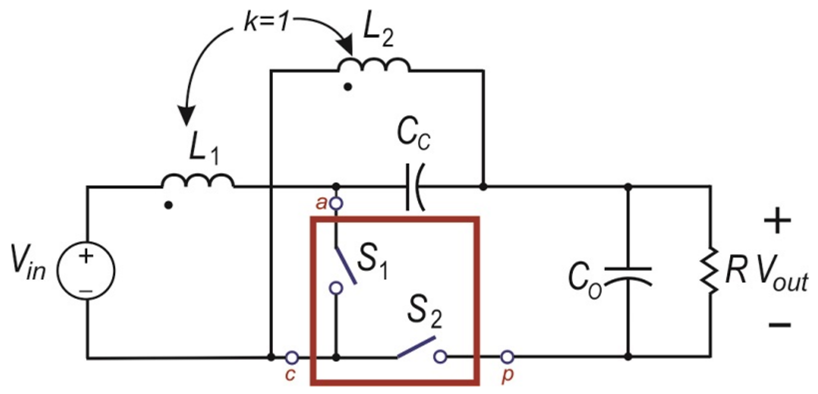


Figure 5.2: SEPIC with rearranged switch and inductor location. Adopted from [5]

The PWM switches are replaced by a PWM small signal model[7]. The coupled inductors are replaced by an equivalent circuit, consisting of a single inductor and an ideal transformer. This results in the circuit from figure 5.3.

From this circuit the control-to-output transfer functions is derived. The equation is adopted from [5].

$$\frac{V_{OUT}(s)}{d(s)} = \frac{V_{IN}}{(1-D)^2} \frac{1 - s\frac{L}{R}\frac{D}{(1-D)^2}}{1 + s\frac{L}{R}\frac{1}{(1-D)^2} + s^2\frac{LC_O}{(1-D)^2}} \tag{5.10}$$

This equation has one zero and two poles. The zero is in the right-hand plane. The poles are a complex pair when $R_L^2((1-D)^2 > \frac{L}{4C_o}$ holds for all values of R_L and D.

$$R_L^{min} = \frac{V_{OUT}^{min}}{I_{OUT}} = 2\Omega$$

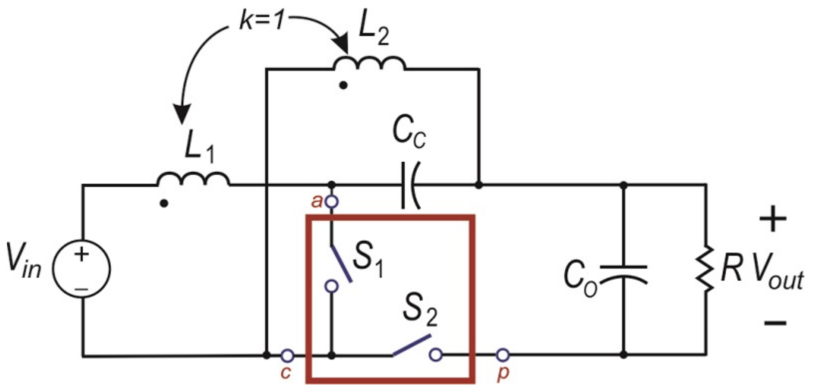


Figure 5.3: SEPIC with rearranged switch and inductor location. Adopted from [5]

So the inequality reduces to $(2\Omega)^2((1-0.7180)^2 > \frac{(3.1\mu\text{H}}{4\cdot 12\mu\text{F}}$ which is true.

$$z = \frac{(1-D)^2}{D}\frac{R_L}{L} \tag{5.11}$$

$$p = -\frac{1}{2C_oR_L} \pm i\frac{\sqrt{4LC_o(1-D)^2R_L^2 - L^2}}{2C_oR_LL} \tag{5.12}$$

Figure 5.4 shows a plot of the transfer function. For this plot the component values calculated in section 5.2.2 are used, and the other values correspond to f_z^{min} : $V_{IN} = 8\text{V}, d = 0.7180, R_L = 2\Omega$.

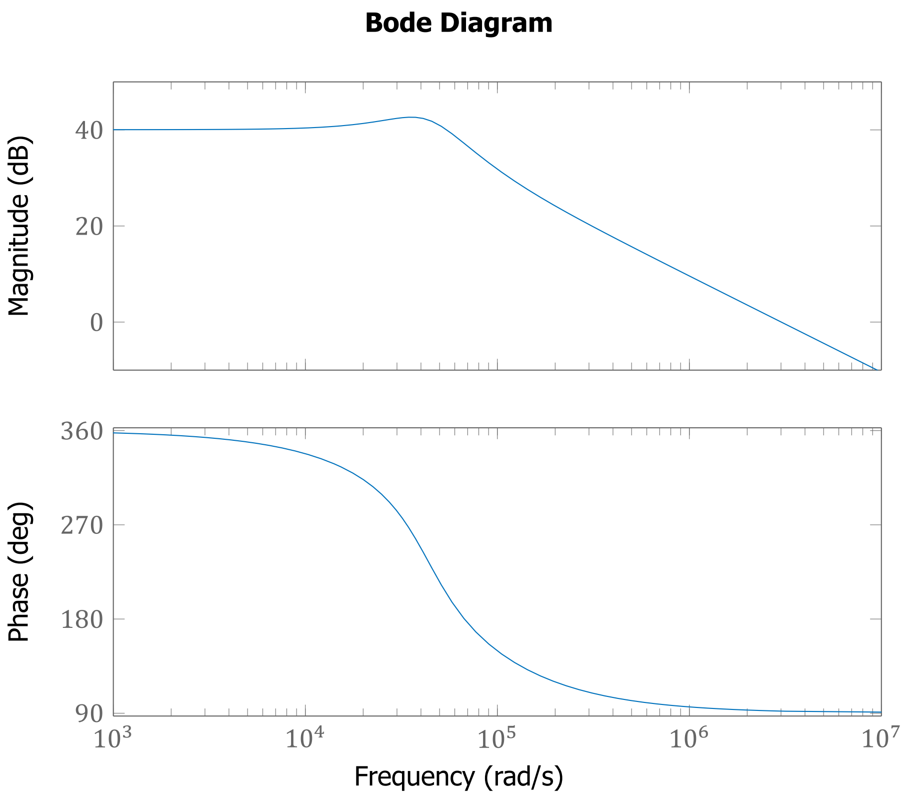


Figure 5.4: Control-to-output transfer function of the SEPIC converter

The minimum frequencies of the pole and zero are:

$$\begin{aligned} f_z^{min} &= \frac{1}{2\pi}|z| = \frac{1}{2\pi}\frac{(1-D^{max})^2}{D^{max}}\frac{R_L^{min}}{L} = \frac{1}{2\pi}\frac{(1-0.7180)^2}{0.7180}\frac{2\Omega}{3.5\mu\mathrm{H}} = 10.1\mathrm{kHz} \\ f_p^{min} &= \frac{1}{2\pi}|p| = \frac{1}{2\pi}\frac{(1-D)^2}{C_oL} = \frac{1}{2\pi}\frac{(1-0.7180)}{\sqrt{12\mu\mathrm{F}\cdot 3.1\mu\mathrm{H}}} = 7.4\mathrm{kHz} \end{aligned} \tag{5.13}$$

The crossover frequency is set as $\frac{1}{10}$ of the right half plane zero:

$$f_c = \frac{1}{10}f_z^{min} = \frac{1}{10}\cdot 10.1\mathrm{kHz} = 1.0\mathrm{kHz} \tag{5.14}$$

The transfer function of the error amplifier is shown below [8]. R_{OUT} and g_m are specified in the LM3481 datasheet [4] and are 152 kΩ and 450 μmho respectively.

$$\begin{aligned} A_{comp} &= A_{EA}\frac{1+\frac{s}{\omega_z}}{1+\frac{s}{\omega_p}} \\ A_{EA} &= g_mR_{OUT} \\ \omega_z &= \frac{1}{R_cC_c} \\ \omega_p &= \frac{1}{R_{OUT}C_c} \end{aligned} \tag{5.15}$$

First, the compensation pole is placed. Therefore the open loop transfer function without the compensation poles and zeroes is analyzed. The amplitude of the transfer function in the region around the crossover frequency is almost 55 dB. Therefore, the compensation pole needs to be placed at a frequency at least $\frac{55dB}{20dB} = 2.75$ decades lower than the crossover frequency ($\omega_p = 2\pi\frac{1}{10^{2.75}}f_c = 11.25\mathrm{rad/s}$) With this information C_c is calculated.

$$C_c = \frac{1}{R_{OUT}\omega_p} = \frac{1}{152\mathrm{k}\Omega\cdot 11.25\mathrm{rad/s}} = 580\mathrm{nF} \approx 620\mathrm{nF} \tag{5.16}$$

The system with just the compensation pole is shown in figure 5.5. It is clear that the amplitude is below 0dB before the RHP zero occurs at 1 kHz.

Next the compensation zero is added to increase the phase margin at the crossover frequency. With this information R_c is calculated.

$$R_c = \frac{1}{C_c\omega_c} = \frac{1}{620\mathrm{nF}\cdot\frac{1}{2\pi}1.0\mathrm{f/s}} = 25.4\Omega \approx 27\Omega \tag{5.17}$$

The total open loop transfer function is shown in figure 5.6. The closed loop transfer function is shown in figure 5.7.

Adjustable CC/CV

In order to control both the output voltage and the output current of the converter, the feedback loop has to be modified. Also, the voltage and current setpoints have to be controllable from the MCU. To achieve this, the voltage and current signals are compared with the setpoints from the MCU by comparators. If the voltage or current is too high, the comparator turns on a transistor which injects current in the feedback pin, increasing it's voltage and forcing the controller to lower the duty cycle. The reduced duty cycle results in a reduced output voltage. The bandwidth of this system is much lower than the switching frequency to avoid oscillations. Also, it prevents the overvoltage protection from activating, which will shut down the converter. A schematic of the system is shown in figure 5.8.

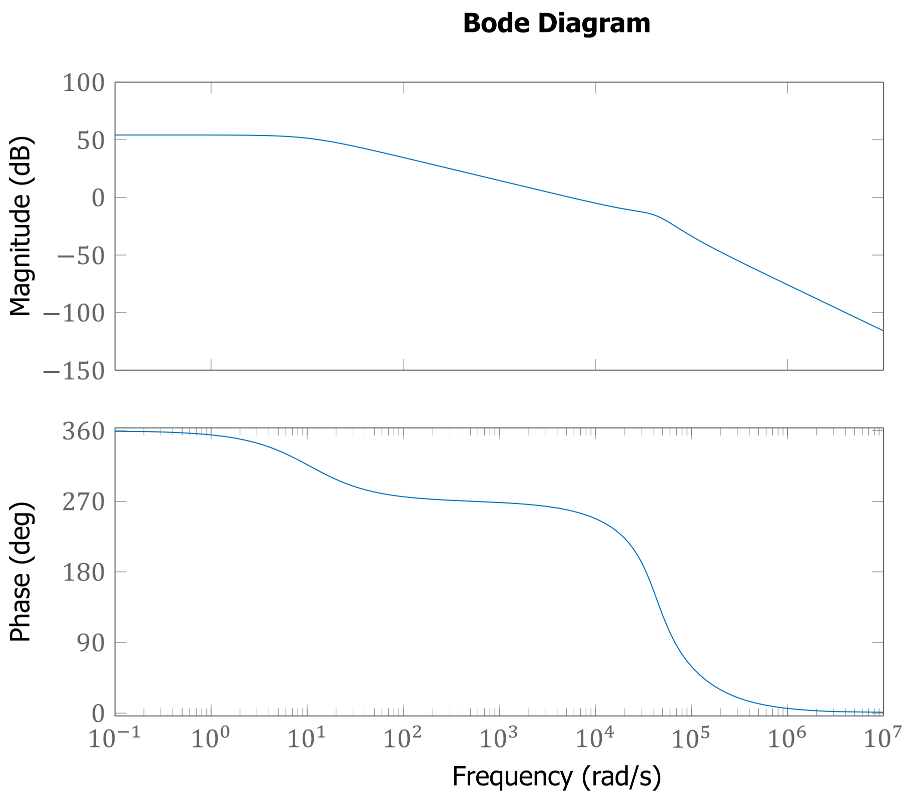


Figure 5.5: Control input-to-output transfer function of the SEPIC converter, with compensation pole

Overcurrent protection

The LM3481 has a built-in overcurrent protection feature that shuts off the controller if the current through the switch is too high. The current is measured by a resistor between the MOSFET source and ground. The current sense pin threshold voltage is 220mV. The current limit should operate when the current is twice the expected peak switch current. The value of the resistor is then:

$$R_{SENSE} = \frac{V_{SENSE,th}}{2I_{SW}^{max}} = \frac{220\text{mV}}{2 \cdot 23.5\text{A}} = 4.67\text{m}\Omega \approx 5\text{m}\Omega \tag{5.18}$$

5.3. Validation

The charger system is too complex to simulate with a circuit simulator. Also, building a prototype would take a lot of time and can only be tested with low power. Therefore we decided to make a complete design of the SEPIC converter on a PCB and then start testing. However, the design is not finished as of now, so no testing has yet been done.

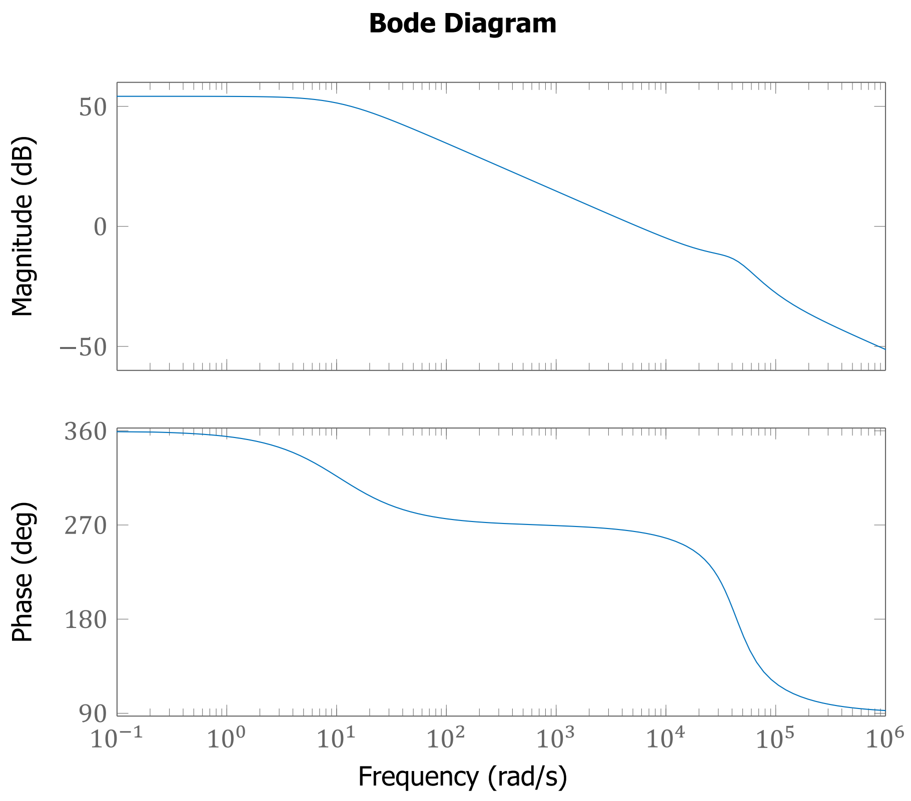


Figure 5.6: Control input-to-output transfer function of the SEPIC converter, with compensation

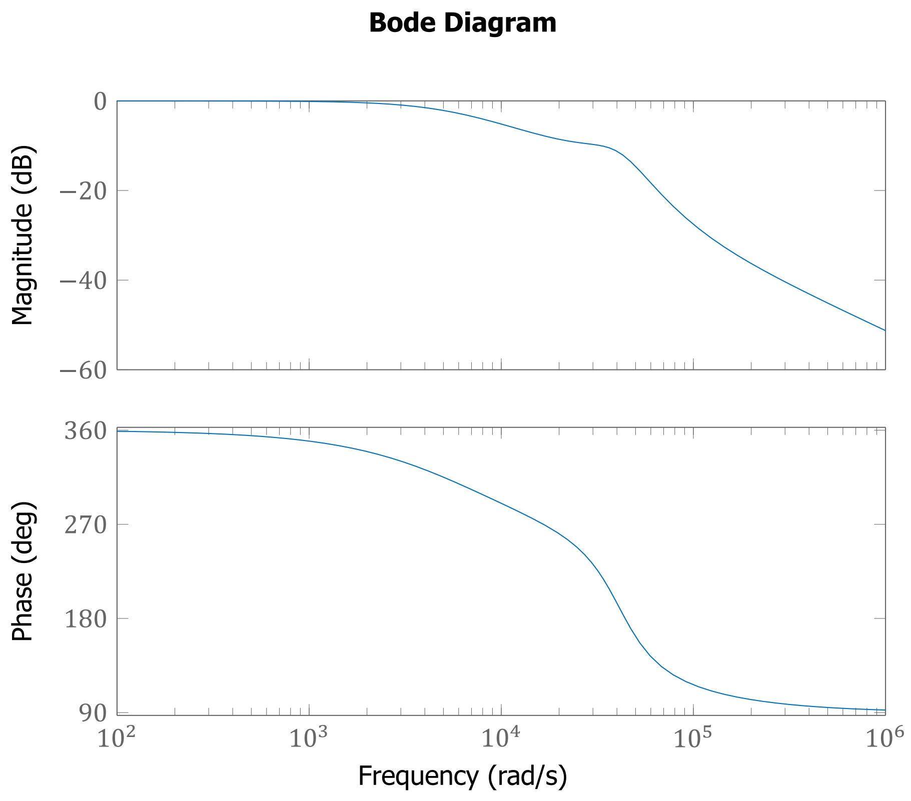


Figure 5.7: Closed loop control input-to-output transfer function of the SEPIC converter, with compensation

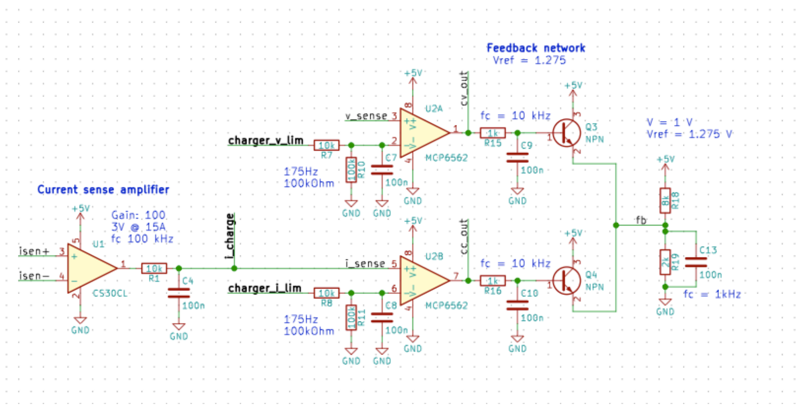


Figure 5.8: Adjustable CC/CV circuit schematic

6

Battery balancer

6.1. Requirements

During operation a battery can become unbalanced. This is due to the fact that the cells of the battery are not exactly the same. Differences in capacity and resistance result in cells with different SoC. This can result in overvoltage when charging or undervoltage when discharging. Both conditions cause damage to the battery cell and, in case of lithium cells, this can even result in fire. A way to avoid this is by charging every cell individually. However, this requires an expensive parallel charger. Because the currents required to keep the cell balanced are much lower than the charging current, it is often better to use one mass charger to charge a battery pack and use a balancer to prevent battery cell damage by equalizing cell state of charge.

The balancer has one task as laid out in the system requirements: Balance battery cells. By doing this, it meets a functional requirement: Optimize battery capacity and lifetime.

6.2. Topology

6.2.1. Overview

Most cell balancing solutions simplify the task of SoC equalizing to cell voltage equalization. The cell voltage is a good indication of SoC, but it requires the conditions of the cells in the battery pack to be roughly the same, in particular the temperature. Cao [9] divides battery balancers in three categories based on circuit topology:

Shunting topologies

The simplest active shunting methods use a dissipating component to remove energy from the highest voltage cells. This can be done digitally with a switched resistor or by using an analog controlled transistor. Non dissipative shunting methods use DC voltage converter topologies to transfer energy between cells.

Shuttling topologies

This category consists of varieties of the switched capacitor topology. A capacitor is switched between higher- and lower voltage cells, transferring energy between them. A distinguishing feature of these types of balancers is that they do not need cell voltage measurements to function.

Isolated energy converter

Balancers in this category use isolated DC/DC converters to transfer energy from cells. These balancers can use multiple converters multi-winding transformers or switched transformers to transfer energy between cells. Balancers from this category usually require complex control and are rather expensive.

6.2.2. Comparison

The comparison in table 6.1 is adopted from [9]. Control complexity was added to this comparison instead of Modular design capability.

Table 6.1: Comparison between balancing methods

Balancing method	Nature	Components for n cell string	Effective period	Control complexity
Dissipative shunt resistor	Shunting	n switches, n resistors	Charging	Simple
Analog shunting	Shunting	n transistors	Charging	Very simple
PWM controlled shunting	Shunting	2(n-1) switches, n-1 Inductors	Charging	Complex
Boost shunting	Shunting	n switches, n Inductors	Charging	Moderate
Complete shunting	Shunting	2n switches, n Diodes	Charging	Simple
Switched capacitor	Shuttling	2n switches, n-1 Capacitors	Always	Very simple
Single switched capacitor	Shuttling	2n switches, 1 capacitor	Always	Moderate
Step-up converter	Isolated	n isolated boost converters	Charging	Complex
Multi-winding transformer	Isolated	1 n winding transformer	Charging	Very complex
Ramp converter	Isolated	1 n/2 winding transformer	Charging	Very complex
Multiple transformer	Isolated	n transformers	Charging	Complex
Switched transformer	Isolated	n+3 switches 1 transformer	Charging	Moderate
Resonant converter	Isolated	2(n-1) switches, 2n inductors	Charging	Simple

The switched capacitor design was chosen, because it can provide relatively high balancing power, while being very easy to control.

6.3. Design

In figure 6.1 a switched capacitor balancer for a four-cell battery is shown. Switches are controlled by a PWM signal to obtain the shuttle behaviour. In the practical design the SPDT switches are implemented using power MOSFETS. The control PWM signal is generated by the MCU.

6.3.1. Analysis of simple model

Because the balancing system is too complex to analyse analytically, the analysis has been done for the most simple case: two cells and one capacitor. The results from this analysis are then generalized to the whole system. For the analysis it is assumed that all switch resistances and capacitors are the same and time-invariant. This analysis also does not take into account switching effects and start-up behaviour, i.e. the system is supposed to be in equilibrium.

The resistors can be replaced by a equivalent resistor: $R = 2R_{sw} + R_{ESR}$. Now a simple RC network is left. Solving the differential equations gives the capacitor voltages in both situations.

$$V_C(t) = \begin{cases} V1 + (V_C(t=0) - V_1)e^{\frac{-t}{RC}}, & 0 \le t < \frac{1}{2}T \\ V2 + (V_C(t=\frac{1}{2}T) - V_2)e^{\frac{-t+\frac{1}{2}T}{RC}}, & \frac{1}{2}T \le t < T \end{cases} \tag{6.1}$$

The capacitor voltage at the end of half-cycle A is the same as the voltage in the beginning of the other half-cycle. Setting $t = \frac{T}{2}$, the equations can be solved for the capacitor voltages at the switching moments.

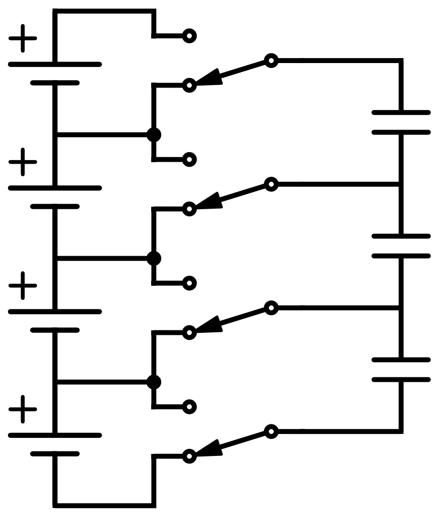


Figure 6.1: Circuit diagram of the balancer

$$\begin{cases} V_C(t=\frac{T}{2}) = V_1 + e^{\frac{-T}{2RC}}(V_C(t=0) - V1) \\ V_C(t=0) = V_C(t=T) = V_2 + e^{\frac{-T}{2RC}}(V_C(t=\frac{1}{2}T) - V2) \end{cases} \tag{6.2}$$

$$\begin{cases} V_C(t=0) = \frac{V_2 + e^{\frac{-T}{2RC}} V_1}{1+e^{\frac{-T}{2RC}}} \\ V_C(t=\frac{T}{2}) = \frac{V_1 + e^{\frac{-T}{2RC}} V_2}{1+e^{\frac{-T}{2RC}}} \end{cases} \tag{6.3}$$

Equations for capacitor voltage, combining equations 6.1 and 6.3

$$V_C(t) = \begin{cases} V_1 + \frac{V_2 - V_1}{1+e^{\frac{-T}{2RC}}} e^{\frac{-t}{CR}}, & 0 \le t < \frac{1}{2}T \\ V_2 + \frac{V_1 - V_2}{1+e^{\frac{-T}{2RC}}} e^{\frac{-t+\frac{1}{2}T}{RC}}, & \frac{1}{2}T \le t < T \end{cases} \tag{6.4}$$

Calculate power for V1, V2, R, C:

$$I(t) = \begin{cases} -\frac{1}{R} \frac{V_2 - V_1}{1+e^{\frac{-T}{2RC}}} e^{\frac{-t}{CR}}, & 0 \le t < \frac{1}{2}T \\ -\frac{1}{R} \frac{V_1 - V_2}{1+e^{\frac{-T}{2RC}}} e^{\frac{-t+\frac{1}{2}T}{CR}}, & \frac{1}{2}T \le t < T \end{cases} \tag{6.5}$$

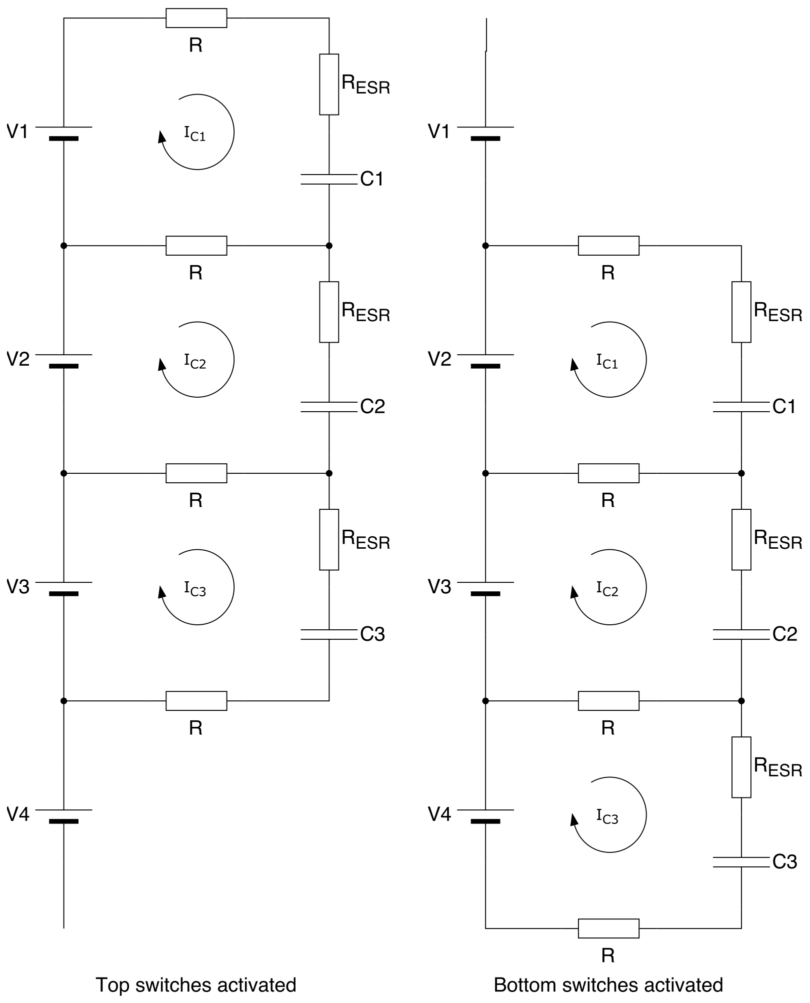


Figure 6.2: Working of the balancer

$$\begin{aligned}
P_{V_1,avg} &= \frac{1}{T}\int -I(t)\cdot V_1 dt = -\frac{C}{T}\tanh\left(\frac{T}{4CR}\right)V_1(V_2-V_1)\\
P_{V_2,avg} &= \frac{1}{T}\int -I(t)\cdot V_2 dt = -\frac{C}{T}\tanh\left(\frac{T}{4CR}\right)V_2(V_1-V_2)\\
P_{R,avg} &= \frac{1}{T}\int I(t)^2\cdot R dt = \frac{C}{T}\tanh\left(\frac{T}{(4CR}\right)(V_1-V_2)^2\\
P_{C,avg} &= \frac{1}{T}\int I(t)\cdot V_C dt = 0
\end{aligned} \tag{6.6}$$

Efficiency is only dependent on cell voltages. When cells are balanced ($V_1 = V_2$), efficiency is 100%.

$$\eta = -\frac{P_{V_1,avg}}{P_{V_2,avg}} = \frac{min(V_1,V_2)}{max(V_1,V_2)} \tag{6.7}$$

This equation can be restructured with $V1 = V_{nom} + \frac{1}{2}\Delta V$ and $V2 = V_{nom} - \frac{1}{2}\Delta V$. Also use the assumption $\Delta V << V_{nom}$.

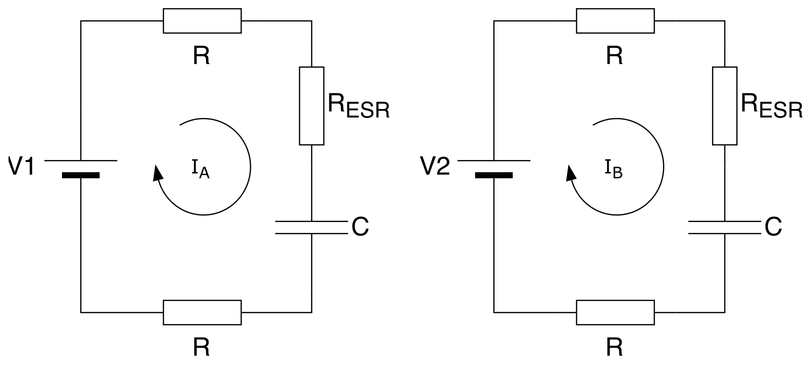


Figure 6.3: Working of the simplest case of a balancer

$$\eta = \frac{V_{nom} - \frac{1}{2}\Delta V}{V_{nom} + \frac{1}{2}\Delta V} \approx 1 - \frac{\Delta V}{V_{nom}} \tag{6.8}$$

Although the balancer has losses, these losses contribute to balancing the cells. Also, because $\Delta V << V_{nom}$, the losses are very limited compared to the balancing power. Using this assumption:

$$P_{V_1,avg} = P_{V_2,avg} = -\frac{C}{T}\tanh\left(\frac{T}{4CR}\right)V_{nom}\Delta V \tag{6.9}$$

The absolute value of the balancing current then becomes:

$$I_{avg} = \frac{C}{T}\tanh\left(\frac{T}{4CR}\right)\Delta V \tag{6.10}$$

By substituting $\epsilon = \frac{T}{C}$, the maximum current can be calculated.

$$I_{avg,max} = \lim_{\epsilon\to 0}\frac{1}{\epsilon}\tanh\left(\frac{\epsilon}{4R}\right)\Delta V = \frac{\Delta V}{4R} \tag{6.11}$$

This result makes sense, because when the switching period and capacitance are optimized, the circuit can be seen as the two battery cells connected by twice the equivalent resistance of the switch and the capacitor, resulting in current $I = \frac{\Delta V}{2R}$. The extra factor $\frac{1}{2}$ is because current is flowing only half the time.

This simple model assumed that there were no switching losses. In reality there needs to be some dead-time while one set of MOSFETs is allowed to turn off, while the other set is not yet turned on. This adds a correction factor to the maximum achievable current:

$$I_{max,corr} = \frac{T}{T + T_{dead}} I_{avg,max} \tag{6.12}$$

6.3.2. Model generalization

The simple model from the previous section can now be generalized to the 4-cell battery used by the deci Zebro. All components are assumed linear. This means that contributions of the different battery cells can be analyzed individually. Then only two options remain: one of the outer cells has a voltage difference with the rest of the battery or one of the inner cells has a voltage difference.

In the case that V1 is different from the other cell voltages, only C1 will contribute to the net energy transfer. C2 and C3 are always connected to cells with the same voltage. By setting these voltages to 0 Volt, the circuit diagram can be reduced to the diagram in figure 6.4. From this figure it is clear that the only difference with the simple scenario is the resistance of the RC circuit. The equivalent resistance in this case is:

$$R_{eq} = 1\frac{1}{3}R_{sw} + 1\frac{5}{9}R_{ESR} - \frac{R_{ESR}^2}{2(R_{sw}+R_{ESR})} - \frac{R_{ESR}^2}{18(3R_{sw}+R_{ESR})} \approx 1\frac{1}{3}R_{sw} + 1\frac{5}{9}R_{ESR} \tag{6.13}$$

The same can be done for the case where one of the middle cells has a different voltage (figure 6.5). For the current through C1, the equivalent resistance is the same as in equation 6.13. For the current throught C2 the resistance is:

$$R_{eq,C2} = 2R_{sw} + R_{ESR} - \frac{2R_{sw}^2}{2R_{sw}+R_{ESR}} \approx 2R_{sw} + R_{ESR} \tag{6.14}$$

The total equivalent resistance is:

$$R_{eq} = \frac{R_{eq,C1}R_{e1,C2}}{R_{eq,C2}+R_{eq,C2}} = \frac{4}{7}R_{sw} + \frac{38}{49}R_{ESR} - \frac{3(9R_{ESR}^3 + 22R_{ESR}^2R_{sw})}{49(2R_{ESR}^2 + 8R_{ESR}R_{sw} + 7R_{SW}^2)} \approx \frac{4}{7}R_{sw} + \frac{38}{49}R_{ESR} \tag{6.15}$$

From these equations the ration between currents from inner cells and current from outer cells can be derived:

$$\kappa = \frac{R_{eq,outer}}{R_{eq,inner}} = 2\frac{1}{3} - \frac{R_{ESR}}{2(R_{ESR}+R_{sw})} - \frac{R_{ESR}}{18(R_{ESR}+2R_{sw})} + \frac{2R_{ESR}^2 + R_{ESR}R_{sw}}{6(R_{ESR}^2 + 4R_{ESR}R_{sw} + 2R_{sw}^2)} \approx \frac{7}{3} \tag{6.16}$$

The current from inner cells is approximately 2.33 times the current from the outer cells, for the same voltage difference.

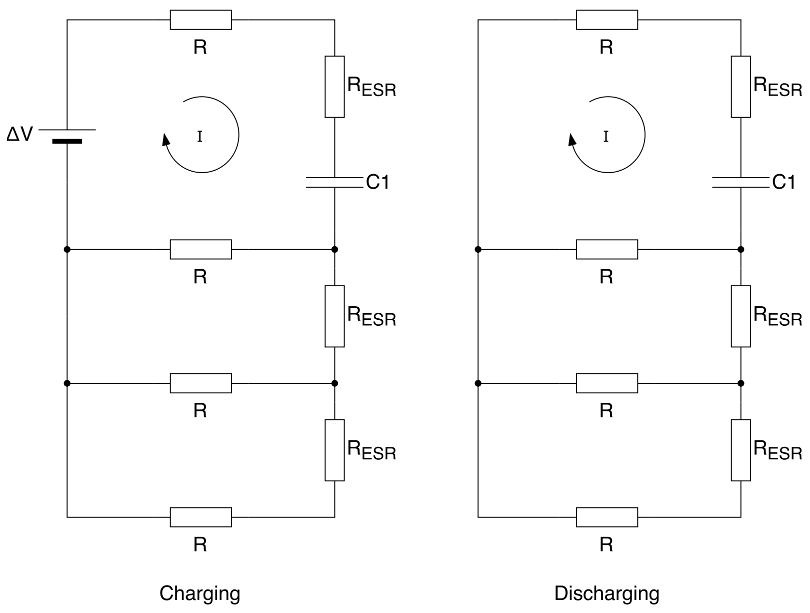


Figure 6.4: Reduced circuit for 4 cell balancer with imbalance in cell 1

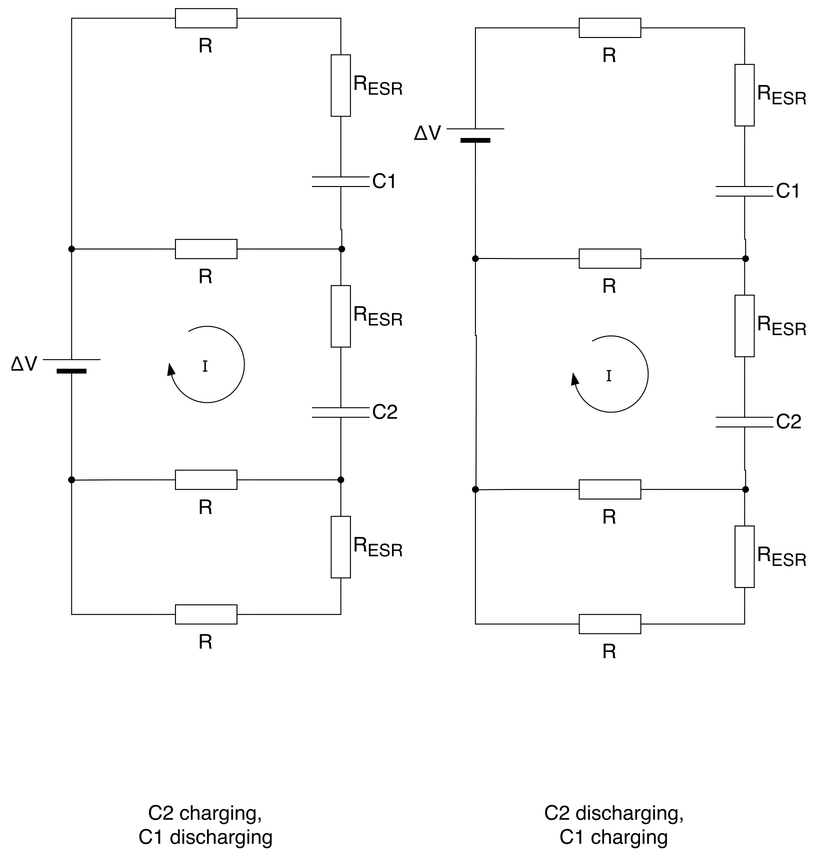


Figure 6.5: Reduced circuit for 4 cell balancer with imbalance in cell 2

6.3.3. Implementation

The battery balancing connector is rated for 3 A. This will be the current the balancer design is aiming for. The maximum imbalance voltage is assumed to be 0.1 V. This means the maximum equivalent resistance is $R_{eq,max} = \frac{\Delta V}{4I_{max}} = \frac{0.1\text{V}}{2 \cdot 3\text{A}} = 8.3\text{m}\Omega$ (see equation 6.11). For R_{ESR}, a value of 1 mΩ is estimated, assuming multiple ceramic capacitors in parallel. Solving equation 6.15 for R_{sw} gives $R_{sw} = 13\text{m}\Omega$. The drain-source voltage needs to be at least 20V because of the battery voltage, and the gate must also be able to withstand this voltage. The threshold voltage must be lower than the lowest cell voltage. Using these criteria a MOSFET is selected with $V_{th} = -1.8\text{V}$, $R_{DS(on)} = 10\text{m}\Omega$ and $Q_c = 35\text{nC}$.

The gates of the MOSFETS are pulled low by other transistors which are controlled by the MCU. The gates are pulled high by a resistance. This causes the MOSFETs to switch off much slower than they switch on. The dead time is therefore determined by the pullup resistors and the gate charge.

$$T_{dead} \approx 5\tau = 5\frac{RQ_c}{V_{bat} + V_{th}} = 5\frac{1\text{k}\Omega \cdot 35\text{nC}}{14.4\text{V} - 1.8\text{V}} = 14\mu\text{s} \tag{6.17}$$

Combining this, the total equation that needs to be solved is:

$$I_{max} = \frac{T}{T + 2T_{dead}} \frac{C}{T} \tanh\left(\frac{T}{4CR}\right) \Delta V = 3\text{A} \tag{6.18}$$

R_{eq} is equal to $\frac{4}{7}R_{sw} + \frac{38}{49}R_{ESR} = 6.5\text{m}\Omega$. Using numerical tools, a possible solution is found: $T = 140\mu\text{s} \wedge C = 10\text{mF}$

The capacitance needed is very high, and space is limited. Therefore, the implementation process is redone, but this time starting with a known capacitance which is as large as possible given a space limitation. For this design the capacitor size is limited to 6 (2 per phase) SMD aluminum electrolytic cans and a few SMD MLCC capacitors. The electrolytic capacitors' function is to store the bulk of the energy. The ceramic caps are there to absorb voltage spikes that occur because of the hard switching. The capacitors used in the implementation are have $1000\mu\mathrm{F}$ and have an ESR of 66 $\mathrm{m}\Omega$.

With these capacitors, equation 6.18 can be solved to obtain the optimum frequency. R_{eq} is now equal to $\frac{4}{7}R_{sw}+\frac{38}{49}R_{ESR} = 26\mathrm{m}\Omega$. The total capacitance is 2 mF. Using a numerical solver, the optimum period is found: T = 95.5 μs with a current of 0.78 A. The frequency is $f = \frac{1}{T+T_{dead}} = \frac{1}{95.5\mu s+14\mu s} \approx 9\mathrm{kHz}$. Alternatively, the performance of the balancer could be expressed in $\frac{I}{\Delta V}$:

$$\frac{I}{\Delta V} = \frac{T}{T+2T_{dead}}\frac{C}{T}\tanh\left(\frac{T}{4CR}\right) = 7.8\mathrm{S} \tag{6.19}$$

6.4. Validation

The implementation described in the previous sections was simulated using a SPICE simulator. The simulation results are in table 6.2. As expected the middle cells have a higher balancing current than the outer cells. All values are much lower than calculated, because the MOSFET on resistance specified in the datasheet is measured with other conditions than this circuit has. Other differences could be because of the different gate-source voltages of the MOSFETS, which also changes the on resistance.

Table 6.2: Balancer simulation results

Imbalanced cell	Balancing current
Cell 1 (top)	344 mA
Cell 2	550 mA
Cell 3	490 mA
Cell 4 (bottom)	300 mA

7
Power Management

7.1. Requirements

The Power Management Unit routes the power from the battery to the Zebro modules. It supplies both the battery management system itself and the other systems of the robot with power. Also, the over-current protection is handled in this part of the system. As stated in chapter 8, most of the fault monitoring and protection is managed in the main control unit. However, the most crucial fault protection protecting the main system from over-current (for example in case of a short circuit) is handled also in the power management. We choose to do this since it would take too much time to let this be controlled by the MCU and it would be already too late to prevent damage. Following from this we reasoned the following specifications.

The PMU has these tasks as laid out in the system requirements:

- Deliver >12V, 10 A to the main system
- Deliver 5V, 3A to the Zebro controllers
- Detect overcurrent fault and recover from this fault

By doing this, it meets the following functional requirements:

- Safely discharge the battery to deliver power to the Zebro system
  - Unregulated power bus for Zebro legs
  - Regulated power line for Zebro controllers (2x Raspberry Pi Zero)
- The Charger Module should implement an Autonomous Module Damage Protection System (AMDPS), be capable of determining short circuits and overvoltage, preventing the module from destroying itself

7.2. Design

The main bus is directly connected to the battery via a load switch. The load switch is controlled by the MCU, which can turn it on or off. The MOSFET turn-on is controlled by a dv/dt limit network to pre-charge the bus capacitance. A diode is included in this network to ensure fast turn-off.

The 5V bus for the Zebro controllers needs a converter to reduce the voltage of the battery to 5V. To achieve this, a step-down (buck) converter is used. Switching converters, like buck converters, provide a much higher efficiency than linear regulators, which are simpler voltages that step-down the voltage by dissipating power as heat. The converter is designed using the reference design from the datasheet [10].

The BMS itself also needs power. The MCU and peripheral IC's need 5V and draw a maximum current of 200 mA. The BMS is the first thing to turn on, so it cannot be powered from the 5V buck converter which is switched on by the MCU. Because of the low current drawn by the BMS (<200mA), to regulate the battery voltage to 5V a simple LDO is used.

7.2.1. Over-current protection

The most important fault from which the system has to recover is overcurrent or short-circuit. The current is measured using a hall-type current sensor. The measured current is passed to a low pass filter to prevent small current spikes from triggering an overcurrent condition. That signal is fed into a comparator, where it is compared to a set-point controlled by the MCU. If the current is higher than the setpoint, the comparator will signal the load switch to switch the load off. The built-in automatic restart will then turn the load switch on again after 200ms. In the meantime, the MCU is notified of the overcurrent shutdown and could prevent restart by disabling the load switch.

The circuit for the load switch with the over-current protection is shown in figure 7.1.

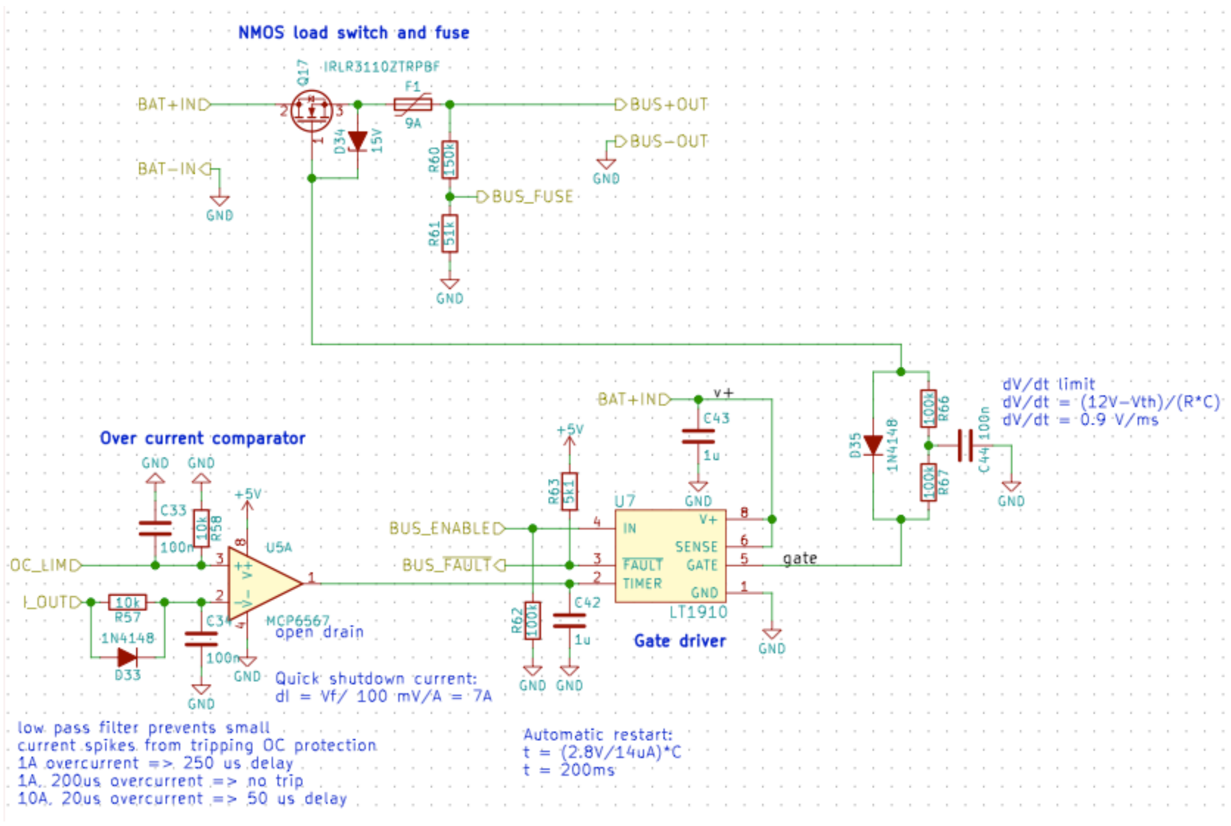


Figure 7.1: Load switch circuit schematic

To improve the safety of the system both the 5V and the bus output are protected by a PTC fuse. The MCU can detect if the fuse has activated, and shut down the output to allow the fuse to cool down and the system to recover.

8

Main Control Unit

8.1. Requirements

The last subsystem in this report is the Main Control Unit. The main control unit can be seen as the brain of our system. Its main task is to report to the rest of the robots system. It mostly supplies or processes data to or from our other systems discussed in the chapters before. Also all of the fault protection, except the main bus over-current protection, is done by the MCU.

Based on the requirements discussed in chapter3, with some requirements added to achieve functioning of other subsystems, for this communication and the needs for our subsystems, a list of requirements for the MCU has been determined:

1. Communicate with the central comminucation bus (requirement 9)
2. Generate two PWM signals for the cell balancer
3. Monitor system values and potential fault signals and values (system requirement 8)
4. Allow the main bus set system values (requirement 9)
5. Initialize sub-systems on start-up
6. Force sub-systems off in case of an error (system requirement 7)
7. Estimate batteries State of Charge (requirement 8)
8. Estimate batteries State of Health (system requirement 8)

8.2. Hardware

A microprocessor is used to process al the input signals, set all the necessary output signals and set-up communication lines. We choose an ATMEGA324PA, since we were already slightly familiar with the working environment of this microprocessor and it matched the amount of needed analog and digital I/O pins, I2C- and SPI ports and PWM generators. To achieve the maximum system speed, an external crystal resonating at 20 MHz is used to use the ATMEGA at its maximum clock frequency [11].

We added a capacitor to the supply pin of our microprocessor to make a regulated shut-down possible. When the battery is disconnected, the capacitor will still have enough charge to let the MCU detect this and handle the shut down state routine as we will discuss later.

Not all the signals can directly be read out by the microprocessor since it can only handle signal values between 0 and 5 Volt. To make sure all the measured values stay below this maximum, eight voltage dividers were implemented into the MCU subsystem. This should result in no signals being able to get a value larger than this maximum. To make sure to protect our microprocessor in the case of an excessive peak, also TVS diodes to all inputs that could possibly excess where added.

The over-current limit in the power management system, even as the current and voltage limits the system needs to set for the charger to achieve CC-CV charging, need to be rather accurate. To achieve this, we added two external Digital-to-Analog converters to the MCU which will be controlled through the SPI terminals of our ATMega chip.

8.3. Software

8.3.1. Top Level

As a first step in designing the software to run our system, a top-level designed was made, which resulted in a flowchart which is shown in figure 8.1. The Zebro robot factually has two operating states, charging and discharging. We decided to add a fault state to realize the fault detection and protection and a state for balancing when needed. Reading out the systems parameters is necessary for every routine as well as updating the systems communication bus. Depending on the state, one of the possible loop routines is executed.

As can be seen from this flowchart, on start up the MCU always start with an initialization sequence. After initializing, the system will enter the loop. This loop always starts with reading out all of the systems digital and analog values. For convenience an overview of all the MCU's in- and outputs is included in appendix A. After finishing, it will go on to determine the operating state the system is in. The four possibilities are all will be elaborated in this chapter. It is important to note the charging state has two sub-states (pre-charging and CC/CV charging), as can be seen from the flowchart. After finishing, the program will update the zebro bus information register (or interrupt in case of a fault) and continue to start again. When the system is turned off (by a hardware switch disconnecting the battery) this will be noted in the determine state phase and the program will run a shut down sequence before the capacitor installed for this occasion runs empty and the chips power is cut off.

As we can also see from the top of figure 8.1, there is an Ah integration algorithm triggered by a timed interrupt. This sequence is implemented for the State-of-Charge (further mentioned as SoC) estimation needed to know the battery state.

8.3.2. Initializing

Before start the software loop starts, the system first has to initialize some of the systems values, where the most important ones are showed in figure 8.2. To maintain the SoC value from the last time the system was active, the system reads it from the EEPROM memory where it was written on shutdown. More about the shutdown sequence and SoC estimation follow in this chapter. Whilst initializing the 5V system bus is also enabled, since it should be active all the time during operation, except in case of an error. At last in this sequence the system sets the external DACs supplying the over-current limit to the hardware over-current protection and the voltage- and current limits to the charger. After finishing this sequence, the system enters the loop and starts with reading out the systems signal values.

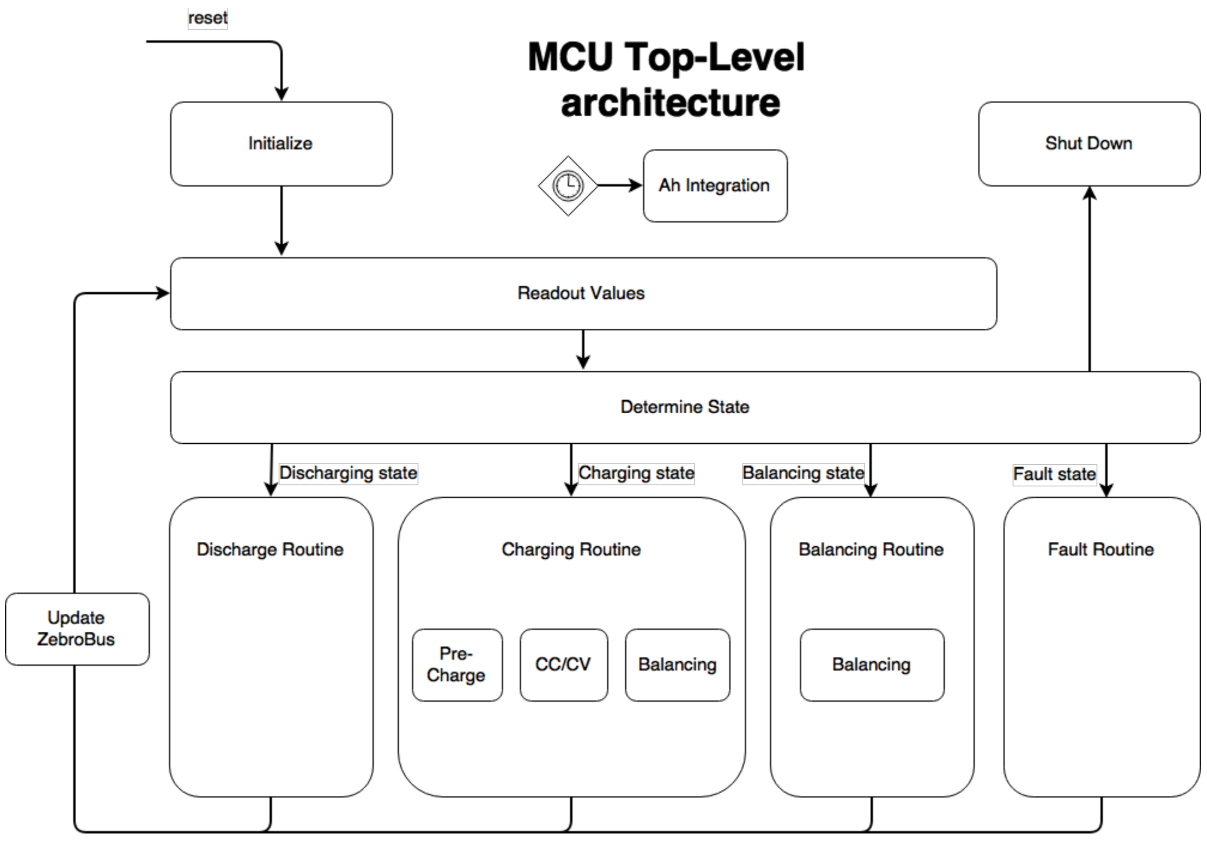


Figure 8.1: Flowchart of the Top-Level software architecture.

8.3.3. Determining the state

In every cycle the system checks again which states routine it should follow. How this is done, can be seen in figure 8.3. The flowchart starts with checking all possible fault conditions. In the case no fault is detected, the system continues to check for the charge state. Since batteries are a lot more usefull when charged, our systems priority lies with charging the battery whenever possible. To do so the charger needs a valid input voltage which is sensed with the signals V IN HIGH and V IN LOW. As a last check before going into the charger state, the 'full' flag is checked. After the battery is completely charged, this flag is set for a specific amount of time to prevent the system from falling in a loop where the charger is started over and over, even if the battery is practically fully charged.

If the system has not detected any errors and is not able to charge, it will now check if the battery is still connected. As we have discussed in the hardware section there is a capacitor keeping the MCU alive for a while after the battery is disconnected, to be able to execute a shutdown routine before actually powering off. This check triggers the shutdown procedure whenever needed. If the battery is still there, as a last check the system will compute the voltage differences between the four battery cells, and when it excesses a predetermined value, the system will go into the balancing state. If even that is not the case, the system will continue to run their active routine, where the system is normally operating and discharging the battery.

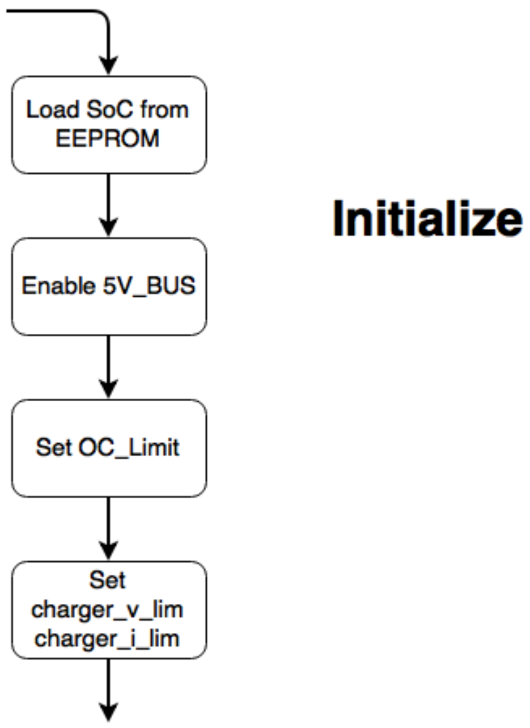


Figure 8.2: Flowchart of the initialization sequence.

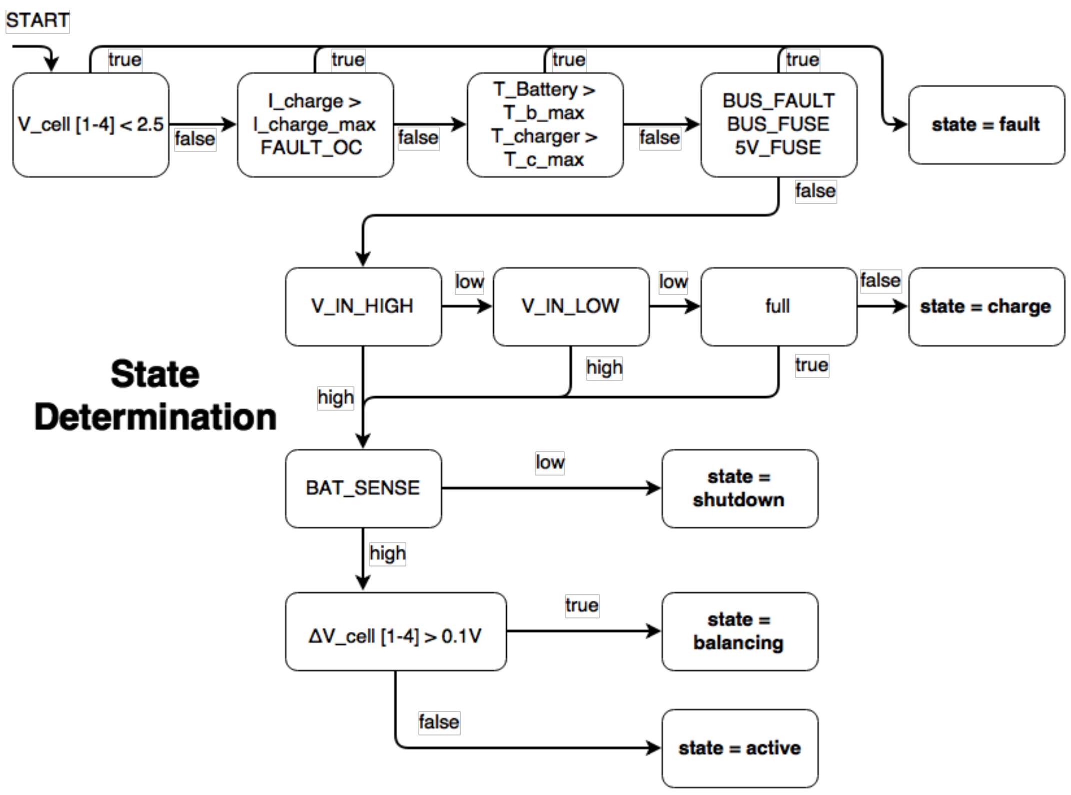


Figure 8.3: The system state is determined following this flowchart.

8.3.4. Charge routine

The first routine discussed in this chapter is the charging routine. As can be seen from 8.4, inside the charging routine two sub-states where created, one for pre-charging and one for normal CC-CV charging. In case the battery is drained too far, the battery can be damaged when directly starting with the CC-CV charging. To prevent this from happening, the system first checks if the cell voltages are below this threshold, to start the pre-charging if necessary. When the cells have been charged back to a higher voltage, the system will automatically start the CC-CV charging. Note that if the cell voltage is even lower (below 2.5V), the system will go into fault mode instead, as can be seen in the state determination flowchart. When the charging current becomes sufficiently low, the 'full' flag will be set, which results in the system quitting the charging routine.

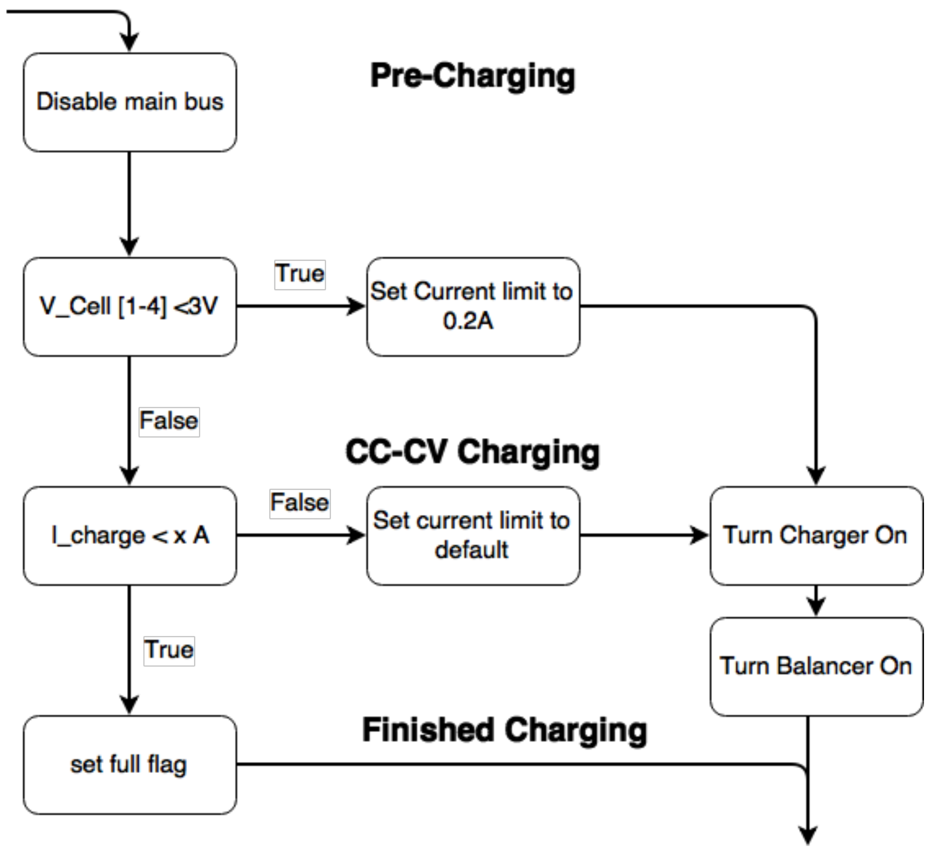


Figure 8.4: The charging routine.

8.3.5. Active routine

The active routine is active during normal system operation. As can be seen from figure 8.5, the first two steps consist of turning of the charger and balancer, in case the previous case was the balancing or charging state and afterwards the main system power bus will be enabled.

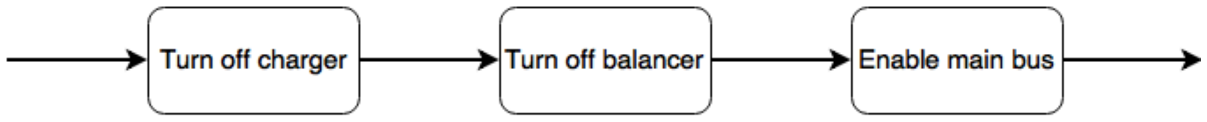


Figure 8.5: The active routine.

8.3.6. Balancing routine

If the system is working correctly, this routine should not be followed often, or even at all. The cells should be balanced enough whilst charging. One can see this routine as a back-up in case it goes wrong anyway. As can be seen from figure 8.6, the first two steps turn off the charger and main bus, in case they where active in the previous state. Afterwards the balancer is turned on and the system will cycle through the routine untill the voltage difference between the cells is acceptable again.

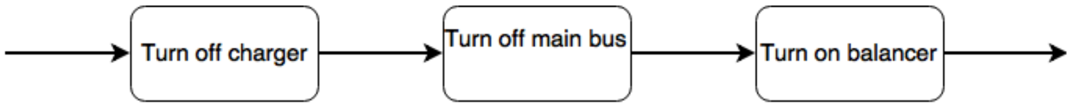


Figure 8.6: Flowchart of the balancing routine.

8.3.7. Fault routine

As can be seen in figure 8.3, there are several cases for which the fault routine can become active. All these cases require different measures. The measures following from each case are displayed in a flowchart in 8.7. This section will briefly walk trough them.

The first case is when any individual cell voltage is below its minimum safe value. In the case this occurs, we need to immediately stop discharging the battery. Both of the buses are switched off. In the case of a battery discharged this much, also charging again is risky, so the battery should be replaced or recovered outside of the system. In case of this condition also the BMS, including the MCU itself, should shut down. But when the system works correctly, this should not be able to occur.

The second case is a bus fault. As was stated in section 7.2.1, to achieve maximum interrupting speed, the system is equipped with a hardware over-current (short-circuit) protection, which after detecting and switching off the main bus, tries to turn it on again after 200ms. To make sure the system does not keep trying to turn itself on again forever, a counter is implemented in the fault routine, to make sure after a pre-set amount of attempts the main bus is disconnected permanently. A fault flag is set which results in an interrupt on the ZebroBus.

Third, we have the main bus fuse and the 5V bus fuse. Since these are self-resetting fuses, we will still turn off the corresponding bus in case they are triggered. These fuses will only be triggered if a big fault occurs in the main system out of control. Just like the bus fault discussed above, a fault flag will be set that will cause an interrupt on the ZebroBus.

In case the battery temperature becomes too high, the system turns off the systems possibly charging or discharging the battery, since one of them needs to be the cause of this fault. When the battery temperature has returned to the allowed range, the system will try starting up again in one of the other system states. Also in this case a fault flag will be set to inform the ZebroBus.

The last two cases consist of a charging current exceeding its maximum or the charger exceeding its maximum temperature. In case this occurs, the charger will be turned off. When the temperature has dropped below its maximum value, the system will continue operating again. In both cases a fault flag is set to notify the ZebroBus.

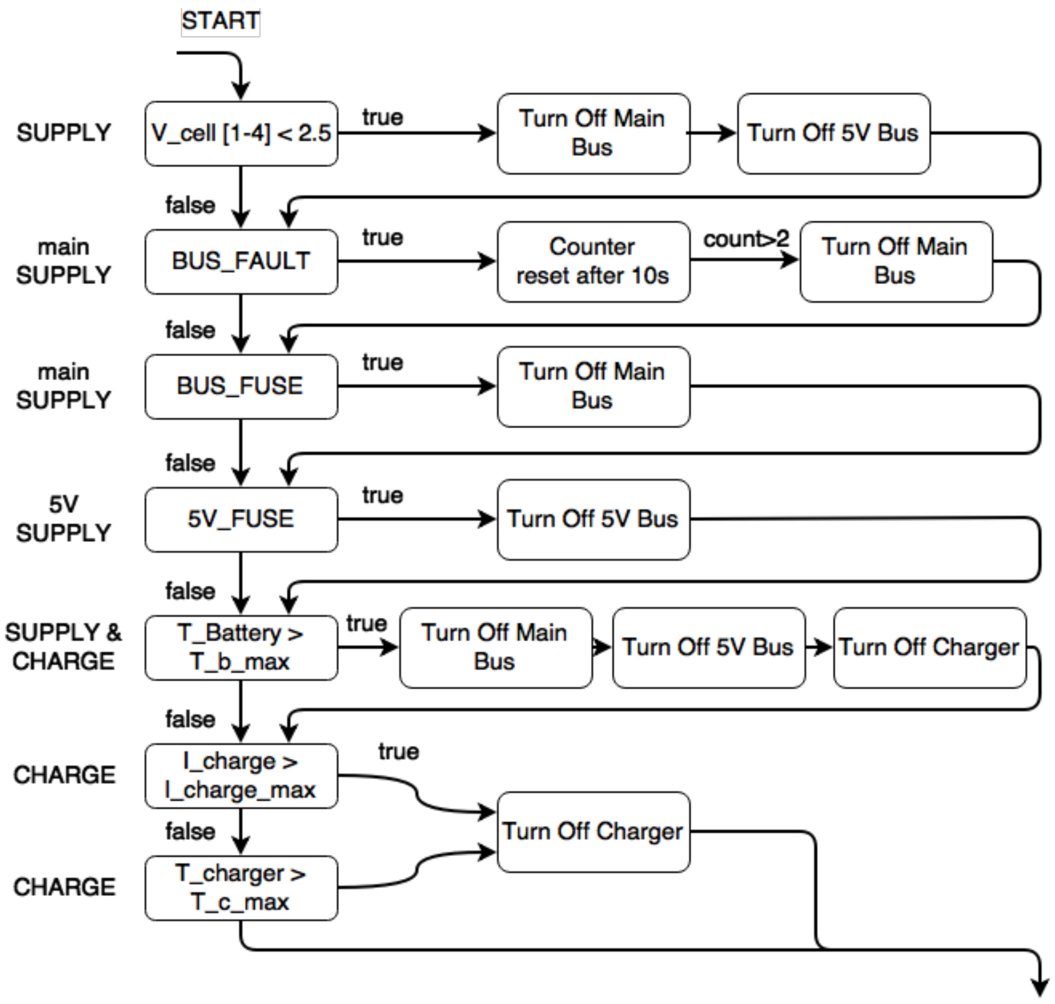


Figure 8.7: Flowchart of the fault routine.

8.3.8. Shutdown

The shutdown state needs to be executed only if the battery is switched off the system. This state will just write the SoC estimation (including the updates on SoH estimation) at that moment to the EEPROM and afterwards wait until the chips power connection is lost.

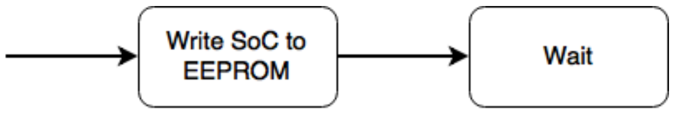


Figure 8.8: Flowchart of the shutdown state.

8.3.9. State-of-Charge estimation

Discharging the battery too much can cause damage to the system. Besides that, it is of great concern for our system to know the amount of charge left in the battery. For example, in determining when a Zebro robot should return to its charging station. To be continuously aware of the charge left, we will try to make an estimation of the State of Charge (SoC) of the battery.

There are different methods to do such a SoC estimation. The most applied manner is ampere-hour integration [12]. By integrating the current flowing in and out of the battery over time a good estimation can be made of the total amount of Ampere-hours stored in the battery. The main advantage of this method is the low computational power needed. The main disadvantage of this method is the cumulative integration error. This error can be resolved by 'resetting' the SoC after measuring the batteries output voltage, but this only works if the battery has been in steady state for a while.

Another way to make an estimation is by the use of models. Models with different level of detail are available to represent the SoC behaviour of lithium batteries, based on the battery voltage and current. To get an accurate estimation though, a very detailed and extend model has to be used, which requires a lot of computational power and is quite complex. [13]

For our specific application, we have chosen for ampere-hour integration. The SoC estimation does not have to be very accurate. Accuracy (and the cumulative error) become most crucial in systems where there is a continuous process of charging and discharging, which is not the case for our purpose. In our case, the value of SoC will be set back to 100 percent after fully charging the battery, eliminating the cumulative error built up so far.

As can be seen in figure 8.1, there is a timed interrupt starting the Ah integration routine. After a pre-set time interval the value of the current drawn from the battery will be sampled and used to approximate the integral over time. We choose to use an interrupt to make sure the interval in between calculations remains the same, which is needed to perform an accurate integration.

The accuracy of the resulting SoC depends on a few factors. First, of course, the accuracy of the current sensor. Second, the sampling frequency is an important factor. The higher the sampling frequency, the more accurate our estimation becomes. But, since we use an interrupt to perform the integration, the time interval has to be large enough to execute a cycle of the main program in between two Ah integration routines. This will be a trade-off we have to make when the source code is finished. As a final factor, the algorithm used to approximate the Ah integral plays a role.

For numerical integration a few different approaches can be used, according to David and Rabinowitz [14]. The most simple one simply takes your sample value and stretches it out over half the interval between the preceding and following sample, as can be seen in figure 8.9. This method is called the rectangle rule.

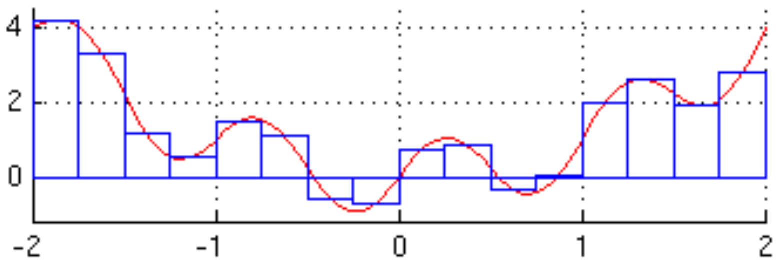


Figure 8.9: Approximating an integral following the rectangle rule.

A slightly more complex algorithm follows the trapezium rule. The main difference with the rectangle rule is they approach the signal between two samples by a straight line passing through these two points. An example can be seen in figure 8.10. In the SoC an algorithm based on this trapezium rule is chosen to be used, since it is still simple and only requires minimum more computational power over the square rule algorithm. One could even further improve the approximation by estimating a higher order function between the samples based on a larger set of samples instead of only the preceding and following ones, but that becomes complicated really fast and is out of the scope of this project.

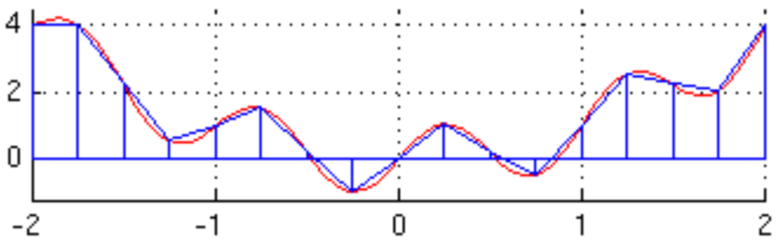


Figure 8.10: Approximating an integral following the trapezium rule.

When our measurement eventually turns out not to be accurate enough, there are multiple possible solutions to increase this accuracy. First, we can improve the software algorithm. According to [12], the optimal SoC estimation is reached by combining Ah integration with an algorithm, correcting for the measurement errors. Also the algorithm approximating the integral can be chosen more accurate, but both of these choises will make the software more complex and require more computation power and time. Finally, one could also add another microprocessor (possibly operating at a higher clock speed), which only performs the SoC integration algorithms.

8.3.10. State-Of-Health estimation

Lithium batteries, like the one used in our system, lose capacity over time. The datasheet of our specific battery specifies that the battery has to be discarded if the maximum charge has reduced to 80 percent of the initial value. The reduction of this capacity also brings risks for our SoC estimation. As discussed before, the system determines the SoC assuming the SoC is at its maximum after charging completely. If this maximum turns out to be only 80 percent of the batteries original capacity, a gigantic measurement error follows which could result in significant damage to the system.

To prevent this from happening we decided to also implement a state of health (SoH) estimation as a sub-system of our SoC estimation. When charging the battery, the system will still perform the ampere-hour integration like when estimating SoC, but now on the charge current. An estimation of the total amount of Ampere-hours charged into the battery, together with the amount of energy our SoC believed was still left in the system, results in a rough estimate of the capacity of the battery at that specific moment. After a few of these measurements, the system can average out these rough estimates and approach the amount of capacity the battery still has. This knowledge can in its turn be used to adapt the SoC estimation algorithm and when necessary discard our battery.

8.3.11. Central Communication Bus

As can be seen from figure 8.1, after finishing one of the systems routines, the system always first updates the communication bus (ZebroBus) to the main system. This main communication bus is a I2C bus, which only has to be supplied with information when requested. To make sure the relevant information is available when a request is received, the routine stores the values into a memory, which can be read out whenever needed. When a fault flag is active, the system will send out an interrupt to the main system on the communication bus, to be able to directly report the fault and which fault.

9
Evaluation

During the BAP, the functionality and schematics of all modules of the BMS were designed. The physical PCB design is not yet finished. At the moment of writing this thesis, no modules have been tested yet, as explained in the respective chapters. However, the balancer design has been verified using simulation and the charger design was discussed with an engineer experienced in DC/DC converter design. Therefore we are confident that in the next week the BMS design can be made into a working prototype.

9.1. Requirements

The module requirements are evaluated in the subsections below. Besides their own requirements, all modules also must meet requirement 1: **Fit into the Zebro casing**. Although the PCB design is not finished, because the modules have been designed with space restricion in mind, it is very likely this requirement will be met.

9.1.1. Charger

Requirements 1 and 2 have been implemented in the charger module. The charger can charge a battery with 6A from an input voltage range of at least 8V - 32V. The charging module is also designed to be safe. It will only charge when controlled by the MCU and no current can flow between the input and the battery when the charger is turned off. This means the charger meets requirement 2: **Safely charge the battery with power from the wireless charging module** and requirement 7: **Charge the battery with power from a wide range of sources (e.g. solar panels)**

9.1.2. Balancer

Requirement 3 has been implemented in the balancer. If the balancer is used at the right time (during charging and whenever the cells are imbalanced), requirement 6 is met: **Optimize battery capacity and lifetime**.

9.1.3. Power management unit

Requirements 4 and 5 have been implemented in the PMU. The PMU supplies 5V, 3A to the Zebro controllers and 14.4V, 8 A to the main system. This means requirement 3 is met: **Safely discharge the battery to deliver power to the Zebro system**.

9.1.4. MCU

The MCU has a lot of tasks, associated with these requirements: 6, 7, 8, 9 and 10. The MCU measures the charge and discharge current, the cell voltages and the battery temperature. It also measures the charger temperature as a measure for the module temperature, since the most power is expected to be dissipated in the charger. With these values the MCU detects faults and recovers from some of them, or reports the fault to the main controller and waits for commands. All measurement values are available on the Zebrobus, together with the SoC, SoH and the capacity, so the main controller has the information to decide when to return to the charging station. The MCU also has a debug port. This means the MCU meets requirement 4: **Estimate battery State of Charge**, requirement 5: **Minimize battery charging time**, requirement 7: **Charge the battery with power from a wide range of sources (e.g. solar panels)**, requirement 8: **Estimate battery State of Health** and requirement 9: **Communicate with other Zebro modules**.

9.2. Future work

The battery module was designed in a short period of time. A few things were not researched thoroughly because it was deemed unnecessary and out of scope for this assignment. First a different battery charging algorithm could be used to further reduce charging times and optimize battery lifetime. This could either be done in software with the current charging module, or by designing a whole new charging module if a exotic charging algorithm like sinusoidal charging is to be used. Secondly the SoC and SoH estimation could be made more accurate by using higher order models. The current implementation only does a first order approximation.

A

MCU Signal List

Pin	Type	Name	Description
PA0	IN ADC	V-cell-1	Battery cell 1 voltage
PA1	IN ADC	V-cell-2	Battery cell 2 voltage
PA2	IN ADC	V-cell-3	Battery cell 3 voltage
PA3	IN ADC	V-cell-4	Battery cell 4 voltage
PA4	IN ADC	I-out	Main bus current sense
PA5	IN ADC	I-charge	Charger current sense
PA6	IN ADC	T-battery	Battery NTC temperature sense
PA7	IN ADC	T-charger	Charger NTC temperature sense
PB0	OUT D	Bat-sense	Battery (voltage) detection
PB1	OUT D	Charger-en	Enable signal for charger
PB2	OUT D	Bus-en	Enable signal for main bus
PB3	OUT D	5V-en	Enable signal for 5V bus
PB4	OUT D	Fault-oc	OC protect LED
PB5	prg	mosi	Programming
PB6	prg	miso	Programming
PB7	prg	sck	Programming
PC0	IN D	Zebrobus-scl	I2C for Communication Bus
PC1	OMNI	Zebrobus-sda	I2C for Communication Bus
PC2	OUT D	Zebrobus-irq	I2C for Communication Bus
PC3	IN D	V-in-low	Charger input voltage detection (low)
PC4	IN D	Charger-cc	Chager cc mode flag
PC5	IN D	Bus-fault	Hardware overcurrent protection oc detect flag
PC6	IN D	Bus-fuse	Main bus fuse 'blown' detect
PC7	IN D	5V-fuse	5V bus fuse 'blown' detect
PD0	IN D	Debug RX	Debug receive
PD1	OUT D	Debug TX	Debug transmit
PD2	OUT D	DAC-lat	SPI for external DACs
PD3	TXD1	DAC-0-sdi	SPI for external DACs
PD4	XCK1	DAC-sck	SPI for external DACs
PD5	IN D	V-in-high	Charger input voltage detection (high)
PD6	OUT D	PWM-1	PWM for cell bancer
PD7	OUT D	PWM-0	PWM for cell bancer

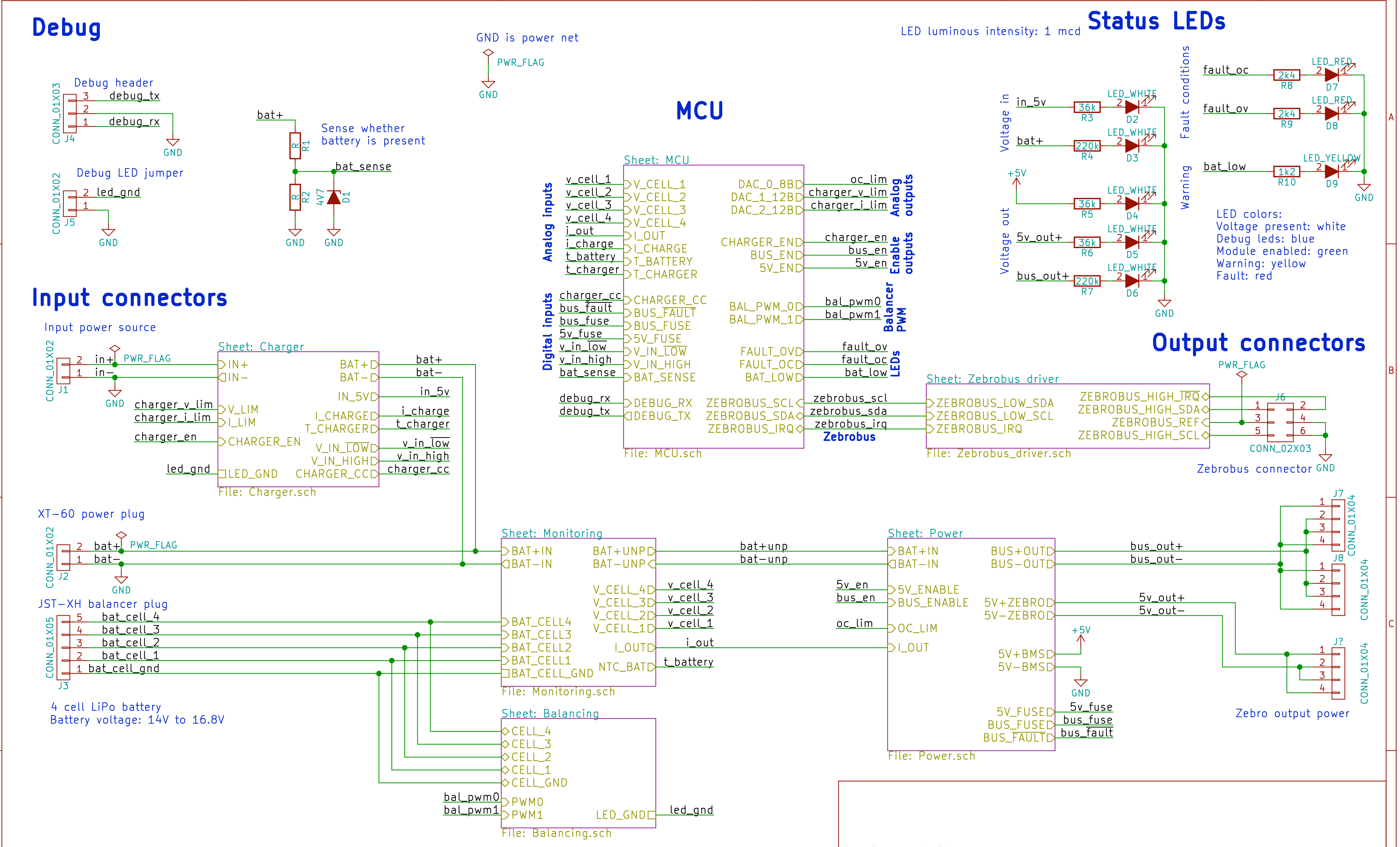
ADC type refers to the internal ADCs of the microprocessor. The letter D refers to a digital signal. prg are pins only used for programming. OMNI is a bidirectional port.

B
BMS schematic

The following pages contain the schematics for the Zebro BMS. It consists of a main sheet and the following subsheets:

- Monitoring
- MCU
- Balancer
- Power management
- Charger
- Zebrobus driver

The schematics have been made with KiCad.



Bob Dullaart & Pieter van der Kamp

Sheet: /
File: Zebro_BMS.sch

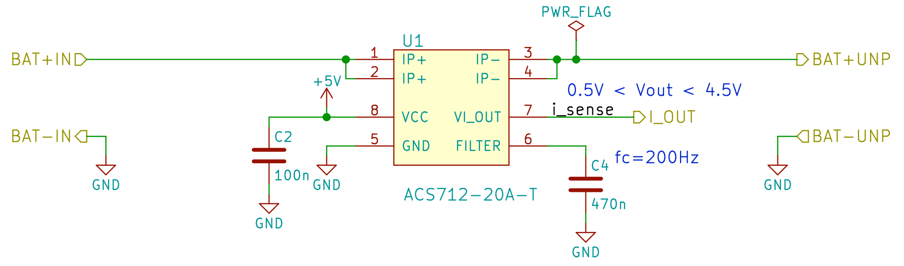
Title: Zebro BMS

Size: A4	Date: 2017-06-19	**Rev: 0.1**
KiCad E.D.A.	kicad 4.0.6	Id: 1/7

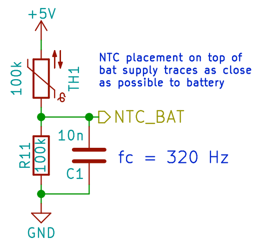
Monitoring

Current sensor

−20A < I < 20A



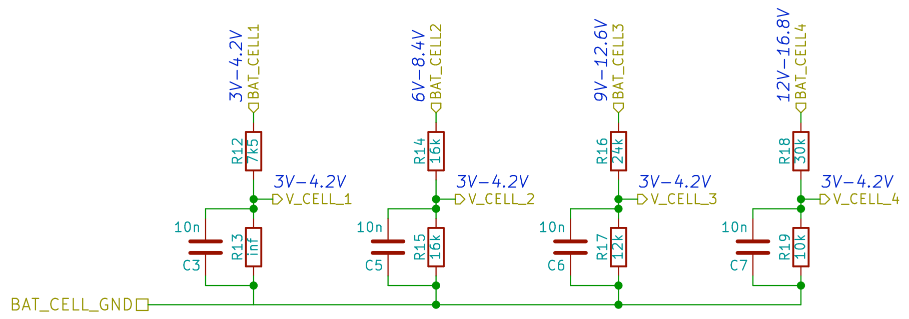
Battery temperature



+5V
100k
TH1
NTC placement on top of bat supply traces as close as possible to battery
NTC_BAT
10n
R11
100k
C1
fc = 320 Hz
GND

Voltage division for MCU

fc = 2 kHz



Bob Dullaart & Pieter van der Kamp

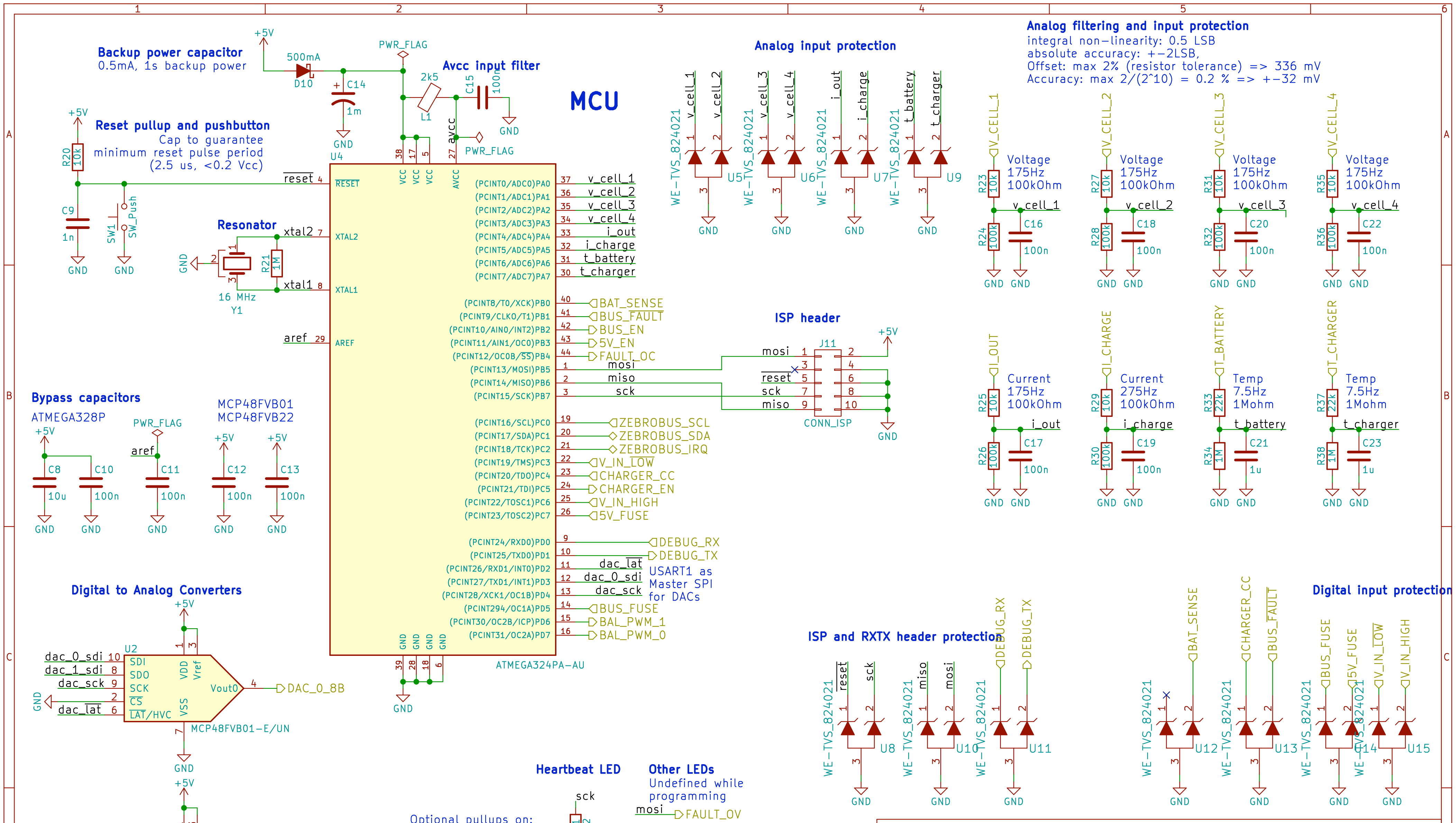
Sheet: /Monitoring/

File: Monitoring.sch

Title: Zebro BMS Monitoring

Size: A4 Date: 2001−06−19 **Rev: 0.1**

KiCad E.D.A. kicad 4.0.6 Id: 2/7



Bob Dullaart & Pieter van der Kamp

Sheet: /MCU/

File: MCU.sch

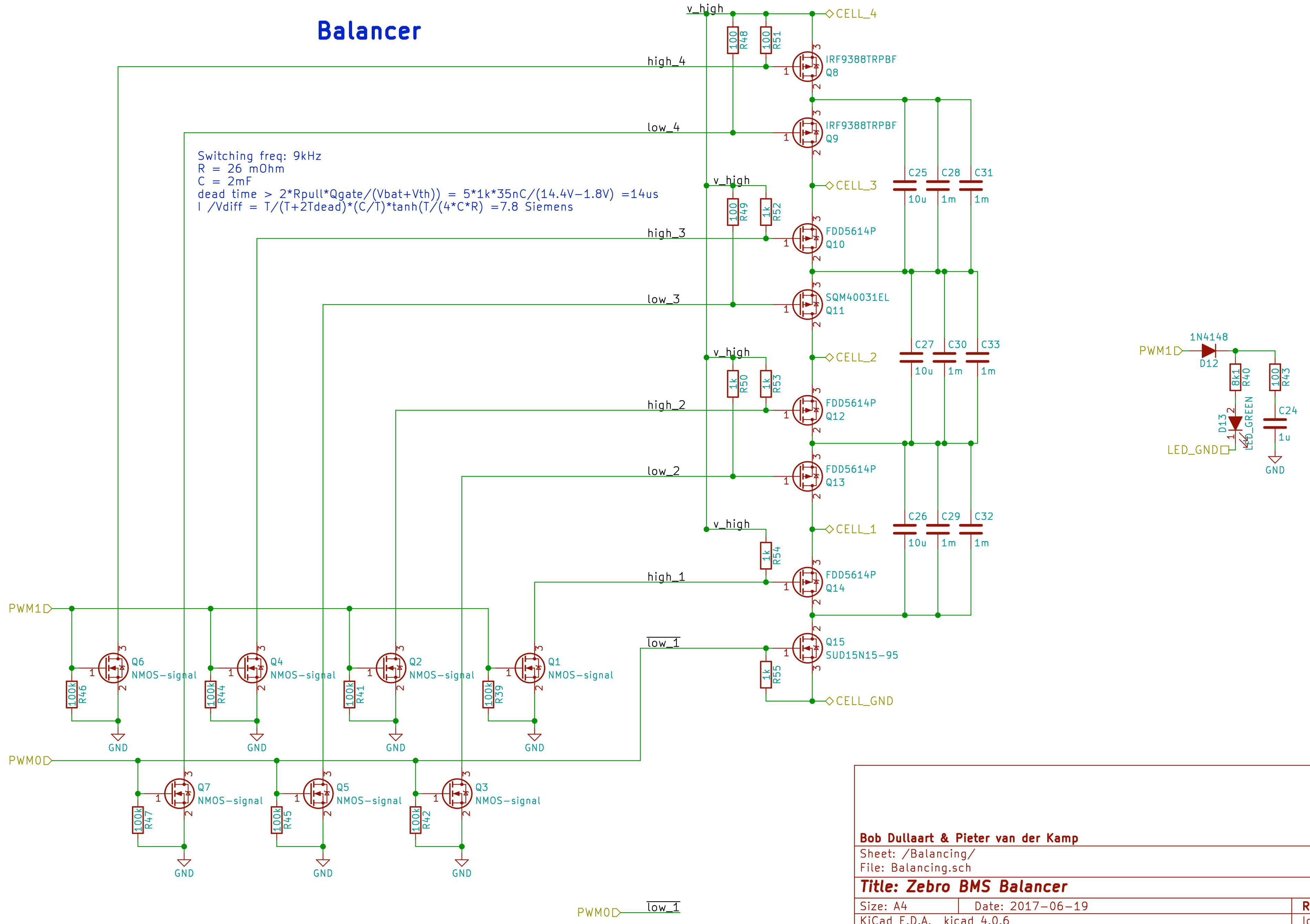
Title: Zebro BMS MCU

Size: A4 Date: 2017-06-19 Rev: 0.1

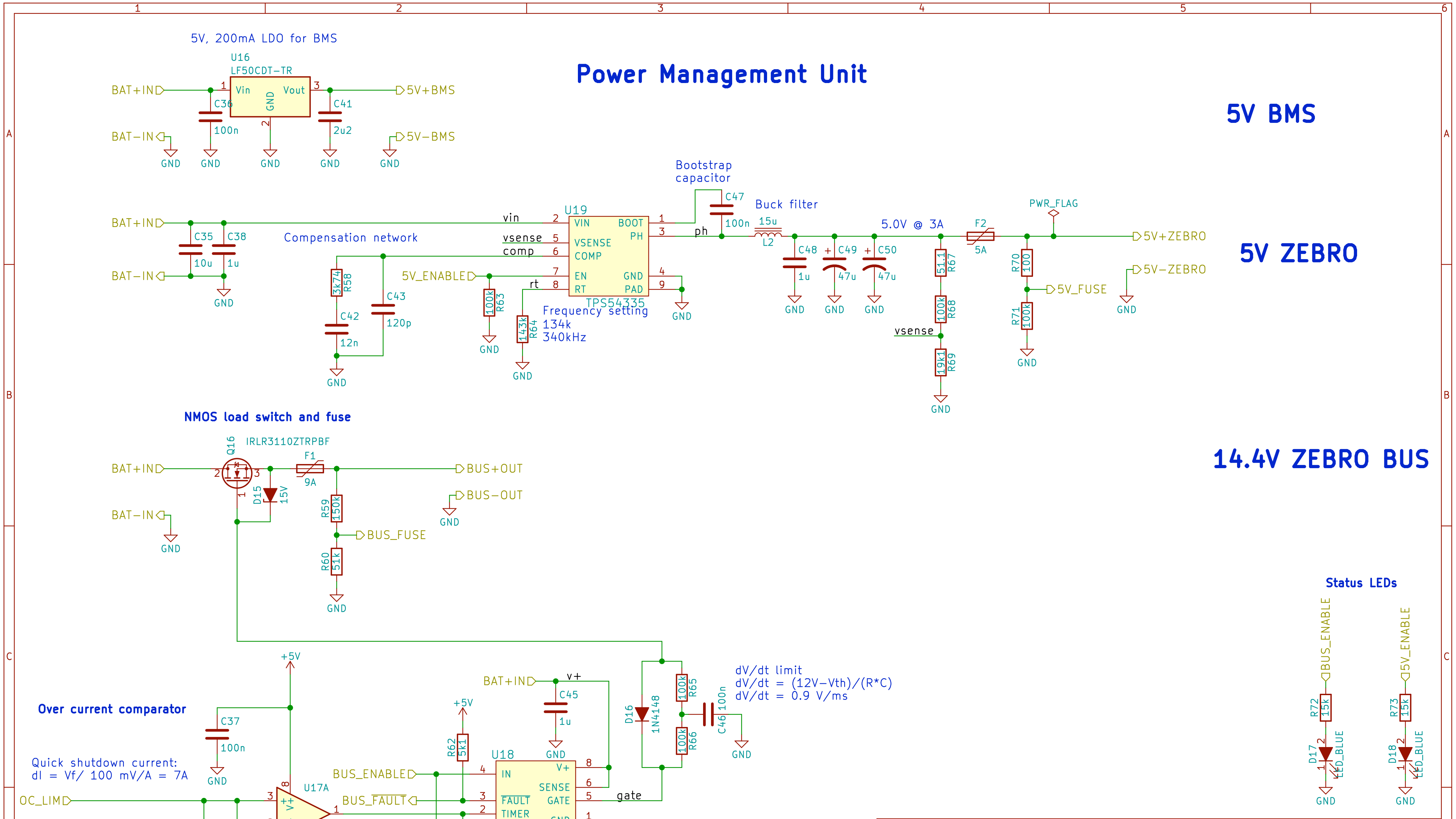
KiCad E.D.A. kicad 4.0.6 Id: 3/7

Balancer

Switching freq: 9kHz
R = 26 mOhm
C = 2mF
dead time > 2*Rpull*Qgate/(Vbat+Vth)) = 5*1k*35nC/(14.4V−1.8V) =14us
I /Vdiff = T/(T+2Tdead)*(C/T)*tanh(T/(4*C*R) =7.8 Siemens



Bob Dullaart & Pieter van der Kamp
Sheet: /Balancing/
File: Balancing.sch
Title: Zebro BMS Balancer
Size: A4 Date: 2017-06-19 **Rev: 0.1**
KiCad E.D.A. kicad 4.0.6 Id: 4/7



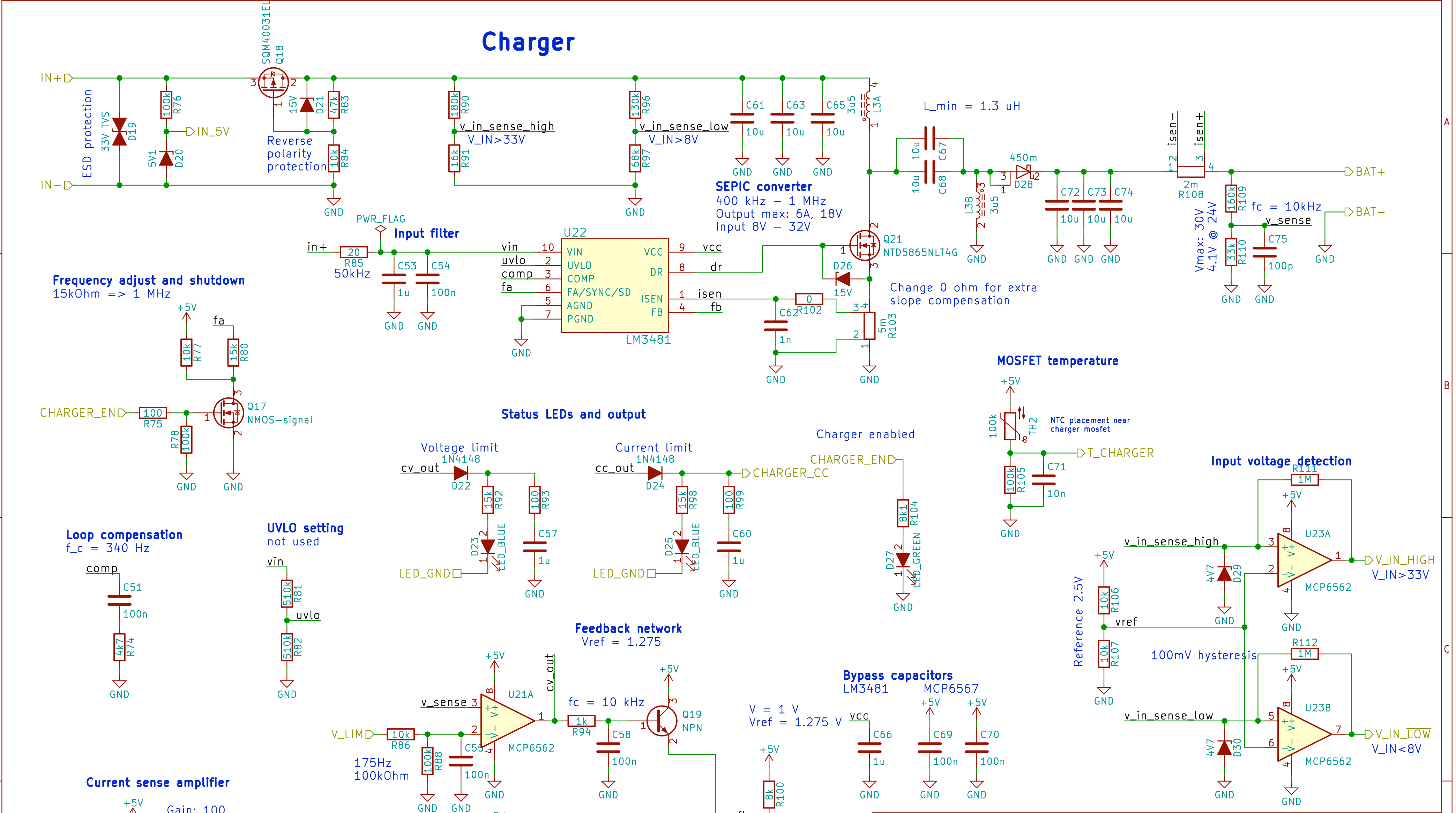
Bob Dullaart & Pieter van der Kamp

Sheet: /Power/
File: Power.sch

Title: Zebro BMS PMU

Size: A4	Date: 2017-06-19	Rev: 0.1
KiCad E.D.A. kicad 4.0.6		Id: 5/7

Charger



Bob Dullaart & Pieter van der Kamp

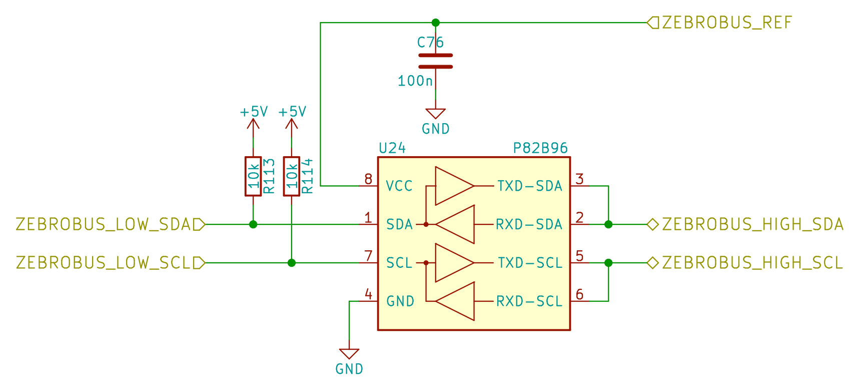
Sheet: /Charger/

File: Charger.sch

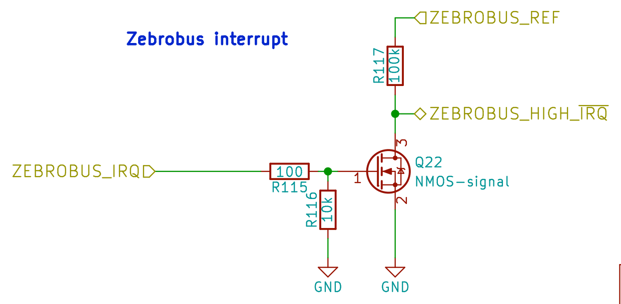
Title: Zebro BMS Charger

Size: A4	Date: 2017−05−03	Rev: 0.1
KiCad E.D.A. kicad 4.0.6		Id: 6/7

Zebrobus driver



Zebrobus interrupt



Bob Dullaart & Pieter van der Kamp

Sheet: /Zebrobus driver/

File: Zebrobus_driver.sch

Title: Zebro BMS Zebrobus driver

Size: A4	Date: 2017-06-19	**Rev: 0.1**
KiCad E.D.A. kicad 4.0.6		Id: 7/7

Bibliography

[1] *The Zebro Project,* https://www.tudelft.nl/d-dream/teams/zebro-project/ (2017), [Online; accessed 12 June 2017].

[2] W. Shen, T. T. Vo, and A. Kapoor, *Charging algorithms of lithium-ion batteries: An overview,* in *2012 7th IEEE Conference on Industrial Electronics and Applications (ICIEA)* (2012) pp. 1567–1572.

[3] L. R. Chen, S. L. Wu, D. T. Shieh, and T. R. Chen, *Sinusoidal-ripple-current charging strategy and optimal charging frequency study for li-ion batteries,* IEEE Transactions on Industrial Electronics **60**, 88 (2013).

[4] T. Instruments, *Lm3481 / -q1 high-efficiency controller for boost, sepic and flyback dc-dc converters,* http://www.ti.com/lit/ds/symlink/lm3481-q1.pdf (2007).

[5] V. Vorperian, *Small signal modeling of the tightly-coupled SEPIC converter*, Tech. Rep. (Ridley Engingeering).

[6] D. Zhang, *Designing A SEPIC Converter*, Tech. Rep. (Texas Instruments, 2006).

[7] V. Vorperian, *Simplified analysis of pwm converters using model of pwm switch. continuous conduction mode,* IEEE Transactions on Aerospace and Electronic Systems **26**, 490 (1990).

[8] T. Instruments, *Compensation for the LM3478 Boost Controller*, Tech. Rep. (Texas Instruments, 2003).

[9] J. Cao, N. Schofield, and A. Emadi, *Battery balancing methods: A comprehensive review,* in *2008 IEEE Vehicle Power and Propulsion Conference* (2008) pp. 1–6.

[10] T. Instruments, *Tps5433xa 4.5-v to 28-v input, 3-a output, synchronous step-down dc-dc converter,* http://www.ti.com/lit/ds/symlink/tps54335-1a.pdf (2014).

[11] *Datasheet ATMEGA328PA,* http://www.atmel.com/images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Complete.pdf (2017), [Online; accessed June 2017].

[12] A. Berrueta, I. S. Martín, P. Sanchis, and A. Ursúa, *Comparison of state-of-charge estimation methods for stationary lithium-ion batteries,* in *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society* (2016) pp. 2010–2015.

[13] M. Partovibakhsh and G. Liu, *An adaptive unscented kalman filtering approach for online estimation of model parameters and state-of-charge of lithium-ion batteries for autonomous mobile robots,* IEEE Transactions on Control Systems Technology **23**, 357 (2015).

[14] P. J. Davis and P. Rabinowitz, in *Methods of Numerical Integration.*