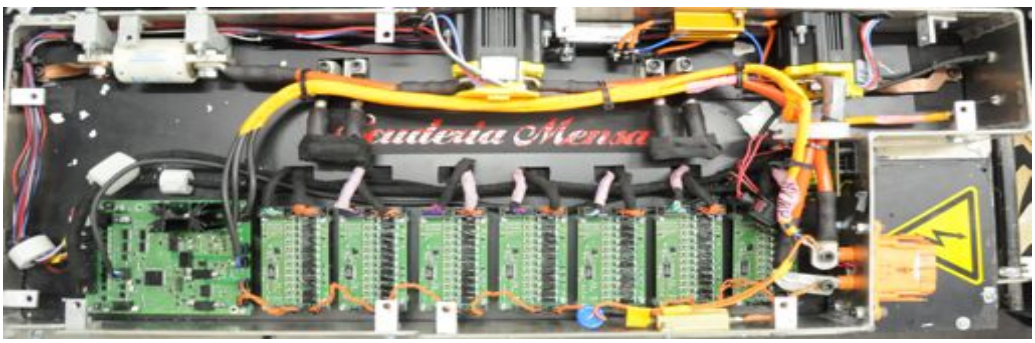
Development of an Battery Management System

for a Formula Student Electric Tractive System Battery



Arbeit von David Brandt (150607)
Betreuer: Mathias Harter

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe, und dass ich alle Stellen, die dem Wortlaut oder dem Sinn nach anderen Werken, auch elektronischen Medien, entnommen sind, durch Angabe der Quellen als Entlehnung kenntlich gemacht habe.

_____	_____
Ort, Datum	Unterschrift

Table of Contents

Development of an Battery Management System

for a Formula Student Electric Tractive System Battery

David Brandt
Hochschule Rheinmain

1 Introduction

This document describes the Battery Manangement System and the decisions taken while developing the system. This student research project has taken place in the Formula Student Team of the Hochschule Rheinmain, which is named Scuderia Mensa. The car for the 2012 season is the second electric car build by the team, this year we try to do a lot of electrical engineering work in contrast to last year, where almost all electric parts where bought from companies. So I started the process of developing a Battery Management System, which was more work than anyone expected, especially becuase I ended up doing everything myself, barely having any help.

Since this is the first BMS developed by the team and our UaS has no staff that ever build something like this, it was though to get good help and information specific to BMS. After a lot of trouble we found good help at Opel/GM. Though even the big OEM and Tier1 component suppliers are struggling to get the knowhow to build good BMS.

The BMS is intended to be open sourced for free use, so this document primarily describes the system, the design decisions and ways to avoid errors. This document should help implementing the BMS to a new car.

A lot of issues will not be part of this document, but it always is important to keep in mind that ESD sensitvity of devices should be taken seriously, that EMC can kill months of development and system integration takes atleast half the time it took to develop the parts of the system.

2 Why a BMS is needed

To keep this document simple I will not go deeper into cell chemistry or physics involved in the charging or discharging process, this alone fills whole books.

I just assume everyone who reads this either knows how exactly these cells work or just accepts whatever the cell manufacturer wrote to his datasheets.

Basically there are two things that kills Lithium batteries, one is damaging the isolation layer and the other are gases that can build up and can even make the cell explode. A damaged isolation layer usually brings gas forming with it.

What you really need to obey are the limits set by the manufacturer for the specific cells, for our Lithium-Polymere cells these are fairly simple:

- maximum 4,2V +/- 0,03V, minimum 2,7V, both at no load

- maximum discharge 300A for 10s, 190A continuous

- maximum charge 72A

- working temperature range -20 to 55°C

- storing at room temperature for one year at 50% SoC (about 3,7V)

2.1 Monitoring

One mode where the cells need to be monitored is the discharging, usually in the car. Here the BMS is operating in a monitoring mode only, it just can react to a fatal error such as a severe undervoltage fault by opening the AIRs.

The BMS sends all data and status information to the CAN Bus, for example all cell voltages, maximum, minimum and average cell voltage, temperatures and so on.

In our case the BMS did open the AIRs on fatal error, such as over and under voltage and temperature. It does the precharge continuously checking for errors while the precharge is running.

Out of this, the BMS is passive only sending all data gathered to the bus, the dSpace should then regulate the output power to keep the minimum cell voltage at a minimum value. This should prevent that the BMS has to open the AIRs with current flowing.

2.2 Charging

While charging the BMS should control a charging power source, in our case this was a large 19” power supply. The current flowing into the cells has to be lowered or shut/on and off to get the cells up to their maximum voltage. As a Lithium cells is pushed up when being charged and then gets down to a lower level with less charge current, the charge current has to be lowered slowly to fully charge the cells. This is called a CC/CV charging, first there is a constant current, then a constant voltage.

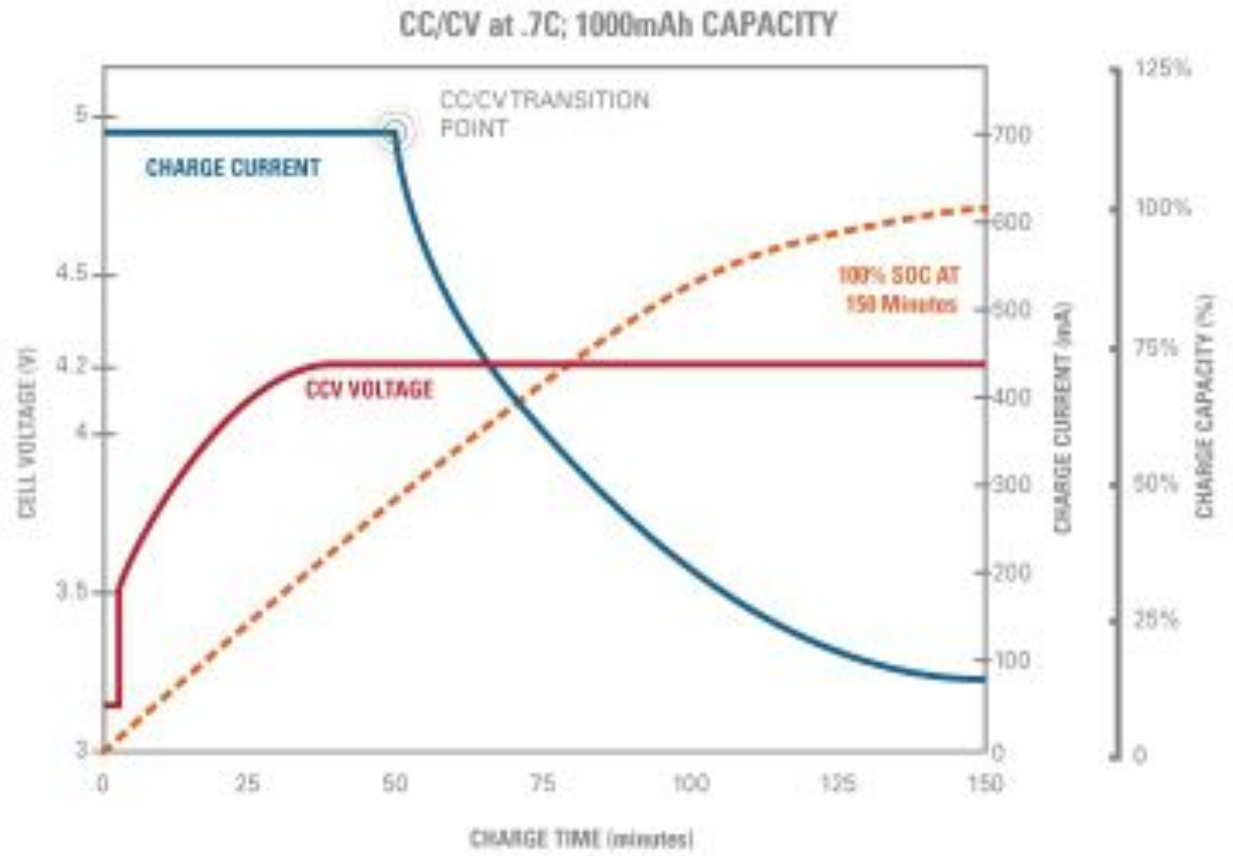


Figure 1: Typical Constant Current/Constant Voltage Diagram

The only other other critical function of the BMS while charging, which not necessarily happens while charging, is balancing the cells. This is needed, because when the first cell hits it's maximum voltage, there may be cells that can be charged some more that this cell. So all cells have to have an equal voltage when charged, this can happen in multiple ways. Due to deadlines and general problems the balancing algorithm was implemented to be started manually and not while charging.

3 Functional Requirements

There are three sources of requirements in this case. The larger part of requirements come from the basic needs of the cells, which where mentioned earlier and are summarised here:

- Monitoring of each cell voltage
- Monitoring of enough temperatues to give a health status
- Controling of a charger
- Balancing all cells
- Control possibility to shut down the tractive system in case of an error

Furthermore the FSE Rules define additional requirements:

- All cell voltages have to be displayed at the event site (not directly mentioned in the rules)
- There has to be an indicator which shows if the the output voltage of the battery is greater than 40V, even if the battery is not plug in to an external energy source
- Temperature monitoring of 30% cells (for LiPos at least)
- Has to have direct control to shutdown the tractive system (i.e. direct control of AIRs)
- Is may not be possible for the BMS to close AIRs if a emergency button is pressed
- A 3-5Hz output for a tractive system active light, which signals greater than 40V at the output, this has to be hardwired without the use of microcontrollers

And then there are our special requirements, which should apply to most formula student cars:

- Fully controllable via CAN Bus, all data should be logged to the bus
- Should be easily adaptable to other types of cell types and other cell counts
- Controlling a power source over RS232 (ET Systems - LAB HP)
- Circuit which monitors voltage drop across the precharge resistor
- Hardwired electronics to prevent closing of AIRs when there is current on the precharge and to prevent closing of the AIRs after an emergency button has been pressed
- Switching for all AIRs and the TSAL on the BMS
- Should work with input voltages in the range of 7.5V up to 15V
- Permanent logging without the need for an external logger (e.g. SD Card)
- Simple status display directly at the battery case without the use of additional tools like a notebook
- No proprietary materials, source or other sources which would prevent releasing the project as open source to other formula student teams
- An real life accuracy of at least +/-30mV including all factors

4 Implementation Choices

When it comes to a real BMS implementation, there are some general decisions. You have to choose the correct accuracy for your cells, in most cases this can easily be done by consulting the datasheet of the cells. A general topology has to be choosen, this can have a influence on cabling and EMC problems that can arise. The last decision is a decision about the communication to the rest of the car. I will detail some choices you have in the next three chapters.

4.1 Topology

By topology I really mean communication topology, this comes, becuase you certainly will end up with multiple modules measuring the voltage for a stack of cells. This comes from the fact that it is incredibly complicated to measure 100 voltages in the range of 0V to 600V, also this would make a maintanence plug impossible. It is desirable to have stacks of a maximum of 120VDC that can be seperated in a maintanence mode, 120VDC is choosen because this is said to be the highest voltage you could possibly touch without a potential risk of life. Not to mention how big a PCB with enough electronics to measure so much voltages at once would be. I will detail the three topologies i found to be useful.

4.1.1 Star Isolated Bus Topology

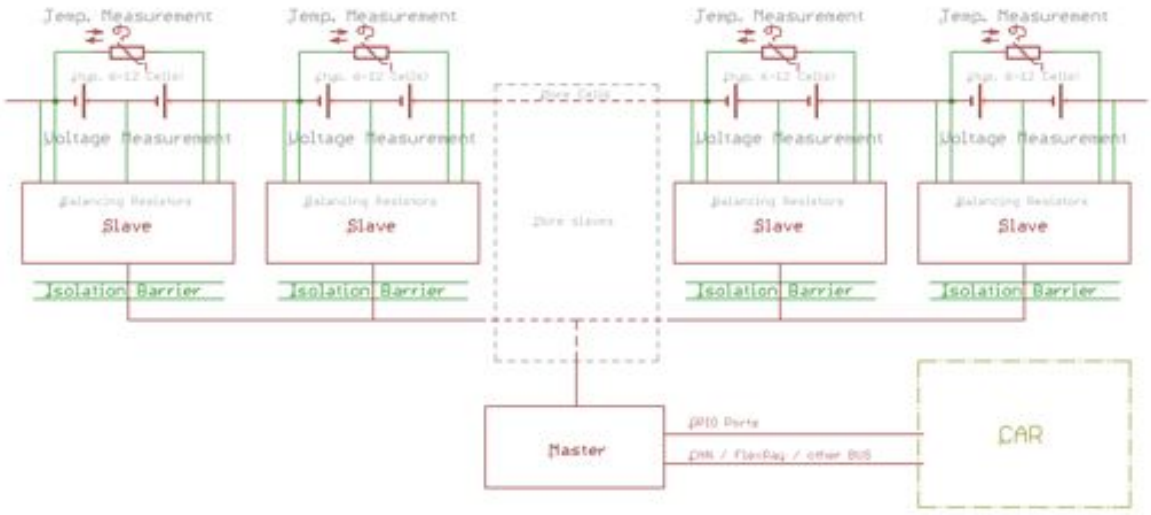


Figure 2: Standard Star Isolated Bus Topology as used in BMS

I really didnt find a better name to describe this communication scheme. It basically is a standard microcontroller bus, like a SPI Bus, with a star formed isolation concept. This means, that there is a bus cable connecting all slave modules and the master, but because there can be a huge voltage difference between the modules, there has to be some sort of isolation between all bus participants.

There are a couple of problems with these systems, a huge downside is the long wiring path, which has to span the whole battery system, this can bring huge EMC problems and it can be hard to guarantee the isolation of this cable and the rest of the battery. Also a lot of isolators can be more

costly than an daisy chained interface, which is described later.

One upside is the ability to know if one slave device is broken or better knowledge where a broken cable might be. Slave ICs designed for this topology usually monitor six to twelve cells.

4.1.2 Daisy Chain

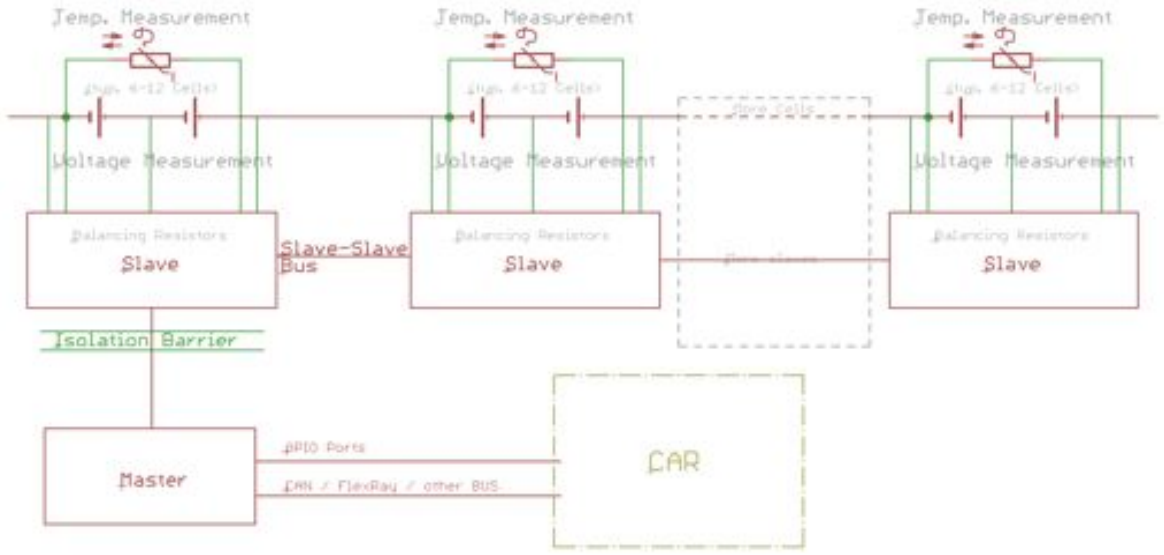


Figure 3: Typical Daisy Chain Topology, shown without Isolation between slaves

A Dais Chain Bus is a little bit different, each slave can only communicate with the next slave and/or previous slave. This has its downside on the protocol side, but gives the big benefit that there is a lot less cabling needed and all cables can be shorter. Some Daisy Chain devices even get all their power from the cells they're monitoring, this eliminates all low voltage wires that could potentially go along the high voltage wires.

The slave modules can be isolated against each other with an optocoupler or similar device or they can have a communication protocol which spans the voltage each stack monitors, usually around 60V. A non isolated interface usually uses some sort of current mode to transfer data, as this is more tolerant to electromagnetic fields.

Small Daisy Chained devices are relatively easy to build from standard devices. For example a slave module using a small 8 bit microcontroller with a 10 bit analog digital converter can monitor a single cell, balance the same cell and communicate with adjacent slaves to tell the voltage to the master module, this only works if the accuracy is good enough.

Ready made BMS ICs with Daisy Chained devices are always for more than one cell, usually six to twelve cells. There are BMS ICs for fewer cells, but these are usually for consumer electronis like mobile phones.

Especially larger Daisy Chained Slave Modules could possibly be the cheapest solution to build a BMS, but when using a specific IC and not a self made solution, they always use a proprietary bus system to communicate between the adjacent modules. This leads to certain problems when a company wants to have interchangeable parts, the master module too would be completely interchangeable if all slave variants that could be used shared the same protocol.

Because system integration and interchangeability is a big deal when it comes to millions of produced units OEMs tend to go a different approach, as described next.

4.1.3 Car Bus Integrated

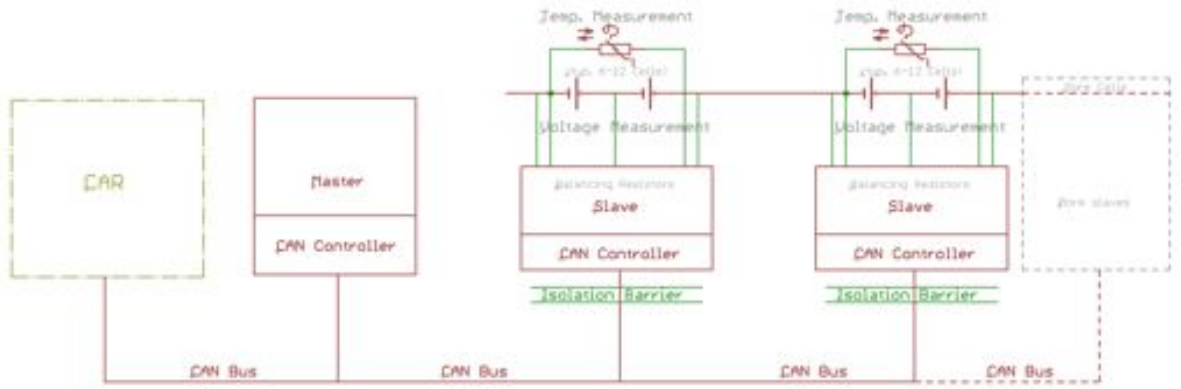


Figure 4: A BMS that communicates over the CAN Bus of the Car

This seems to be the weapon of choice for most OEM BMS, this design has some upside and downsides, yet you have all BMS communication on the standard bus of the car.

The downsides here are obvisous, first, the car communications bus (e.g. CAN) has to go through the whole battery and to every slave module, which can be a problem. Most times having a CAN Bus in the battery shouldn't be a problem, since these busses are insensitive to electro magnetic fields. A controller for the car bus is an additional part and so it is more expensive then most other solutions.

One big upside is that you have all messages available for logging, which can cut on software and hardware routing all information to the bus, as the data is needed for motor control and to present a charging state to the driver this can save work at the master. Also, due to the fact that in most cases a microcontroller is needed for the car bus, one can do conversions and basic logic inside the slave modules themselves, this gives some indepency on what BMS IC to choose. Every BMS IC gives data in a little different format, which normaly has to be calculated in the BMS Master, having all slaves communicate in a standard format gives the ability to interchange them without modifying the Master.

4.2 Accuracy

The next thing to decide is the total accuracy needed for the BMS. This is a rather easy decision for the cell voltage, most BMS ICs have 12 bit and higher resolution which results in a real life resolution of usually at least 10 bit, which then leads us to an accuracy of roughly 5mV. Most cell datasheets have boundaries of +-/30mV and alike, only when doing small slave devices with 10 bit analog to digital converters it can be tricky to stay within the cell boundaries.

The accuracy of the temperature sensors is even less of an issue normally, it is suffice to measure a range from -40°C to about 85°C with an accuracy of 1°C. This ends up in 7 bit resolution. Most cells have temperature ranges of -20°C to 55°C.

4.3 Car Communication

In most cases this is no design choice, as the bus system, usually CAN, already is set.

Where there is no bus system established in the car, there is a possibility to only use digital in and out to communicate with the battery. The most essential communication are “battery enable” and a “battery ok” lines. This totally excludes any power regulation when a cell is almost empty or when the temperature resis to fast by the motor controller, but it is sufficient to have a working car.

As the battery in a formula student car has just enough capacity to run an endurance, it gets very important to have a chance to reduce the power when the battery is almost empty. Otherwise the BMS would have to cut off the power at a point where it starts to see low voltages, as the cells go down on voltage when there is load. If you have a lot of load on the cells, the voltage drop can easily get to 500mV from idle voltage. As the BMS is unable to tell this situation to the motor controller, it just has to open the AIRs and throw an error, which renders the car unable to complete the competition.

With a good communication between the BMS and the motor controller it is possible to use 100% of the charge the battery can hold. The motor controller has to quickly reduce the power when the BMS sees low cell voltages. The car will get slower but this can give the car the extra lap it needs to finish an endurance run.

So the basic communication from the BMS to the car consists of cell voltages, cell temperatures and min/max/avg of these values. For cutoff only the minimum cell voltage and the maximum cell temperature is accounted, so the motor control will rely on these values. Other communication can be warning and error messages, also some sort of debugging messages for a prototype car. As all messages sent over a car bus system should be logged in a prototype car, more messages can help find issues that occur when driving.

The car, probably the motor controller or an human machine interface, tells the battery to start and stop the HV system, closing or opening the AIRs. For security and redundancy reasons it may be good the let the BMS know how much voltage the motor controller gets from the battery, to detect a broken wire. Other than that there is not much the BMS needs to know.

When charging, which in formula student cars is usually outside the car, there is some kind of control communication to the battery, telling it to start charging and to communicate with the power source.

4.4 Actors

The BMS should have direct control over a few parts of the HV system. Usually it has to do the precharge, loading the capacitors in the motor controller, before closing both AIRs. In the case of formula student cars it is a bit special, because there has to be a hardwired emergency circuit which has to open the AIRs without the BMS doing anything.

Generally the BMS itself only has the AIRs as actors, additional things may be some sort of cooling control for the cells, like fans and pumps.

5 Choosen Hardware Implementation

5.1 General

Choosing the right solution was a difficult job and to be honest, was done wrong in a lot of areas, which led to longer development time and redone PCBs.

The battery for the season of 2012 had to following parameters:

- 84 cells LiPo all in series, 7,5kWh capacity
  - 2,7V minimum, 4,2V maximum
  - -20°C minimum, 55°C maximum (discharge)
  - 24Ah
  - peak current 300A, continuous current 190A
- total system voltage range 226,8V to 352,8V
- 260x140x8mm, wide cells with tabs on top
- BMS sits at the top of the battery, isolated with a GRP layer to the tabs/connectors
- cells are not intrinsically safe, leads to 30% temperatures to be monitored
- BMS should control AIRs and TSAL directly

While the decisions on the master were pretty straight forward and simple to do, the opinions on how to do the slave where wide spread and almost everybody had something different to say, so it was though.

My first intention was to develop small daisy chained devices each monitoring and balancing one cell, seeing the biggest benefit in the fact that the slaves could fit between the cell connectors. A lot of people started telling me this was a bad route to go, most notably because of more development work and more problems due to more software compared to a slave without any logic.

I ended up choosing a BMS IC with no built in logic which monitors twelve cells and communicates over a daisy chained SPI bus with the master, which consists of a simple 8 bit MCU and different controlling possibilities for AIRs, TSAL and general purpose GPIOs. All solutions are detailed in the next two chapters.

5.2 Slave Modules

The first step in the development phase was to select an BMS IC and design the PCB, three chips were evaluated, two from Linear Technology and one from Texas Instruments.

The Texas Instruments chip is the BQ76PL536A, which can monitor three to six cells and has a stackable Daisy Chained SPI interface with additional error signal lines. A simple PCB was made to test the basic functionality. This IC lost the race because we wanted less slave modules that can monitor more cells at once.

After the first tests the Linear Technology ICs, which are named LTC6803, were set as the candidate for the slave modules. They were suggested by a couple of people, have great Application Notes and Examples and thus provided a fast way to get to a working BMS. One downside which led to a lot of wasted time was the ESD sensitivity of these devices, care has to be taken here.

The schematics are held closely to the examples from the datasheets of the device, I will briefly explain what has been done and why. Lets start with the basic pinout of an LTC6803-3.

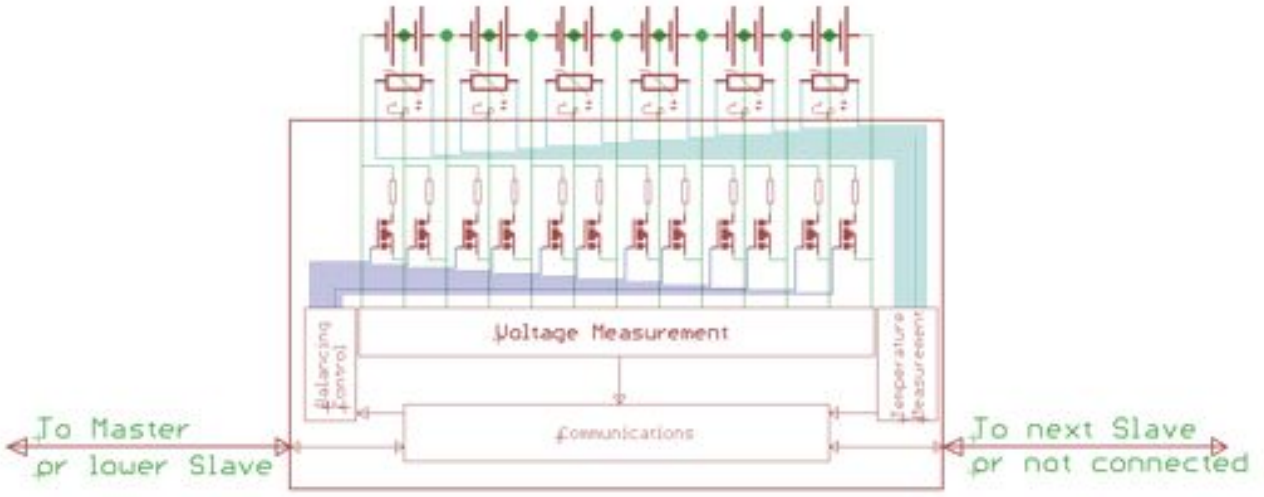


Figure 5: Slave schematic functional diagram

5.2.1 Basics

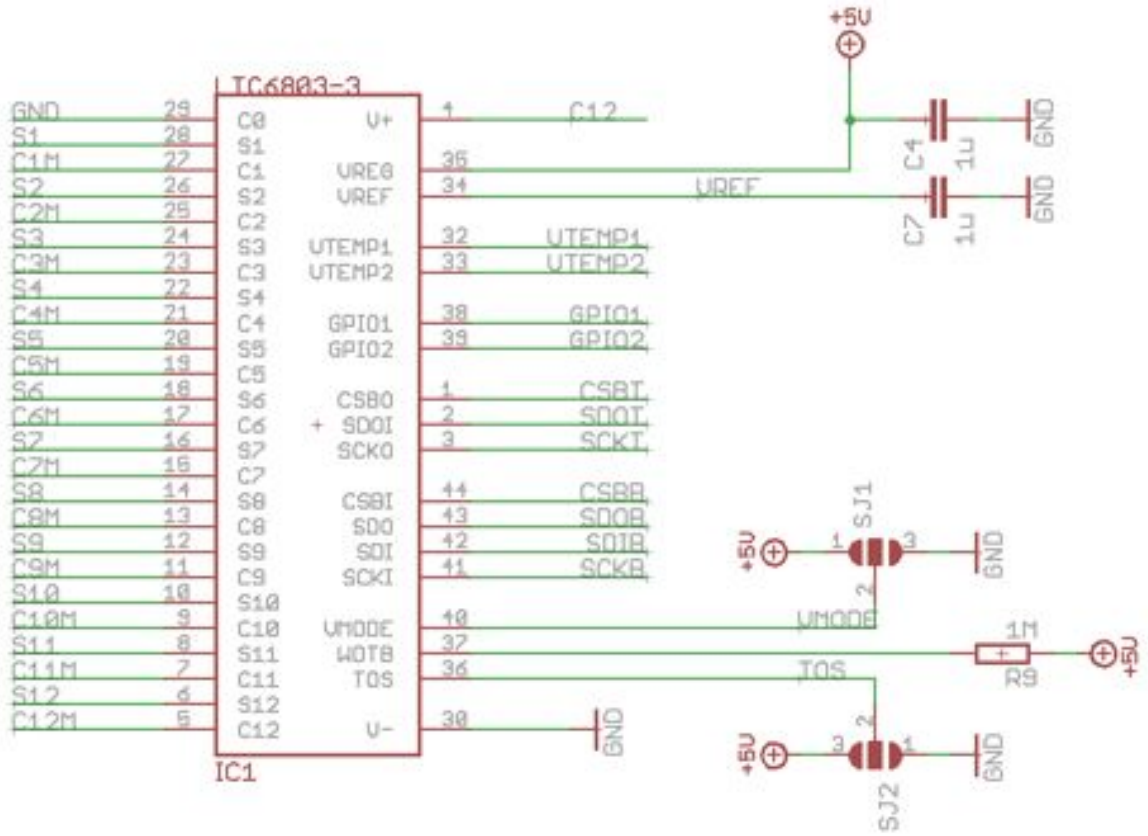


Figure 6: Basic pinout and some circuitry around the LTC6803-3

Figure 6 shows the basic pinout, on the left side are the cell voltage inputs and the balancing switch outputs. They're designed to have a maximum of one cell voltage difference between each pin. The balancing outputs have an output resistance of 20Ω, there must be a resistor connected in series or an external p channel mosfet can be connected there. The cell voltage measuring inputs should have an rc-circuit to reduce input noise. The circuits connected to the cell inputs and balancing outputs is described later on.

On the right side are the power supply pins on top and bottom, theyre just connected to the top and bottom cell input of the board, not the measuring lines. Vreg and Vref are outputs of the IC, Vref is the internal 3,065V reference and Vreg is an output from the 5V linear regulator, which can supply up to 4mA of current to external circuitry.

The Vtemp1 and Vtemp2 inputs are additional analog inputs, which are measured with the same range and accuracy as the cell inputs. They can be read in with a additional command to the slave modules.

The both GPIO pins are input or outputs, which can be configured by software, in this circuit they're used later with the analog input for the temperature sensors.

All CSB, SD and SCK pins are for communication with the master module or with the adjacent

slave modules.

The Vmode pin is an input to select whether the IC communicates with the master over a voltage based SPI protocol or if the IC communicates to next lower IC over a proprietary current based protocol.

The TOS pin is an input that tells the IC whether it is the top of stack (hence the name) device or if it should communicate with the next higher slave over the current based daisy chained bus.

The WDTB is the watchdog timer output, which is unused in the circuit.

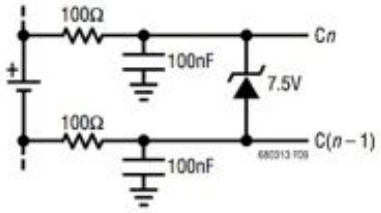


Figure 7: Cell input circuit shown in the LTC6803 datasheet

All other circuits on the slave modules are built around this IC, I will now explain what and how was designed into this board.

5.2.2 Cell measuring and balancing circuits

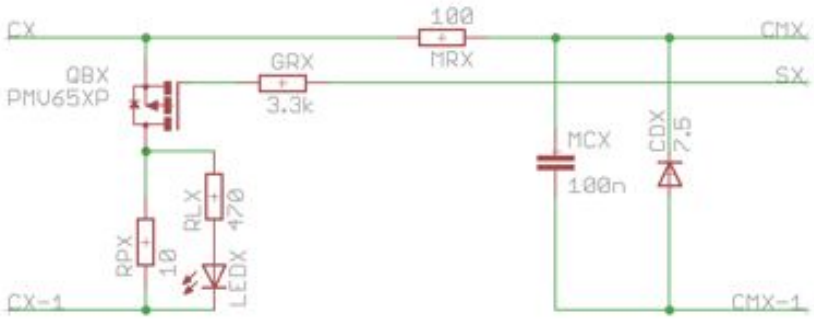


Figure 8: Cell measurement and balancing circuit that connects the cells to the IC

Figure 7 shows the measuring input for the cell voltages and the balancing circuit for each cell.

The cell input is filtered with a 100Ω resistor and a 100nF capacitor, for overvoltage protection there is a 7.5V zener diode. This is derived from the datasheet figure to the right.

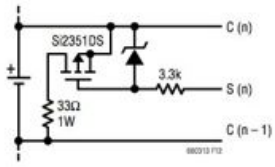


Figure 9: Balancing circuit as shown in the LTC6803 datasheet

The discharge circuit is also copied over from the datasheet, only with a different mosfet, which could handle up to 3.9A continuous.

Only the diode is left out, it is a part not necessarily needed for the circuit, it just limits the gate voltage of the mosfet. As the gate voltage is already limited by the 7.5V zener diode at the inputs.

5.2.3 Temperature Sensor Inputs

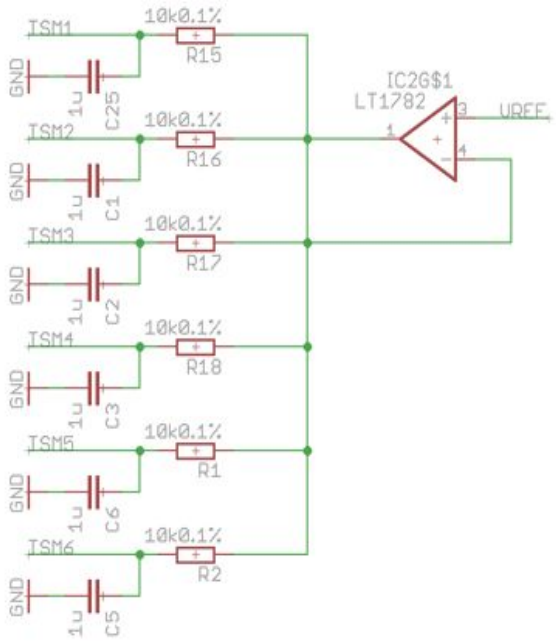


Figure 10: Temperature Sensor Circuit

To measure the temperature of at least 30% of the cells like the rules require, I implemented measuring of six cell per stack, which is more than enough.

The TSMx wires are routed to a dual row Molex twelve pin connector, which is connected to TSMx on one row and the other row is connected to GND.

The sensors are 10k NTC elements directly attached to the cells for good heat transfer.

In this configuration the NTC and the 10k resistor forms a simple voltage divider, which is filtered with a 1uF capacitor.

To have an accurate measurement the LTC6803's internal voltage reference is amplified by an operational amplifier in a unity gain configuration.

The TSMx signals are also fed into the analog multiplexer together with the full stack measurement. The analog multiplexer is described later.

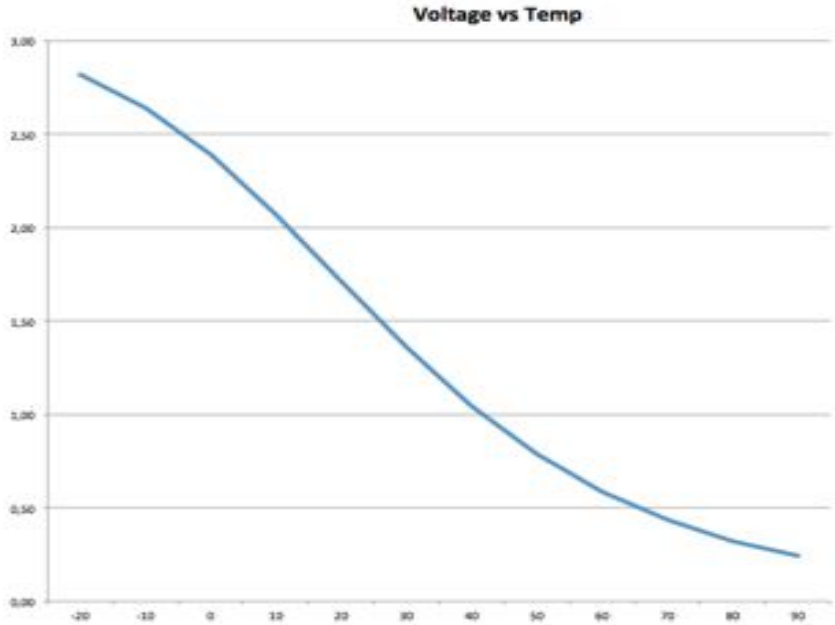


Figure 11: Temperature (x) and Voltage (y) Graph showing the output of the Temperature Sensors

5.2.4 Full Stack Measurement

The full stack measurement is a simple voltage divider which scales the input voltage down, so it can be digitalized by the LTC6803.

There are only two things that should be considered, the current must be high enough to maintain an accurate measurement and the current flowing shouldn't exceed the power ratings of the resistors. The datasheet states +/- 10uA current into the analog pins.

When a minimum voltage of 30V per stack are assumed, we get the following maximum resistance:

R = 30V / 0,000001A = 30M Ohm

This is not hard to achieve.

The choosen resistances are 332k and 20k, which yields a input/output ratio of:

k = R2 / (R1+R2)

k = 0,05682

So we at the maximum stack voltage of 50,4V (12x4,2V) we have:

U2 = U * 0,05682 = 50,4V * 0,05682 = 2,864V

The maximum input voltage of the LTC IC is 3,065V this limits the stack voltage to:

U = 3,065V / 0,05682 = 53,94V

This voltage would irreversible damage the cells, there would be worse problems than a broken BMS Slave.

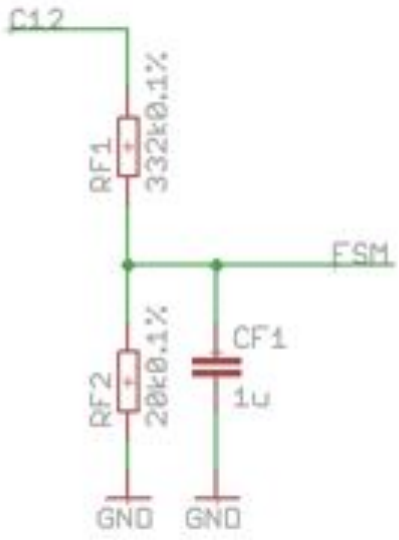


Figure 12: Full Stack Measurement circuit

5.2.5 Analog Muxer

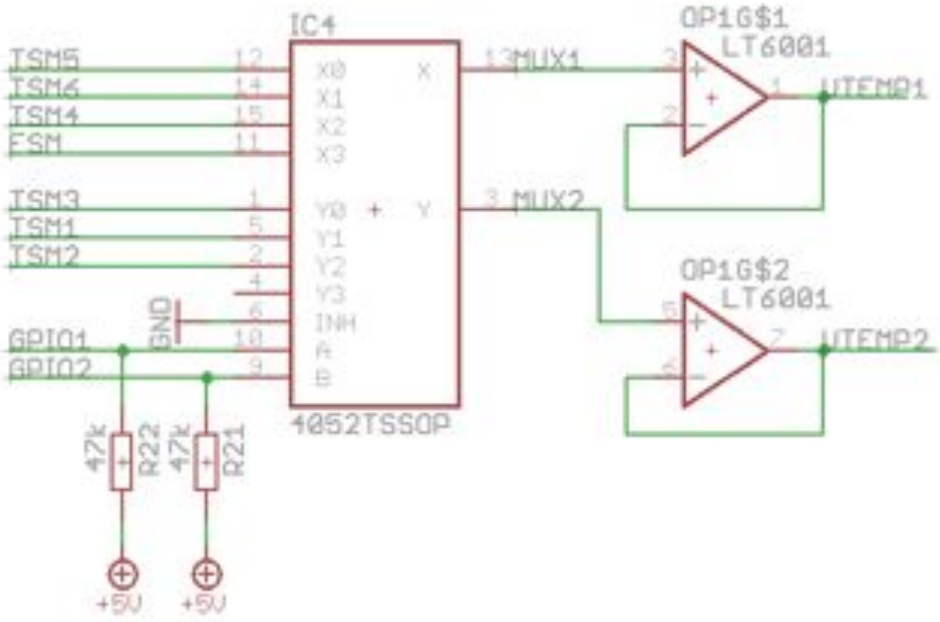


Figure 13: The analog multiplexer with an operational amplifier at the output

Because the LTC6803 only has two analog inputs and we need at least 4, there had to be something done. Gladly the datasheet pointed to a solution for this problem, by using the GPIO outputs to drive an analog multiplexer the total analog input count is increased to eight.

This way all six temperature sensors and the full stack measurement can be fed into the two Vtemp pins of the LTC6803.

The output of the mutliplexer is than amplified with a unity gain circuit, to have a stable measuring signal at the ADC.

5.2.6 Communication

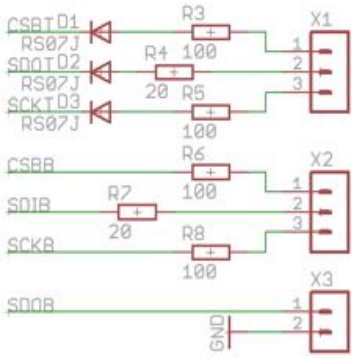


Figure 14: Resistors and Diodes at the communication pins

The resistors and diodes connected to the communication pins are shown in figure 14, these are exactly as specified in the datasheet.

For further information please consult the datasheet, this basically is a blackbox for the circuit design and although I did read and understand the concept, redundant explanations to the datasheet are omitted here.

5.2.7 Power consumption of the parts

One thing to consider is the power consumption of the parts that get power from the LTC6803 5V linear regulator, it should not exceed 4 mA in total, otherwise the LTC6803 can overheat and stop working. I did a table with the minimum and maximum current consumptions, to check if there is anything critical.

M74HC4052TTR	0.16mA	Analog Multiplexer, maximum quiescent supply
LT6001	2x0.034mA	Operational Amplifier, maximum supply current
LT1782	0.1mA	Operational Amplifer, maximum supply current
Temperature Sensors	6x0.3065mA	10kΩ resistance if the NTC is 0Ω, the amplified reference is 3.065V. Current is then calculated 3.065V/10kΩ = 0.3065mA each.
Pull Ups	1x0.005mA 2x0.11mA	1MΩ WDTB pin, 2x47kΩ GPIOs
Total	2.392mA	This leaves enough room for errors.

Table 1: Parts supplied from the 5V linear regulator each with the maximum current consumption

5.2.8 Layout

Figure 15: One of the final slave modules built, note that they're covered with epoxy

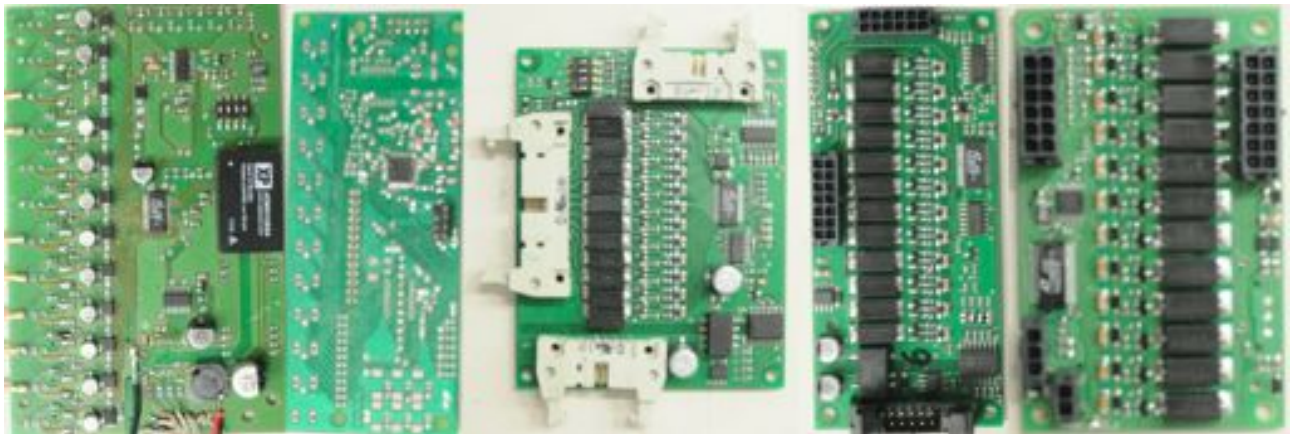


Figure 16: Linup of some of the development steps the slaves went through

In the final layout I tried to keep the board as small as possible, the final PCBs are 90x50mm in size, which was good for our battery assembly. There are some problems with this layout, as you can see, the pads for the large resistors are to big, the resistors float around once the solder is liquid and don't line up properly, this can easily be fixed.

5.3 Master Module

The master module is built around a AT90CAN128 which is an 8 bit AVR microcontroller featuring CAN besides the other common interfaces. This choice may not be the smartest done, as the code is more complex and especially more computationally intensive than I thought when choosing the controller. Although it works really well now, the measuring cycles could be a few milliseconds faster when using a faster controller. I don't know if the controller will handle the microSD card well when it is used, this uses a lot of power from the microcontroller.

There are five areas isolated against each other, first there is of course the microcontroller which has its ground plane connected to the car ground. Then there is the area for slave communication, which sits on the negative pole of the battery. The battery output measurement and 40V measurement is connected to the slave communication area when the battery is active, but is isolated when not, so has its own area. The last two areas are the one connected to the precharge resistor and the CAN Controller, which is not needed because CAN ground is car ground in some other devices in the car.

There are two safety circuits, one of them is always good, the other can be omitted. One prevents reopening of the AIRs after the emergency has been pressed by using two flip flops which have to be set with the microcontroller and have a reset input connected to the emergency chain, this prevents reopening in a microcontroller deadlock / hardware damage. The other circuit monitors the precharge and prevents closing the second AIR while there still is precharge current flowing.

In the following chapters only the important parts of the schematics are shown, a lot of decoupling capacitors are not shown, they are needed but don't add functionality. Everything in this document is written to understand the full schematic, not to have the schematic here. If you want further information you should look directly in the full schematics.

On the next page there is an overview of the finished topology for the battery:

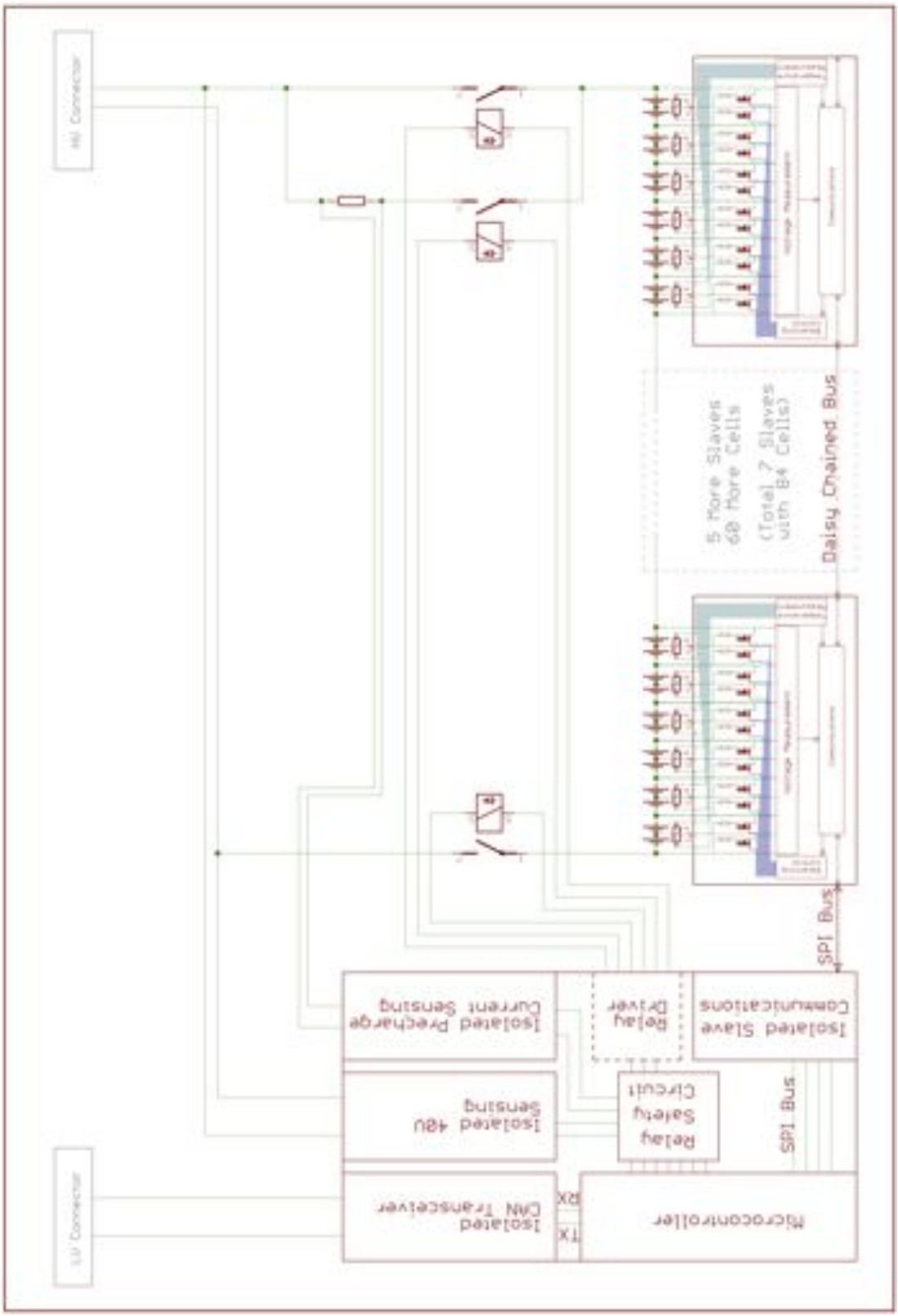


Figure 17: Overview of the whole BMS

5.3.1 Microcontroller

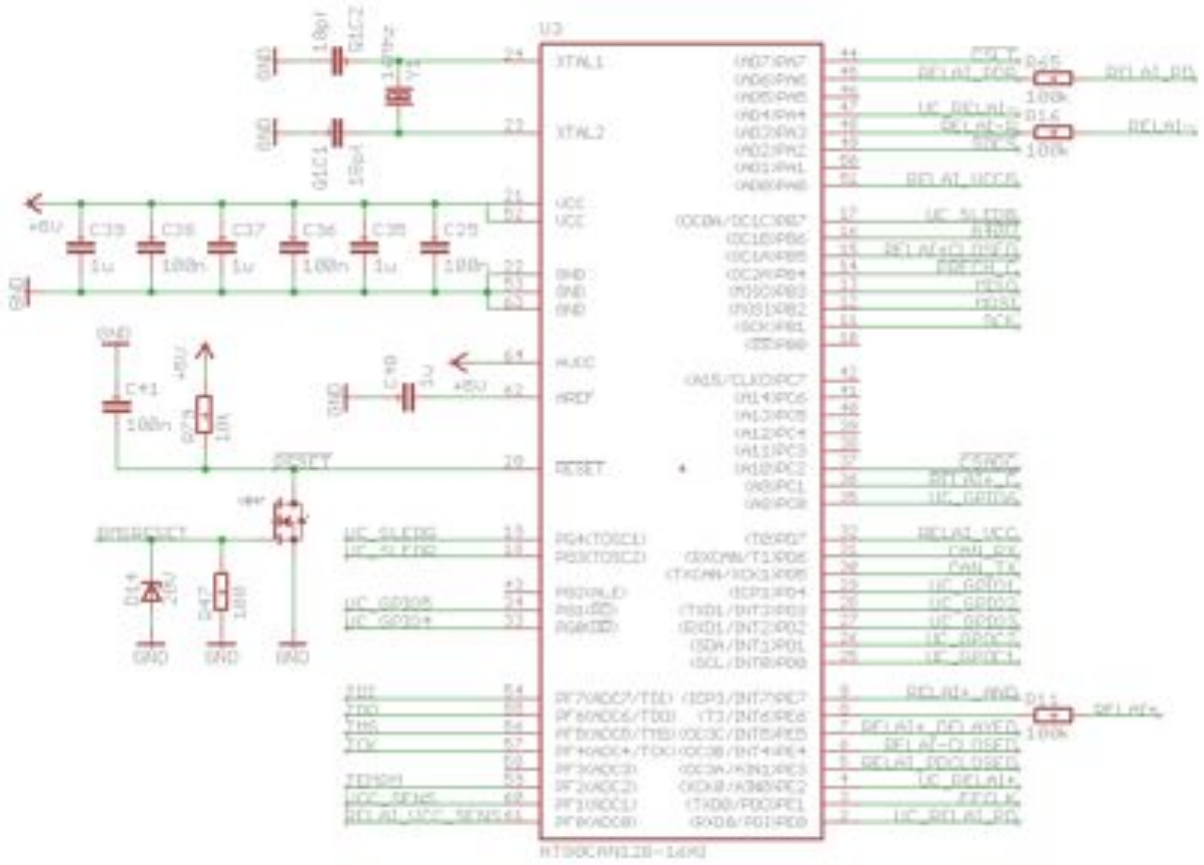


Figure 18: AT90CAN128 with basic circuitry and all signal names

I will only briefly explain the circuit around the main MCU, it is an AT90CAN128 which is a part from the famous AVR 8-bit MCU family with CAN Bus support, besides that, there is nothing special about this chip.

Beside decoupling caps, crystal circuit, there are two things to mention. First there is a mosfet to have an external reset output, though this part is unpopulated for stability reasons, it can help when debugging the part. The second thing are the 100kΩ resistors, they let the controller see signals that control outputs, but it can't overwrite logic signal. This prevents the IC from randomly changing signals in case of a program error.

Also, there is a connector for a three color LED that can visibly signal battery status.

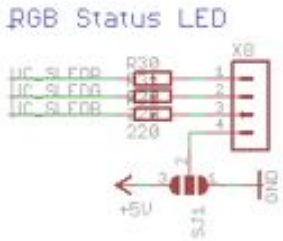


Figure 19: The three color status LED

5.3.2 PCB Temperature

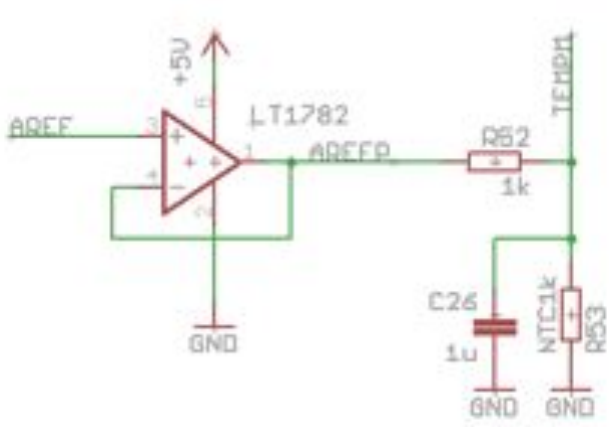


Figure 20: Unity gain amplifier with a NTC in a voltage divider configurationz

On one analog input, a temperature sensor is connected. The internal reference voltage of the AVR is amplified and then fed into a simple voltage divider consisting of an 1kΩ resistor and a 1kΩ NTC.

This yields a similar behavious as described for the slave temperature sensors.

5.3.3 GPIO

Ofcourse a good universal BMS should have some GPIOs, exposed on a dual row 14 pin Molex connector, are six general 5V input/output, directly connected to six pins of the MCU, two of them are a serial (UART) interface. The are two additional GPIO pins which are connected to n mosfets in a open collector configuration, without pull-ups, these pins can be used to switch on relaiys in our car a mosfet GPIO is used to close a relay connecting the HV DCDC converter (350V → 12V).

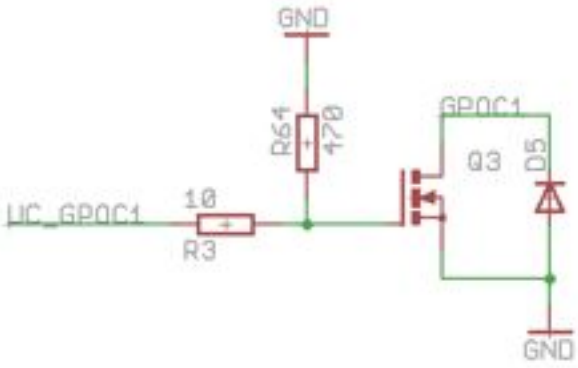


Figure 21: Open Collector GPIO circuit

5.3.4 Power Supply

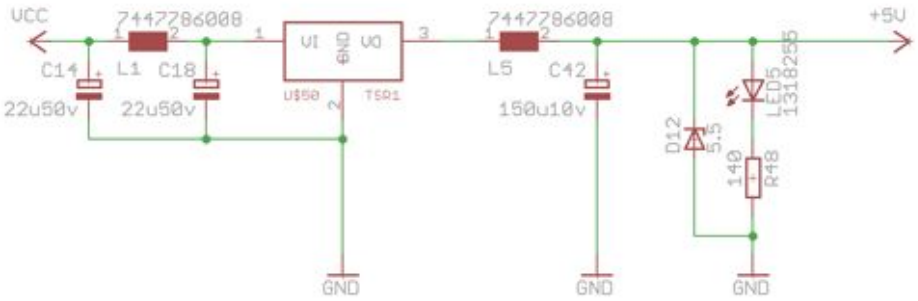


Figure 22: The DCDC converter and circuit that converts 6 to 40V into 5V

The BMS master is supplied by the vehicle power supply with usually 12 to 15 volts, to get better efficiency there is a small DCDC converter that does this job.

I did an estimation how much current is drawed as a maximum value from the 5V power source, these figures are way beyond the normal operation and show that the 1A rating of the DCDC converter will sufices:

MCU + Logic	100mA
Status LEDs	60mA
Each DC-DC	4x100mA
SDHC Card	50mA
Isolators / Optocoupler	4x50mA
Total	910mA

Only the supply for the AIRs is a special connection, as they have to be directly powered off the emergency circuit, which can't be controlled by the BMS. The BMS uses mosfets to switch the ground connection for the AIRs and the +12V for the AIRs are coming directly from the emergency circuit.

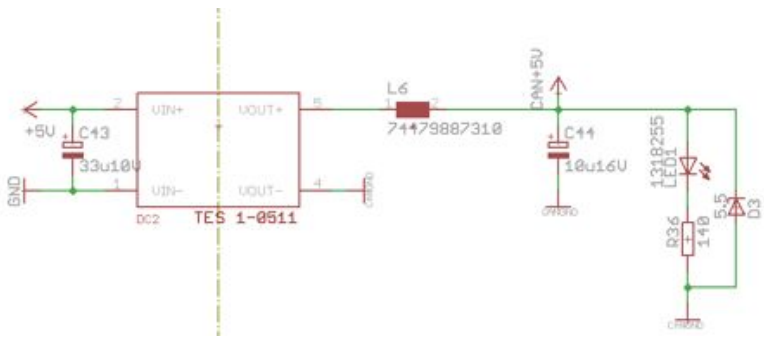


Figure 23: Isolating DCDC converter that supplies one of the isolated areas

The isolated areas (precharge, battery output etc.) are powered by isolating DCDC converters, they have 5V as input and input and thus serve only for isolating purposes.

5.3.5 Supply Measurement

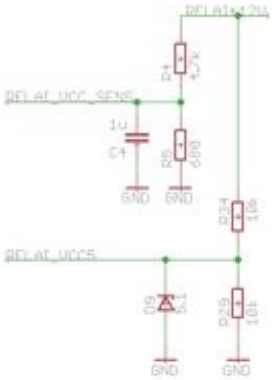


Figure 24: Supply measurement for the relai supply

The supply voltages are both measured with a simple voltage divider circuit which feeds the voltages to the MCU, which will prevent closing the AIRs when a voltage is to low or will shutdown the complete tractive system if an under voltage condition on the low voltage supply may occur. This can be incredibly important, because with a low oscillating supply voltage to the AIRs, they can start to open and close randomly, which in turn can kill them by melting together the contacts. This is an dangerous condition, because the battery can't be disconnected from the HV system and thus the HV system is under permanent high voltage.

Also worth a note, the relai supply voltage is converted to a 5V signal with a simple voltage divider and a zener diode, yielding a constant true signal up to about 5V of supply. This circuit is there to have a fast digital signal for the emergency circuit flip flops, which are explained later.

5.3.6 CAN

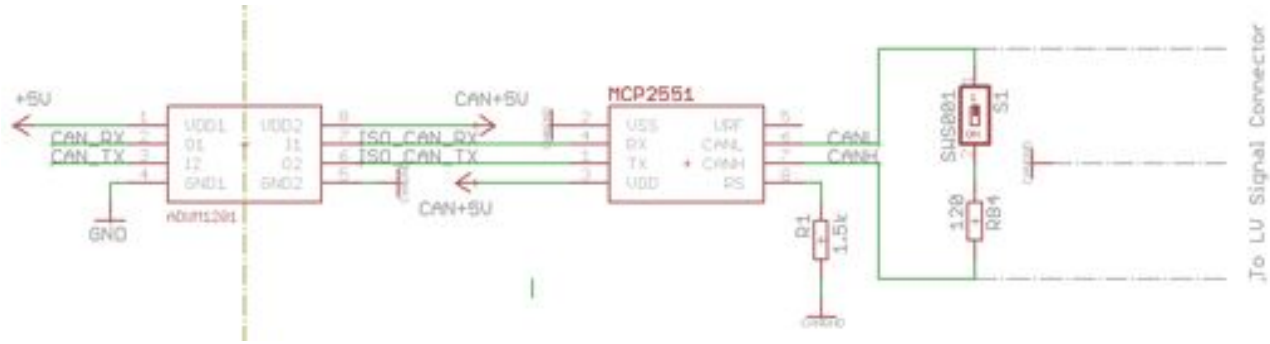


Figure 25: Two Channel Digital Isolator with CAN Transceiver and Termination Resistor

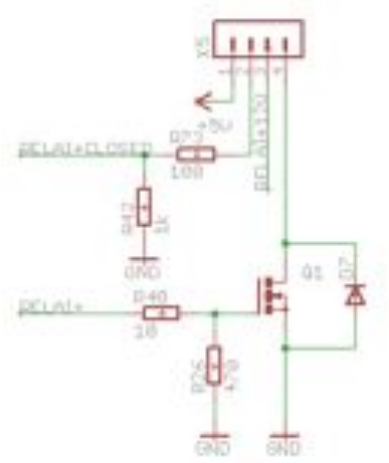
To be flexible to all car design concepts the CAN bus is isolated to the rest of the low voltage ground. This gives the possibility to avoid some EMC problems when only using CAN to communicate to the rest of the car.

The circuit is fairly simple, there is a two channel digital isolator which feds its signals into a CAN transceiver. There is a 1.5kΩ resistor to adjust the slew rate and a 120Ω resistor with a dip switch to have a selectable termination on board.

The digital isolator is a ADUM1201 or similar, the ADUM family are based on inductivity and can easily handle data rates of up to 1 Mbit. Besides decoupling capacitors there is no external circuitry needed to handle digital signals.

The power comes from a 5V to 5V isolated DCDC converter as described in the power supply section.

5.3.7 Relay Driver



The relay drivers are fairly simple, a mosfet with a gate resistor and a pull down resistor for fail safe reasons. The mosfet is protected with an additional free wheeling diode.

The other two connections to the connector are connected to the auxiliary contacts of the AIRs, so the MCU has an active high feedback when the relay is closed.

Figure 26: Example of a Relay driver circuit

5.3.8 Slave Communication

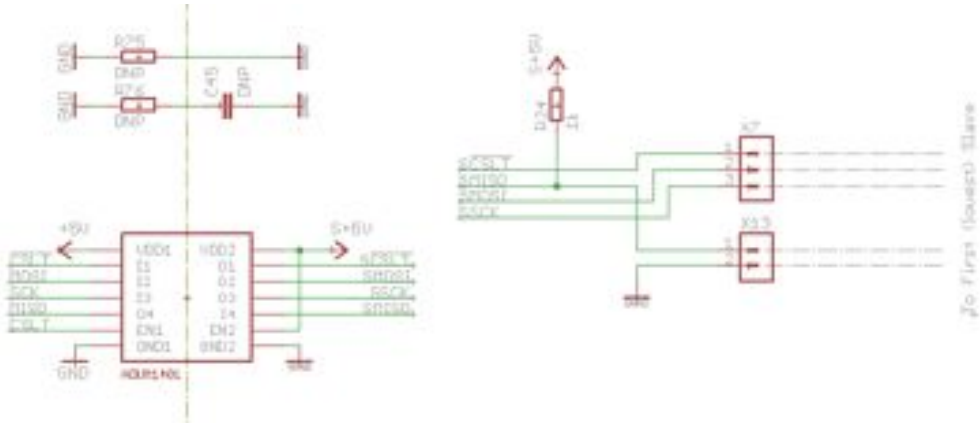


Figure 27: Digital Isolator and Connectors for the Slave Communication

The slave communication must be isolated, as the slave modules have their ground potential on the cells, the slave the master communicates with is the lowest slave and is connected to the negative pole of the battery pack.

The isolation of the SPI bus is again done with a digital isolator based on inductivity from the ADUM series, the only difference to the one used for the CAN isolation is that this one has 3 signals from the master to the slave and one the other way. The allows to transmit a normal SPI bus with MISO, MOSI, SCK and a Chip Select line. There also is a pull up resistor for the open drain output line of the slave.

Other than that, there are only two resistors and a capacitor to couple the ground planes together, they can be left out if there are no EMC problems, if the slave communication experience strange behaviour or a lot of data errors, the capacitor and the resistors can help minimizing the problems.

5.3.9 Relay safety circuit

This is one of the complex parts of this schematic and one of the most important. Here two things are done, the first is to prevent re-closing of the AIRs in case the MCU is deadlocked and an emergency button is pressed. When the pressed emergency button is released again, the AIRs would close right away, resulting in probably unwanted voltage on the HV system, this can be an dangerous condition with the possibility of serious injury. There is a flip flop circuit to prevent this. The second security circuit uses the outputs of the microcontroller / flip flops, the output of the circuit to check for >40V at the battery HV connector and the precharge current measurement output to close the AIRs only when there is voltage on the HV and there is less then a defined threshold current on the precharge.

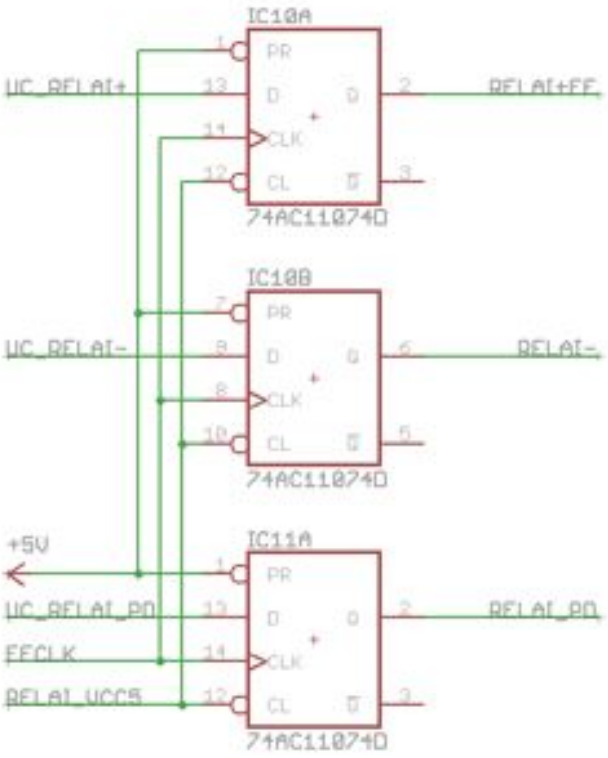


Figure 28: These flip flops prevent accidental re closing theAIRs

The circuit to prevent accidental re closing the AIRs consists of three flip flops, one for each relay.

The microcontroller has to create a rising edge on its FFCLK output, only then the inputs of the flip flops are carried over and saved at the output, this way the microcontroller can only change the outputs as long as the code is running. This is a very useful feature to prevent two things, first if there is a reset loop in which the microcontroller gets reset uncontrolled, the outputs could open and close the relays, which must be avoided as possible.

The clear inputs of the flip flops are connected to the 5V signal, which is high only when there is voltage for the relays. This keeps the outputs low as long as there is no voltage to close the relays, this also prevents a solid closed state of the relays when an emergency button is pressed. As soon as the emergency button is pressed, the voltage for the relay coils is 0V, then the RELAI_VCC5 is 0 and the output is 0V, even if the microcontroller is unaware of the situation.

Figure 29: Relay logic preventing closing of both AIRs without waiting for a successfu with simplified timer circuit

The circuit in figure 29 prevents the AIRs from closing without doing a proper precharge cycle. The first and gets both closed signals from the charge and minus relays, these signals come from the auxiliary contacts as described in the relay driver section. The RELAI+FF signal comes out of the flip flops. So, this and gate only gives an output voltage when the precharge should already be running. To be sure that the relays are not only closed, but are closed long enough to conduct current, there is a 100ms delay timer on the output. This is necessary because relay contacts can bounce, for the AIRs the closing time is specified with 75ms, hence the 100ms delay. The delay circuit can be seen in the full eagle schematics.

The inputs to the next and gate are the signals coming from the 40V and precharge detection described later, the B40V signal is true when there are more than 40V present at the battery output. The PRECH_C signal is true when there is no current over the pre charge resistor. The output of this circuit directly drives the mosfet switching the AIR on the positive battery pole.

The precharge is one of the crucial parts of operation for the BMS, the worst case are damages relays that can't be opened which leads to an uncuttable high voltage on the tractive system, this can lead to injury or death of the operating persons. The it is very important to do the precharge right.

The precharge works in the following steps:

Step	Condition for next step	Error conditions
Start command	Less than 40V, no precharge current	More than 40V or precharge current
Close precharge relay at resistor (note: this step is mainly necessary because the smaller precharge relay should not be closing under the full precharge current)	Wait closing time of relay	More than 40V or precharge current
Close AIR on non precharge pole	Wait closing time of relay	-
Check that precharge current is applied after closing of both precharging relays, typically 100-300ms	None, only error checking	no precharge current
Safety time which a precharge should take minimum, on our car typically 3s, this checks that there is a capacitive load (broken wire detection)	None, only error checking	No precharge current or voltage < 40V
Check Pack voltage vs output voltage (voltage drop across precharge resistor)	Voltage < 15V for our car	Time exceeds normal precharge duration (e.g. short)
Close second AIR, precharge done	-	-

5.3.10 TSAL Circuit

The circuit driving the TSAL is a simple NE555 timer circuit producing a 50% duty cycle PWM with variable frequency. This has been verified and calculated with LTSpice, the files are with the data that come with this documentation.

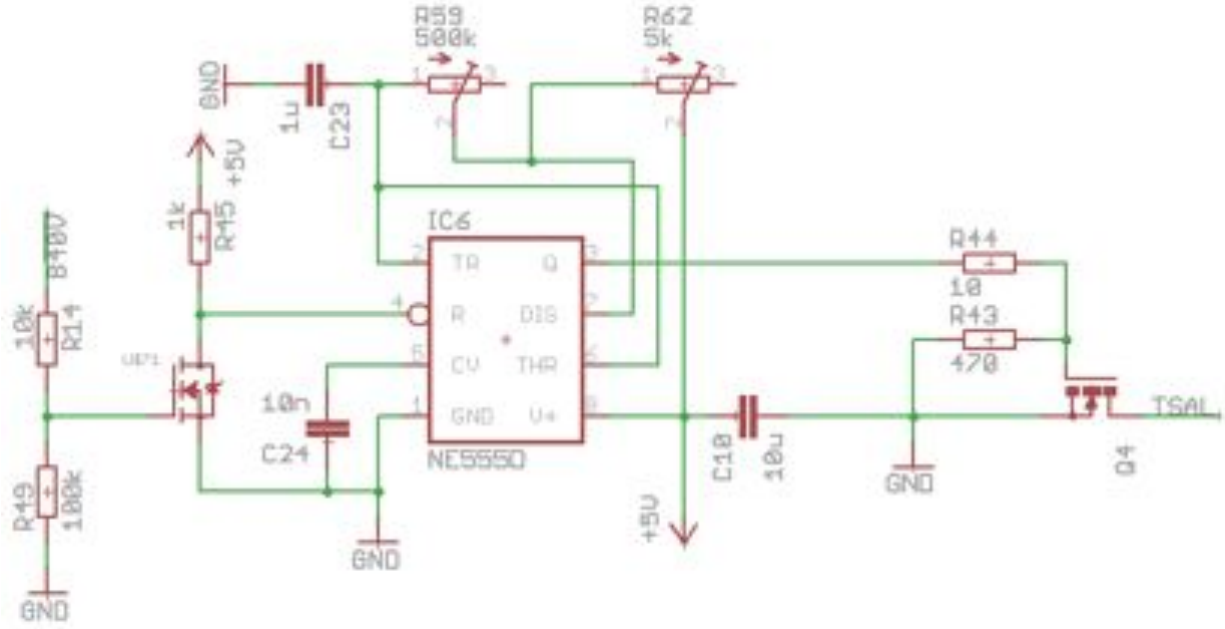


Figure 30: TSAL Timer Circuit using a NE555 Chip

5.3.11 Battery Output ADC Measurement

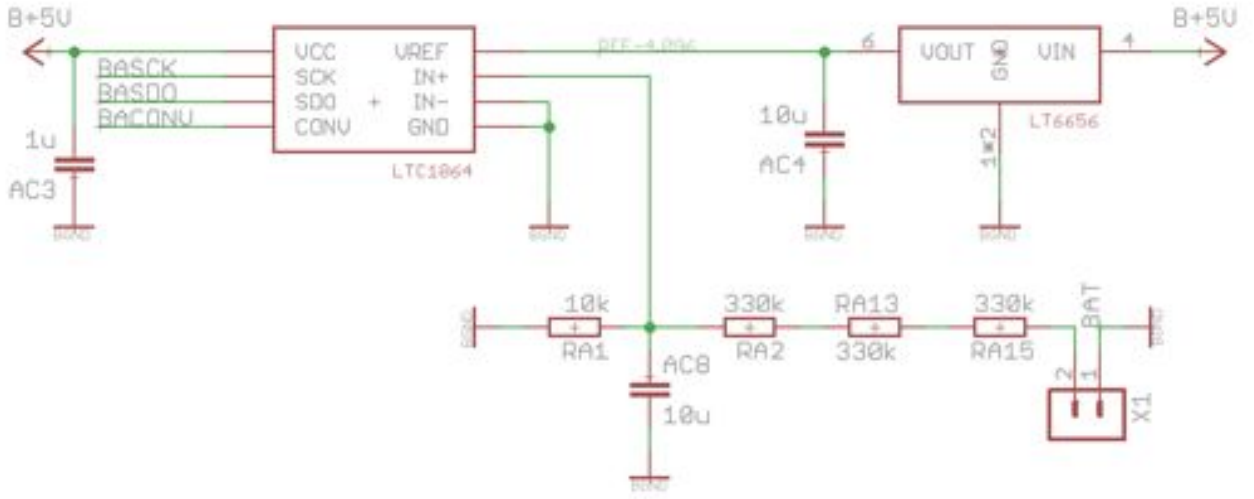


Figure 31: SPI AD Converter with reference and voltage divider used to measure the battery output voltage

The battery output voltage, more precisely the voltage at the battery connector, is monitored for security and debugging reasons. This circuit can be omitted and the battery will work without any major security issues.

Actually not yet used in software, the output of this circuit was intended to check the precharge voltage at the output and to decide if the second AIR should be closed or not. Since the motor controller already does a measurement that is sent to the can bus, this circuit is obsolete in our system. However, if there is no external voltage measurement and total safety is required, this circuit can prove useful.

The circuit consists of a 4.096V voltage reference, a 16 bit Analog to Digital converter with SPI and a 1:100 voltage divider. This results in a measuring range of 0-409.6V, this could be extended, but the layout would have to be changed, because the detection of more than 40V at the battery output relies on the 1:100 divider.

A future version of the BMS should include separate connectors and voltage dividers for the 40V detection and for the SPI voltage measurement. Generally it might be a good idea to rely on ready made analog to digital converters for high voltages, as the small converter proved unreliable when voltage peaks are experienced.

5.3.12 Battery Output >40V Detection

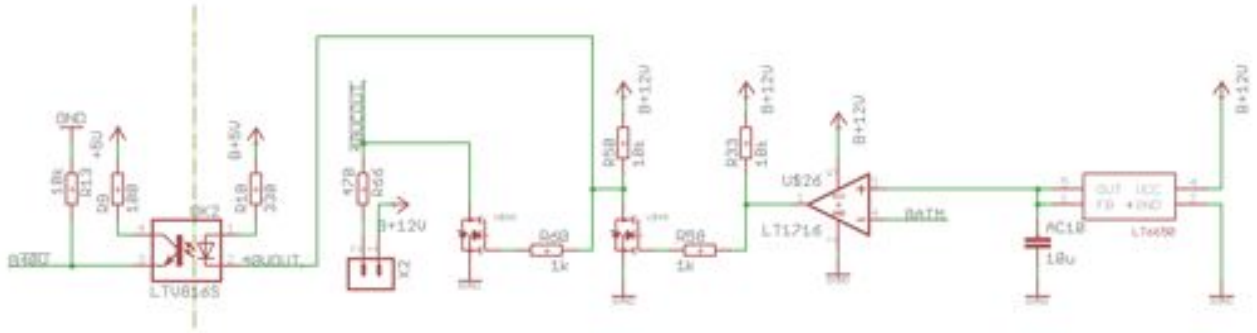


Figure 32: Circuit to detect >40V at the battery output and to drive the LED and the Optocoupler

The detection of more than 40V is one of the essential functions of the master, this circuit directly drives the LED signaling that more than 40V are on the battery output. It also drives an optocoupler, which outputs the signal B!40V to the relay safety circuits. The at the output of the comparator are only to deliver more current than the comparator for the 40V signaling LED, this could be simplified down to one inverting stage with the +/- inputs of the comparator flipped and ommitig the first inverting mosfet.

We also had problems with some sort of oscillation sometimes, which resulted in a stable switching only at more than 43V, sadly we never had time to investigate the problem.

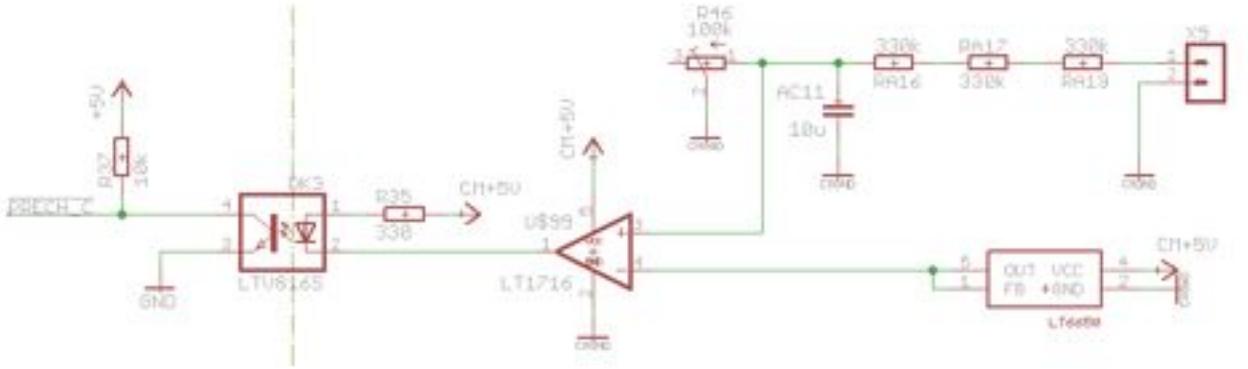


Figure 33: Circuit to detect current over the precharge resistor

5.3.13 Precharge Current Detection

The circuit to detect current over the precharge resistor is similar to the one measuring the 40V at the battery pole output, except that the voltage divider is not at a fixed 1:100 range and the comparator directly drives the optocoupler. The voltage divider can be adjusted to set the treshold at which the circuit sends a “precharge current free” message to the microcontroller and to the relay safety circuit.

5.3.14 microSD Card Slot

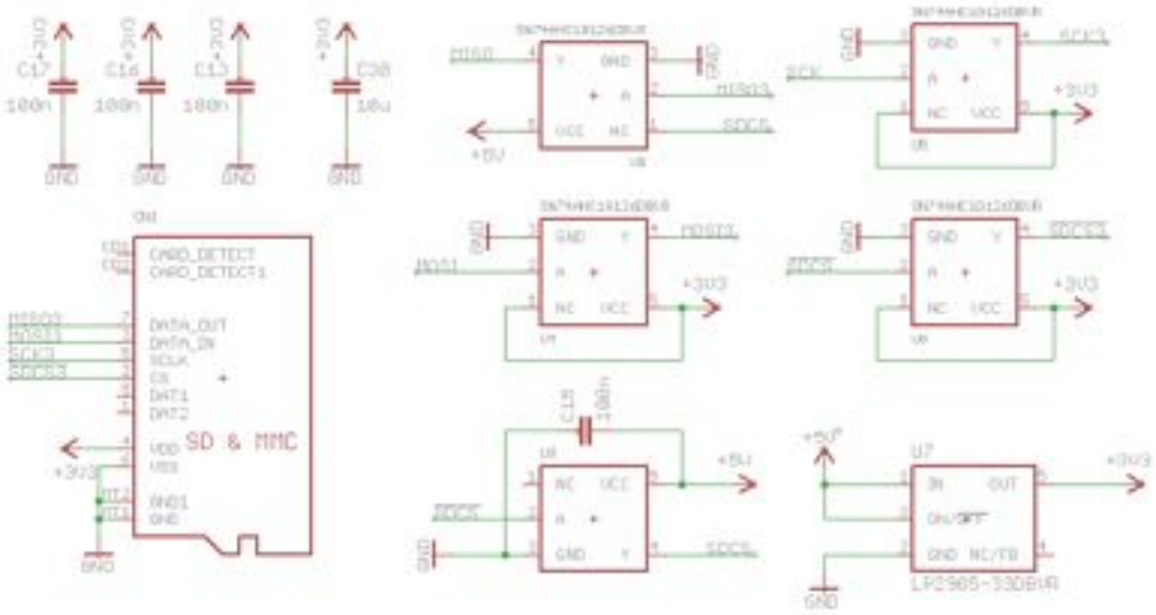


Figure 34: microSD power supply and level shifters

This provides a 5V to 3.3V linear regulator and level shifters, this circuit has not been tested but worked in other circuits.

5.4 Costs

Unfortunately no one kept track of the costs, as this project sadly was a one person only problem the cost management fell of the time table. Especially because a lot of parts where reused or ordered in quantities that were enough for all boards made.

But yet, I can estimate the costs for the boards. The costs can vary a lot depending on the supplier and quantities ordered, especially for quantaties less than 25 of one part the prices vary as much as 500% percent.

For the slaves this meant for us 15€ per PCB made, adding 5€ per PCB for the laser cut solder paste stencil. The part costs are roughly 50€ per board when ordering quantities for nine boards, with the LTC6803-3 weighing in at 20€ each. A lot of parts can be sampled if there is enough time. This made about 70€ per slave board.

The master boards were always hand soldered and thus there was no stencil which would have cost a lot for only one piece. One board came in at 50€, with a lot of parts already in our stock and some sampled parts, especially the HV Linear regulator, we came to about 100€ part costs per master. So this adds up to 150€ for the master.

All other connectors and cables come at around 50€, so the final BMS without the development costs did cost us around 700€. With all development costs this would be around 3000€, which is a lot but should be in the normal range.

6 Software

The software is written solely in C for the AT90CAN128 on the master, since the AT90CAN128 is a device of the 8 bit AVR family, care has to be taken when programming with larger data types than 8 bit. This can result in locks of several seconds for a single calculation.

The program has a main loop and interrupt driven input/output where possible. So the SPI, UART and CAN communication is interrupt driven and as less as possible of this is done in the main loop, generally data is put into a buffer and sent out until it is empty, for received data it is received until a defined buffer is filled for SPI, a specified character is reached or a CAN message if fully received.

All calculations, checks and everything else is done in the main thread by calling functions that always execute only one step in a calculation. There are basically four tasks running in the main thread, the first one is controlling the relay state including receiving can commands, one is sending data over the can bus, one is reading data from the slaves including calculations and checks for the data and the last task is an analog to digital conversion of the supply voltages of the BMS.

The relay controlling code does the precharge, reacts to can commands telling the battery to enter a state and does checks for pressed emergency buttons.

The code sending data over the CAN Bus waits for a flag to be set and then starts sending all relevant data to the CAN bus, clearing the flag stops sending and setting it again will continue at the previous position.

The slave reading code starts all necessary SPI commands until all data is transmitted, once this is the case all necessary conversions are done, checks are made if cells are in a fault condition like under voltage or over temperature, when done with the conversions it sets the flag to send the data over the CAN Bus.

Outside of this, there is a timer routine which generates an interrupt every millisecond, there are counters and general timekeeping code in the interrupt. There is for example a milliseconds counter which checks, that measurement data from the slaves is collected more often than 2 seconds and an error code is issued if this time elapses.

I tried to do meaningful comments in the source code, so for further information the source itself should be consulted, it is at least explained what every function is doing and when and how it should be called.

Charging and balancing are not done, due to time problems, which seems normal in formula student team. The balancing algorithm is a simple approximation, which tries to guess the time the balancing resistors have to be turned on the get all cells to the voltage of the lowest cell. This algorithm can only be started in an idle condition and is not used while charging. The charging is done compltely manual by closing the AIRs and then hand controlling the charging rates.

7 Datasheets / Information

7.1 Slave datasheet

	Unit	Min	Avg	Max
Cells		4		12
Temperature probes				6
Power consumption operating (calculated)	mA	1,2	2,29	3,68
Power consumption standby (calculated)	mA	0,3	0,49	0,68
Measurement accuracy	Bit		12	
Measurement error	mV		1	2
Measurement cycle for 12 cells	ms	11	13	15
Measurement cycle for 6 temp proces	ms	8,4	10,2	12,3
Communication interface to host			SPI	
Communication interface speed	Mbit		1	
Maximum discharge/balancing current	mA		50	
Dimension (height depends on connectors)	mm	90x50x10mm		90x50x30mm

Table 2: Slave working Parameters

7.2 Master datasheet

	Unit	Min	Avg	Max
High Voltage	V	20		400
Low Voltage	V	6		40
Power consumption	mA	10		1000
Precharge switch point	V	1		30
CAN Interfaces			1	
Serial/UART Interfaces			1	
General IOs			6	
General IO max current	mA		+/- 40	
Open collector IOs			2	
Open Collecotor IO max current	A		5	
Relai ports			3	
Relai ports max current			5	
Isolation voltage	V		1500	
MCU			AVR 8-bit	
MCU Debug			JTAG	
TSAL Frequency	Hz	0.1		10

8 Problems

Oh, where do I start with all the problems I had developing this BMS.

The first trouble started when assembling the first PCBs, everything looked really good, but there were always some sort of shorts inside the BMS ICs, it seemed something always broke the ICs when plugging in the batteries, a long quest to search the error in the schematics were started. After consulting several professors, other student, people from different companies, no one noticed a mistake. So, I thought it must be something in the layout, because after doing three PCBs with the same result, I really couldn't think of anything else, since the schematics basically had been copied entirely from the datasheet at that time. Well, what can I say, after almost two months of searching my solder iron broke and I bought a new one. The old and the new one had an ESD safe label. From that point on all PCBs worked fine, I soldered more than ten ICs and never had a problem again. Maybe the old solder iron was broken and so did some strange stuff to kill the ICs, but I never tried to solder with another iron from that point. So keep in mind, ESD sensitive sometimes can be a serious warning on how to treat the ICs and it can't event be considered that doing most ESD precautions is enough, sometimes you have to do them all.

In general production was a problem that I never thought about earlier, putting together one module is a days work, soldering two masters and seven slaves for two battery packs takes at least a full week. Since I had to do a lot PCBs more than once, I'm sure I soldered at least 150 hours during this project, although the final batch of slave modules was populated by an automated machine.

A rather disappointing problem I had was that most of the time I had no help. There was always someone to ask, but there were no teammates, no one on the formula student team ever overlooked my schematics or layouts, although I did send them around multiple times. I even tried to find ways to keep them up to date, for example all schematics, layouts and source is in an accessible git repository. This did cost me quite some extra time and money. A second person reviewing everything can cut development time in half, even if the person only spends 1/10th of the development time. I even produced a PCB where a ground connection was left out, this was fixed easily, but could have been avoided. As a single person you can't see everything, especially after staring at a layout for a few days. When starting such a project, either plan with enough time or be really sure there is someone who really supports the project. I did this in a team which was hopelessly under-staffed and the needed support came to late, I would really advice not doing such a project when there is no one you know will help. Never rely on statements made at the beginning of a formula student season, at least 30% of the people will disappear during the season and 50% are complete morons from the beginning, this will leave only 20% people left. With enough budget is possible doing it solely by yourself, but you should plan with 6 months full time working.

Another thing which caught me even though I should have knew it is the isolation distances, I really tried everything to have good spacing between the battery voltages and the low voltage vehicle power, which I did pretty well. But I forgot that there should be at least the same isolation distance between the positive and negative pole of the battery, which led to a 3mm spacing connector being used to get the battery voltage to the master PCB, this works in normal conditions, but we had an overvoltage event while testing which destroyed the PCB.

After everything was assembled and tested successfully in the battery, the day came that we did put

everything together, the system integration started. We had no unusual problems, but the black magic of EMV killed our hopes for quite some time and kept us busy for good two months. The counter measures ranged from developing a completely new BMS (daisy chain to replace SPI) to wrapping wires in aluminum tape. A combination of ferrite cores, a lot of aluminum tape, a new BMS, metal casings near the power electronics and smarter cable management finally solved the problem, or at least made it working since EMV is always an issue even if it can be minimized.

9 Summary and Forecast

This project brought me a stressful time, but I couldn't have learned more anywhere else in that time. I started out as an electronics interested mechanical engineer with a solid IT background, I did some circuits before and had etched some PCBs at home. Now I'm able to develop whole systems for a tractive car system, I learned a lot about electronics, manufacturing and quality control. When people started to see that the whole project was working, I even got questions from different people what and how I did.

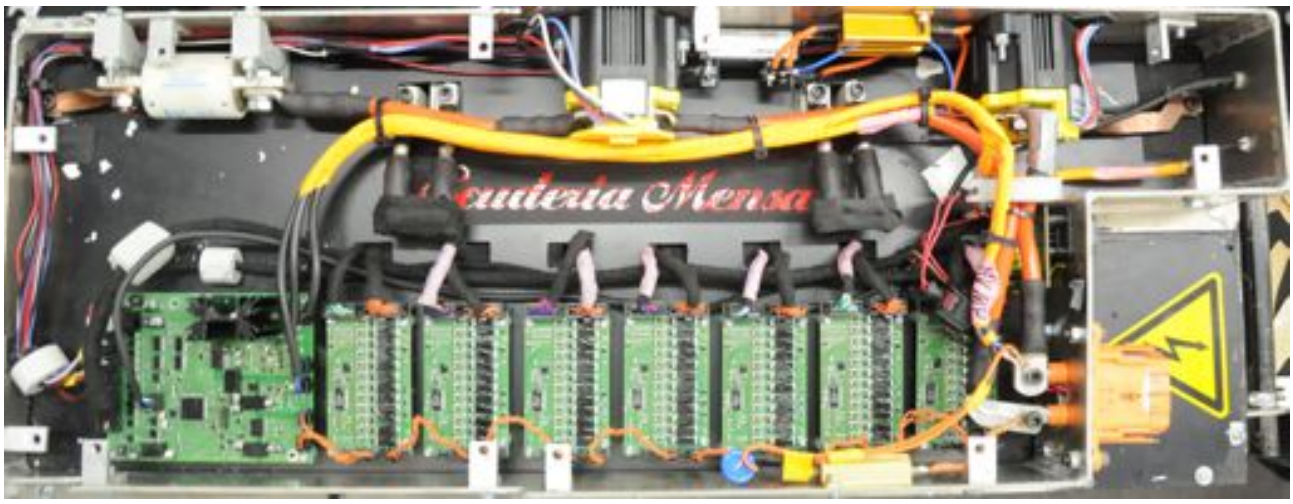


Figure 35: The finished BMS installed in the electronics compartment of the battery case

When starting the project I didn't realize that I was going to do something that even OEMs still don't get done in the way they want. Also there seems to be a market for prototyping BMS systems, there would have to be a lot of work done to sell this BMS, but it may be possible. Especially the software problems in BMS seems to bother people, for example finding a good balancing algorithm. This project will be open sourced in the current state with a non commercial license, which at least should open it up to formula student teams, to get a universal BMS a lot of work would have to be done and I don't want to do this with that hardware. For example there should be eeprom configuration options, most of the definitions that are now in the config.h header, e.g. time outs, voltage thresholds and so on, could be configurable from a GUI. Maybe someone else will start working on the project for next years formula student season. If time allows I will stick to the project and help other formula student teams to adapt the BMS to their car, if there is interest.

However, I will try to do a second version with a different communication approach, more distributed modules for measuring the HV parts, with less hardware safety functions and relying more on software. I never had a single case where the hardware safety circuits did prevent an

overcurrent when closing the AIRs, but if it is the case, it can end really bad for the hardware and also the people around the car.

This new version will use a CAN Bus to communicate between the slaves and the master, this will bring some additional costs to the slaves, but not as much as I thought when choosing daisy chain over can bus. Actually I already did a slave with a microcontroller and CAN, but discarded the idea because of the ESD issues I had. A completely CAN integrated BMS has a lot of advantages, but this new concept won't be part of this document.

All together this project brought me a lot of ups and downs, most of the time it was fun doing all the new things. Only the stressful weeks before the events where really bad for me, as we were struggling to find the failures in the system. We worked 14 hours a day seven days a week for two weeks at least. But no studies could have brought me more knowledge than this project, so in the end is was good for me to do it.

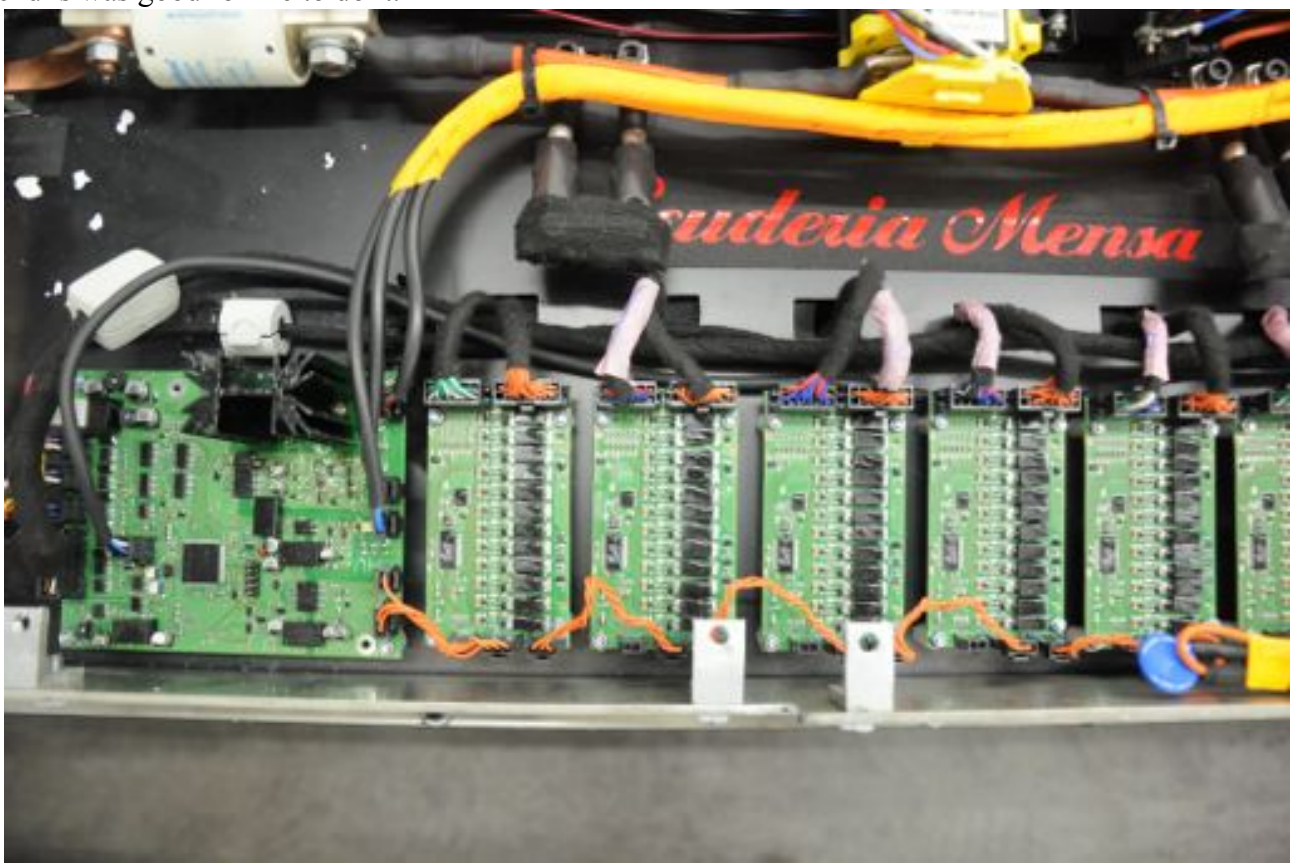


Figure 36: Closer view of the installed BMS

List of Figures

Reference List

Davide Andrea: “Battery Management Systems for Large Lithium-Ion Battery Packs”, Norwood 2010

Linear Technology Corporation: “LTC6803-2/LTC6803-4 Multicell Battery Stack Monitor Datasheet”, Milpitas 2011

Linear Technology Corporation: “DC1653A LTC6803-3 Battery Monitor HARDWARE/SOFTWARE USERS GUIDE”, Milpitas 2011, Handbook for the Demo Board for the LTC6803

Texas Instruments Incorporated: “3 to 6 Series Cell Lithium-Ion Battery Monitor and Secondary Protection IC for Applications, bq76pl536a Datasheet”, Dallas 2012

Texas Instruments Incorporated: “Improving Communications With the bq76PL536 – Application Report”, Dallas 2012

Linear Technology Corporation: “Single, Dual and Quad, 1.8V, 13µA Precision Rail-to-Rail Op Amps, LT6001 Datasheet”, Milpitas 2005

Linear Technology Corporation: “Micropower, Over-The-Top SOT-23, Rail-to-Rail Input and Output Op Amp, LT1782 Datasheet”, Milpitas 1999

Linear Technology Corporation: “µPower, 16-Bit, 250ksps 1- and 2-Channel ADCs in MSOP, LTC1864 Datasheet”, Milpitas 2007

Linear Technology Corporation: “SOT-23, 44V, Over-the-Top, Micropower, Precision Rail-to-Rail Comparator, LT1716 Datasheet”, Milpitas 2002

Atmel Corporation: “8-bit Microcontroller with 32K/64K/128K Bytes of ISP Flash and CAN Controller, AT90CAN128 Datasheet”, 2008 San Jose

Eberhard Sengpiel: “Berechnung: Spannungsteiler”, http://www.sengpielaudio.com/Rechner-spannungsteiler.htm last visited 20.08.2012

Thomas Krueger: “NE555 Rechner”, http://www.dieelektronikerseite.de/Tools/NE555.htm last visited 15.07.2012

John Hewes: “555 and 556 Timer Circuits”, http://www.kpsec.freeuk.com/555timer.htm last visited 14.07.2012

Elithion Incorporated: “elithion Homepage, comparision charts, documentation”, http://elithion.com/ last visited 28.08.2012

LION E-Mobility AG: “Homepage, Beschreibungen und Dokumentationen”, http://www.lionemobility.de/ last visited 01.07.2012

ICC Nexergy: “CC/CV Charging Curve”,

http://www.iccnexergy.com/technology/vvt/ http://www.iccnexergy.com/wp-content/uploads/2011/03/CCCV-Charging1.jpg last visited 31.08.2012