Battery Management System

- Project focus: Instrumentation -

Project Report

AIE3-3-E22

Aalborg University
Department of Energy

The tool utilized to write the report is the software called Overleaf from Latex.

AALBORG UNIVERSITY
STUDENT REPORT

Department of Energy
Aalborg University
http://www.aau.dk

Title:
Battery Management System

Theme:
Instrumentation

Project Period:
Autumn Semester 2022

Project Group:
AIE3-3-E22

Participant(s):
Justus Anton Pfaff
Hubert Dabrowski
Mateusz Matejko
Nils Albin Emil Nolemo

Supervisor(s):
Daniel Ortiz Arroyo
Dil Muhammad Akbar Hussain

Copies: 1

Page Numbers: 46

Date of Completion:
December 20th, 2022

Abstract:

As the electric vehicle industry is growing, challenges with regard to the batteries for these vehicles are expanding. This spans from sourcing of materials, production, maximizing performance and longevity, to reuse and recycling. In this project the issues of performance and longevity have been looked at from the vantage point of the battery management system. The initiating problem guiding the research was defined as: **Can energy management in Electric Vehicles be improved?** This lead to the development of subsystems which are common in battery management systems; state of health measurement, cell balancing and charging. The subsystems were built physically or simulated, and put through testing to verify their function. Although not all tests performed nominally, they did show what areas may need further research in order to implement the systems in a more efficient manner. A software architecture was built to encompass the different systems, such that further development tying together the subsystems could be achieved.

Contents

Preface

This report was written by third semester students studying Applied Industrial Electronics, BSc at Aalborg university Esbjerg.

Aalborg University, December 20, 2022

[illegible]

Hubert Dabrowski
<hdabro21@student.aau.dk>

[illegible]

Justus Anton Pfaff
<jpfaff21@student.aau.dk>

Emil N

Nils Albin Emil Nolemo
<nnolem21@student.aau.dk>

[illegible]

Mateusz Matejko
<mmatej21@student.aau.dk>

Chapter 1

Introduction

The electric vehicle (EV) industry is one of growing importance in the vehicle and environmental sectors. Between the years of 2019 and 2020, the sale of EVs increased by 40%, up to 3 million per year[1]. This growth may in large part be due to the expanding focus on lowering emissions and environmental impact. The European Environmental Agency highlights this in a report, explaining that a significant amount of greenhouse gas emissions (GHG) will need to be reduced within the transportation sector, and that electric vehicles can be one of the potential tools in doing so[2]. More and more car manufacturers have been entering into this expanding market, most notably Tesla Motors, who lead the sales of plug-in electric vehicles among all car manufacturers in 2021[3]. Innovation is and will become increasingly important in order to further develop electric vehicles, as both economic and environmental pressures grow. Engineering these vehicles to be more efficient with both the energy used to power them and the materials used to build them will be important. With these factors in mind, an initiating problem was formulated in order to frame the further study into this industry and technology: **Can energy management in electric vehicles be improved?**

Chapter 2

Problem Analysis

This chapter aims to give an overview of current development and technologies in the EV and battery sector, what types of batteries are used, and technologies used to interact with these batteries.

2.1 Economics

Although electric vehicles have existed since the early 1800s, it is only in recent decades that the industry has had substantial growth, as gasoline powered vehicles have been by far the most dominant type since the early 1900s[4]. The modern electric vehicle market had its beginnings with the sales of Hybrid electric vehicles (HEVs). The first large scale production of an HEV, the Toyota Prius, is said to have been a pivotal point in setting this industry in motion[4]. Whilst the HEV market grew, the founding of Tesla Motors in 2006 and it subsequent rise has paved the way for mass produced, fully electric vehicles, and more automakers have since then entered the market, with 143 Electric vehicles having been released in 2019 alone [5]. As of 2021 the global expenditure on EVs, both plug-in hybrids and fully electric battery vehicles, reached 120 billions dollars[6]. This is likely to keep growing rapidly in the coming decade, with a prediction from Deloitte expecting around a 29% year on year growth rate up until 2030[7].

As these vehicles are becoming more prevalent, new challenges and innovations have been emerging. The rise of software use in vehicles has become more common, integrating safety features and autonomous driving features into the vehicles. This specialized automotive software sector is expected to grow over 5% per year up until 2026 [8]. This technological development has presented challenges in terms of legislation, with different countries allowing differing levels of autonomy for these vehicles. However, the challenges of testing and integrating autonomous vehicles in a uniform manner are being worked on by the federal state of the United States, as well as the European Union, showing a large government interest in forwarding this technology[9] [10]. Another prominent challenge within the electric vehicle space is that of batteries. The storage of electricity in vehicles is done by several different types of batteries, which will be discussed in further detail in section 2.2. One of the main challenges with batteries is the supply of raw materials for production. A 2020 report by the European Union identified the production of Li-on batteries as a potential supply risk, as the European Union countries only account for around 0.2% of global Lithium Ion battery production [11]. Both short and long term solutions to this issue are currently being worked on at the governmental as well as research level. Relieving some of the scarcity issue can be done both by implementing recycling capabilities of the batteries, reducing the need to obtain more raw materials, as well as reusing and extending the life cycle of already produced batteries. A paper out of the University of Mcgill discusses these issues, pointing out that EV batteries can often be reused for lower power applications [12]. They do however point out that one of the issues with re-use of these batteries is the uncertainty of the state of health of old batteries, which along with the already assuredly degraded performance after use in EVs lowers the economic value, which lowers the incentive to invest in re-use.

A potential for further research into improving diagnostics and performance of new and old batteries could thus be an interesting area to explore. The following chapter will explore the types of batteries currently in use in industry and how these are managed technically.

2.2 Batteries for EVs

Batteries in electric vehicles require both the battery cells themselves to power the motor and peripheral systems, and a system which controls this power. These two parts are commonly referred to as battery cells, and battery management system (BMS). In this section the battery cell is the focus. The battery management system will be the focus of section 2.3.

2.2.1 Construction of a galvanic cell

A standard battery cell, also know as galvanic cell, converts chemical energy into electrical energy. It consists of an anode, a cathode, an electrolyte, a separator and a current collector. The cathode and the anode are the electrodes, where the anode is the negatively charged and the cathode the positively charged electrode. The anode and cathode are connected by a electrolyte. The electrolyte is a substance which does redox-reactions to transport charge between the electrodes. The separator is usually a membrane, which separates the electrolyte in two parts. One part is connected to the anode and the other part is connected to the cathode. The function of the separator is to separate the anode from the cathode to not create a short circuit, additionally it is permeable for ions so the electro-chemical reactions can take place. The current collector is collecting the current from the anode. This basic structure opens up a lot of possible configurations of substances used, like Nickel Metal-Hybrid (NiMH), Lead Acid, Lithium-ion (Li-ion) batteries and many others. Not all of the batteries listed are equally suitable for an electric vehicle.

2.2.2 Type of batteries suited for EVs

Deciding what type of battery to use for an electric vehicle can require many differing levels of analysis. One can look at the economic costs, performance, lifespan, safety aspects, weight, to name a few. In practice a combination of these are used to decide what type to utilized, since there is no perfect technology the main focus will be on the energy density, more specifically on the theoretical capacity, commonly measure in (mAh/g).

Lead Acid

The lead acid battery is a battery technology that that has been on the market for a long time. The advantages of this battery type is that it is a reliable and robust technology. Additionally it can supply high currents due to its low internal impedance. The disadvantages of the technology are that lead is a heavy metal (11.3 kg/dm^3) and the cell voltage reaches 2.0 V fully charged [13].

Nickel Metal-Hybrid

The Nickel Metal-Hybrid can be seen as a further developed nickel cadmium battery, which was forbidden in vehicle applications due to its environmental damages [14]. A advantage of this technology is that it lasts a lot of charging cycles. A big disadvantage is that the battery gets very hot.

Lithium-ion

Lithium-ion batteries are the most popular batteries in electric vehicles do to its high energy density.

Table 2.1: Comparison of properties of metal-ions[15]

Metal	Redox couple	Standard reduction potential (V)	Theoretical capacity (mAh/g)
Lithium	Li^{+}/Li	-3.0	3860
Sodium	Na^{+}/Na	-2.7	1170
Magnesium	Mg^{+2}/Mg	-2.4	1340
Aluminium	Al^{+3}/Al	-1.7	2980

Table 2.1 concludes that lithium batteries have a high energy density compared to others. There are three different lithium batteries: Li-ion, Li-ion polymer and Li-polymer/ Li-metal polymer. These batteries are categorised by two types of negative electrodes, Li-ion and Li-ion polymer have a insertion/conversion negative electrode type. Common insertion and conversion materials are graphite, alloys and oxides. For Li-ion batteries graphite is usually used. The Li-polymer/ Li-metal polymer uses metallic Li as the negative electrode. Li-polymer batteries are batteries, aim for an electrolyte that is in a solid state, this makes the battery more resistant to temperature. Additionally it has the possibility to reach a higher energy density that a Li-ion battery, since it is in a solid state, which means - higher density. This technology is going to play a big roll for EVs in the future, but this technology is not ready for the market. The difference between Li-ion and Li-ion polymer is if found in the electrolyte. The advantages of the Li-ion over the Li-ion polymer batteries are that it has a higher energy density and a longer lifetime, the advantages of an lithium-ion polymer battery are that the charging time is lower and that it is more robust and can be produced in many different form, unlike a Li-ion battery [16]. In conclusion the Li-ion battery is the most suitable for an electric vehicle due to its high energy density and long lifetime.

2.3 Battery management

Given that EVs require higher voltages and currents than can be provided by a singular cell, this is achieved by connecting batteries in series and parallel [17]. When individual cells are connected in a pack with others, because of charging and discharging a battery pack, battery cells may stay at different voltages due to:

- Internal resistance variation. Over time of using a cell, it can degrade, because of mechanical and chemical factors[18]. It can increase internal resistance of the cell, which can drastically change performance of the battery[19].
- Temperature conditions In large battery packs, cells can be distributed over a

significant volume which can cause differences in their temperature. Because of that, there are differences in charging and discharging speeds.

2.3.1 Cell balancing

To gain maximum efficiency all cells should stay at equal voltage. Cell balancing is a method of maintaining all battery cells at equal voltages. It allows for improving the efficiency of a battery pack.

Passive

Passive balancing is a mode of cell balancing in which batteries in series are equalized to that of the one with lowest voltage. This is done by through the ability to connect and disconnect individual batteries in battery pack to a resistive circuit, a bleed resistor, dissipating excess energy in the form of heat, until the batteries are at equal voltages. Controlling such circuits can be done with switches such as MOSFETs[20]. While this can cause some issues with excessive heat, or being slower than other methods, it is a cheaper option to build and maintain due to the simple circuitry and being mainly hardware based[21].

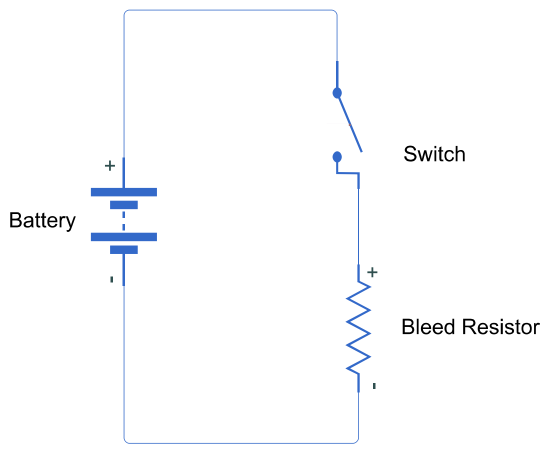

Figure 2.1: An example of passive balancing using a bleed resistor[22]

Active

The active balancing method is a more efficient technique than passive balancing, and comes with some advantages compared to the passive balancing method. When battery cell voltages are not equal the system transfers energy from one cell to another[18]. This can be achieved with additional hardware, but usually also requires some software[21]. Active balancing is more efficient, it saves energy and doesn't produce as much heat as passive balancing, however it is more expensive to create such a system and maintain it due to additional parts and development costs[21].

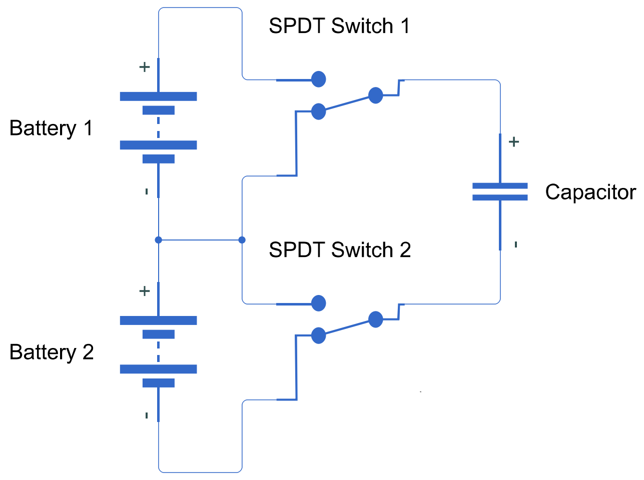

Figure 2.2: An example of active balancing using charge shuttles.[20]

This example of active balancing works by charging up a capacitor using the battery with the higher voltage, and once charged switching the switches to connect to the adjacent battery and charging it.

Lossless

This method uses a matrix switching circuit to add or remove a cell from the battery pack during charging or discharging. It is a recently developed method which includes significant software parts and does not require as complex hardware circuits as in the case of active balancing. However as all batteries are connected with each other in circuits, and there are no instruments storing charge between them, balancing is possible only during charging or discharging. There is a concern if this system would be suitable for electric vehicles. Specification requirements for this system in usage in electric vehicles.

- Allowing only for small deviation of state of charge
- Sensitive sensors measuring low change of value

If the state of charge deviation is very small, probably it is enough to balance batteries only during charging or discharging, though this would require sensitive voltage sensors. This solution has a big potential because of its relatively simply circuit structure, minimizing failures [23].

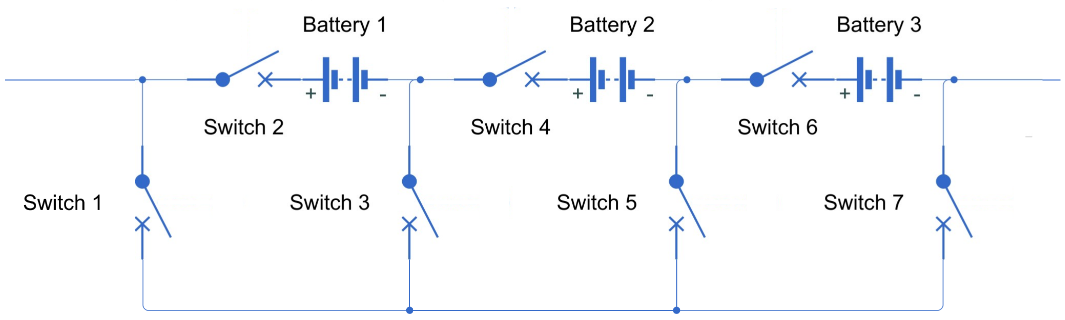

Figure 2.3: An example of a lossless balancing circuit[20].

This utilizes switches to connect directly to other batteries, without the use of a capacitor as in the active balancing example. This system is built in such a way that a battery can charge up any other battery in the series, and not only the adjacent one.

2.3.2 State of Charge

State Of Charge (SOC) determines how much charge is left in a battery compared to its maximum capacity, which is for example useful for defining how long of a range can be driven by an electric vehicle. There are many different mathematical ways of estimating SOC, include Coulomb Counting, Open Circuit Voltage measurement and Kalman Filtering[24]. Any technique chosen will require atleast one of the following measurements to define the state of charge; voltage, current or internal resistance[25].

2.3.3 State of Health

State Of Health (SOH) defines how much of a battery is usable compared to its capacity from when it was newly produced. This is an important aspect of a battery management system as a battery degrades over time[26]. For example, implementing overcharge protection is based on information about the state of health, as the maximum charge a battery can be safely charged up to will go down across its lifetime. The state of health is thus also an important part in what is called the state of safety. The state of safety is a group term of all parameters measured in a battery pack that can be the cause of any damage to the battery, which can include voltage, temperature, humidity and more[27]. State of health can be modelled using many of the same techniques as for state of charge. Deriving the capacity of a battery can for example be done by fully charging and discharging the battery and measuring the expended energy using the Coulomb Counting method, as it integrates the current over time[24].

2.4 Problem definition

Given the likely continued growth of battery use, both in the electric vehicle industry and others, more investment into the technology to support and improve it is likely necessary. Extending the lifespan of each battery is important from an environmental and sustainability perspective. A longer lifespan will in turn mean that a battery can provide more power for a longer duration. This may become more important in future applications, with electric freight truck development underway from several automakers[28], meaning heavier loads and more power requirements. Developing a battery management system with lifespan and safety metrics in mind would be an important step in this direction.

2.5 Problem delimitation

The research done in the problem analysis has highlighted key areas which would be of importance for a BMS that focuses on lifespan and safety aspects. These are:

- Cell balancing of battery packs. This means being able to control the voltage between the batteries during discharge, and being able to control the charging of each individual battery.
- Measure the SOC and SOH of the battery. This means being able to measure quantities such as current, voltage and internal resistance, and using mathematical techniques to extract information regarding SOC and SOH.

These will be implemented for a battery pack of lithium-ion batteries, as they are most common in vehicle applications.

Discussion with Dansk IngeniørService A/S

The problem delimitation is partly based on a discussion between the project group and a project engineer from Dansk IngeniørService A/S, which was at the time of making this project working on a hybrid performance car, and thus had some knowledge and input on current requirements within this field.

Chapter 3

Problem Solution

This chapter is about the implementation of the solution. It is split in different subsystems, which are listed in the next section.

3.1 Hardware overview

The following circuit is showing an overview of the hardware subsystems, which are the voltage regulator, the charging circuit, the voltage sensor and the cell balancing circuit. To charge the batteries a DC voltage source with an output of 10V up to 230V can be used. This source connects the voltage regulators in parallel. The voltage regulator is represented as a block in the following figure. Connecting the voltage regulator to the batteries is a MOSFET, which is triggered by a transistor connected to a pin on the arduino. On the right side of the figure is the cell balancing circuit. The transistors are connected to the arduino to operate the gate of the connected MOSFET. More details about the single systems are in the following sections.

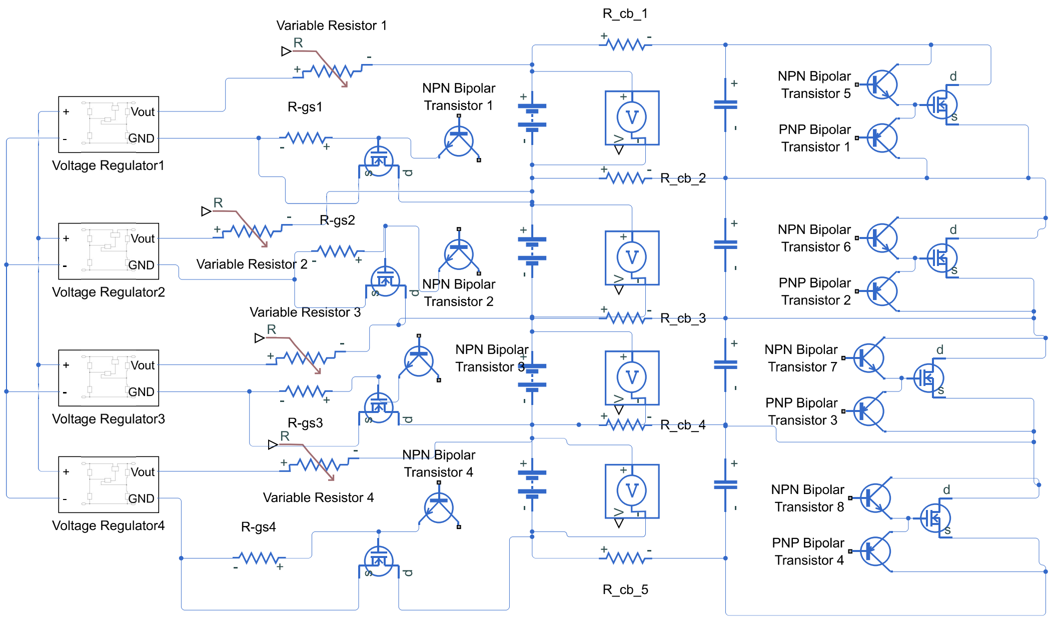

Figure 3.1: The figure shows an overview of the systems circuit.

3.2 Software overview

3.2.1 Architecture

To develop software C++ language has been chosen based on ease of implementation on ATMEGA AVR microprocessor that is used in the project. To implement code, architecture has been developed, to keep the code clean and organized. The Architecture has been based on Clean Architecture and SOLID principles[29]. Although the Clean Architecture and the SOLID principles are mainly used in .NET, so in languages like C#, it was possible to use the same conventions and rules to follow the structure.

SOLID stand for:

- Single responsibility
- Open–closed
- Liskov substitution
- Interface segregation
- Dependency inversion

Single responsibility principle stand for that each class in the program should have one and only one responsibility, so that means each class should have just one job. In our implementation it can be easily spotted, that each sensor has it's own class, so each of these classes is responsible for controlling one type of sensor.

Open-Closed principle stand for that each entity should be open for extension, but closed for modification, so designing an entity like a class it has to be possible to add a method to perform other task, without modifying existing parts of the class.

Liskov substitution principle is definition of that is any class A inherits from another class B, then in the code class B can be replaced with class A. It can be seen that each sensor inherits from "Sensor" class and this can be replaced with the children class.

Interface segregation principle says, that code can not depend on another part of code, which it does not use.

Dependency inversion principle demands that everything should depend on abstractions e.g. interfaces instead of concretions like classes, also higher level classes should not depend on lower level classes.

Another very important aspect is naming convention. During development it is necessary to name classes, variables, etc. To make it easier to understand, one should use precise naming conventions. For this project .NET naming conventions have been used with some possible modifications[30].

- For classes, methods, public properties Pascal Case has been used. For local variables Camel Case has been used.
- For interfaces Pascal Case with preceding capital I has been used.
- For private properties Camel Case with preceding underscore has been used.

3.2.2 Program structure

In the program the highest level object is "CoreService" class, that manages all other services. In the CoreService we have lower level services like ChargingService or BalancingService. Each of the services has one responsibility according to letter S in SOLID. These services use lower level classes. The lowest level classes are made one for each sensor or another task. On top of that the program contains Utility class that contains static methods that help in other classes. The program is based on Dependency Inversion, so higher level classes cannot depend on lower level classes, that's why interfaces are used. Each class has its interface. Interface is an abstract class that contains only names of the methods. If any class inherits an interface it accepts to implement all the methods that the interface contains. Because of that it is known that the method from interface can be used when utilizing the object. If a higher level class has to use a lower level class it refers to its interface and instance of the class is injected by contructor method. It's called contructor injection. All those objects are created in IoCContainer, and can be later used in the whole program.

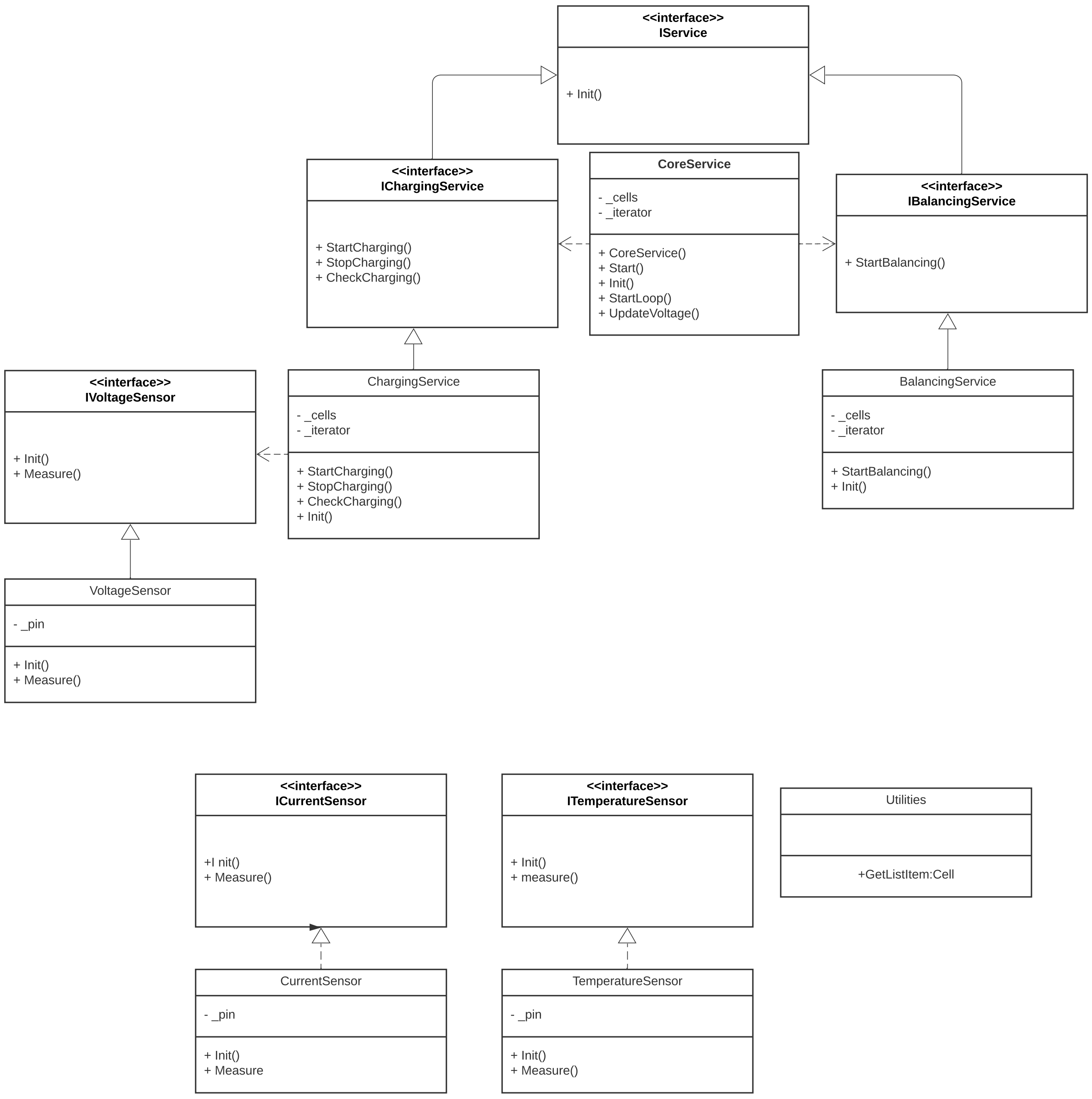

Figure 3.2: A UML diagram of the whole program.

Figure 3.2 is a UML diagram of the whole program, including also parts which were not incorporated physically in the project yet, but are a framework for future

work, such as "TemperatureSensor". In CoreService class we have Start and Init methods that are used in the beginning of the programs lifecycle. The Init method initilizes all required pins and creates a list with cells depending on the number of cells in the system.
Start method is higher level and is called first, it prepares all the pins taking it from static PINS class, that contains definitions of all used pins, and calls Init, which is passing those pins as parameters.

```
void CoreService::Start(int cellNumber)
{
  int chargePins[4] = {PINS::CELL_CHARGE_0, PINS::CELL_CHARGE_1, PINS
    ::CELL_CHARGE_2, PINS::CELL_CHARGE_3};
  int voltagePins[4] = {PINS::VOLTAGE_SENSOR_0, PINS::VOLTAGE_SENSOR_1
    , PINS::VOLTAGE_SENSOR_2, PINS::VOLTAGE_SENSOR_3 };
  int balancePins[4] = {PINS::BALANCE_01, PINS::BALANCE_12, PINS::
    BALANCE_23, PINS::BALANCE_30 };
  Init(cellNumber, chargePins, voltagePins, balancePins);
}
void CoreService::Init(int cellNumber, int chargePins[], int
    voltagePins[], int balancePins[])
{
  for (int i = 0; i < cellNumber; i++)
  {
    Cell cell;
    cell.Init(chargePins[i], voltagePins[i], balancePins[i]);
    _cells.push_back(cell);
  }
}
```

Listing 3.1: "Start" function

```
void Cell::UpdateVoltage()
{
  Voltage = _voltageSensor.Measure();
}

void Cell::Init(int chargePin, int voltagePin, int balancePin)
{
  ChargeMOSFET.InitOutPut(chargePin, false);
  _voltageSensor.Init(voltagePin);
  BalanceMOSFET.InitOutPut(balancePin, false);
}
```

Listing 3.2: "Cell" class

Above class "Cell" contains the function "UpdateVoltage" which copies the analogue value from the pin which corresponds to the voltage sensor. Function "Init" initiates first states of the sensor's output pins.

3.3 Subsystems and components

In order to build the larger circuits defined in section 2.5, smaller subsystems and components were necessary to utilize or build. Below is a description of these, and how they were designed and work.

3.3.1 Current sensor

The current sensors used in this project were built using the physical fact that the current in a circuit is proportional to the voltage drop over a known resistor. This is generally done by utilizing a shunt resistor, a resistor with a small resistance that can withstand higher amounts of current. Measuring the voltage drop over this resistor, the current can be calculated using Ohms law[31]. The reason shunt resistors are usually utilized is to minimize the voltage drop compared to the circuit load of the rest of the system, so as to have minimal power loss in the circuit. This however presents another challenge, the ability to read very small drops in voltage over the known resistance. To do this a non-inverting Operation Amplifier circuit was added in order to more amplify this small voltage in order to more accurately read the voltage. Below is a schematic of the current sensor circuit.

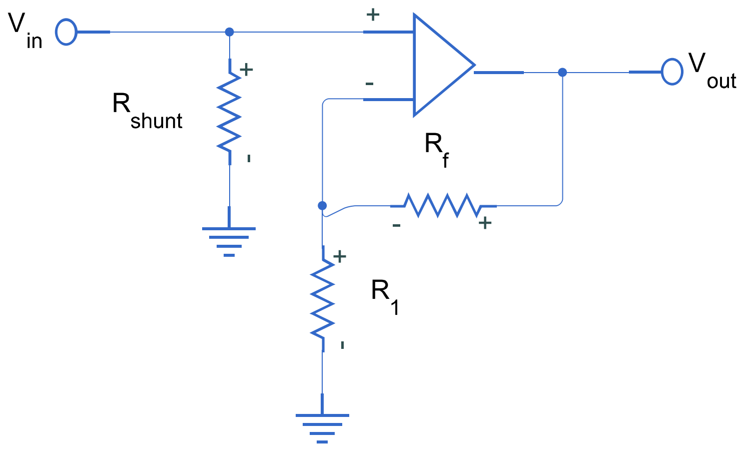

Figure 3.3: A diagram of the current sensor.

Here the voltage between shunt and ground is divided and used as the positive input of the operational amplifier. The gain of the amplifier is regulated by the resistors denoted R_f for feedback resistor, and R_1. To derive a mathematical equation relating the input voltage, R_f and R_1 to the voltage output, characteristics of an ideal operational amplifier are assumed. Assumptions made for modelling an ideal amplifier means assuming the positive and negative input terminals of the amplifier have the same voltage[32]. Applying Kirchhoffs Current Law to the node between R_f and R_1 the following is derived:

$$\frac{0 - V_1}{R_1} - \frac{V_1 - V_{out}}{R_f} = 0. \tag{3.1}$$

Since it is assumed to be ideal, V_1 is equal to V_{in}, and the equation can be rewritten and rearranged as

$$V_{in}(1 + \frac{R_f}{R_1}) = V_{out}. \tag{3.2}$$

With this equation, the required values of the resistors can be calculated for the application. The primary considerations here being the input voltage, related to the current by the shunt resistor, and the maximum voltage that can be read by the micro-controller, in this case an Arduino Uno, which is 5V[33]. Another limiting factor is the way in which the Arduino measures voltage. It utilizes a 10 bit Analog to Digital Converter (ADC), thus outputting a value between 0 and 1023 to represent 0-5V. This means that the resolution of each output step in the measurement is as shown in the equation:

$$\frac{V}{step} = \frac{5000mV}{1024} = 4,883mV \tag{3.3}$$

Depending on how large of a current is expected to be measured, the resistors are set so as to maximally amplify the voltage drop up to but not over the 5V which the Arduinos pin can manage. There is thus a trade off between accuracy and range which must be considered.

3.3.2 Voltage regulator

The voltage regulator is used to have a constant voltage supply to the batteries. It is a component that can get different input voltages, but always has the same output voltage. In this prototype the voltage regulator is needed for the charging of the battery cells. Each cell has one voltage regulator, so for one pack of four cells there are four voltage regulators in parallel. The following equation relates the output voltage to the properties of the components of the voltage regulator [34].

$$\frac{-BV * (R_2 + R_3)}{R_3} = V_{out} \tag{3.4}$$

The equation is multiplying the reverse breakdown voltage of the zener-diode with the voltage divider consisting of the two resistors. The breakdown voltage is negated, because the zener-diode is compared to conventional diodes integrated reversed, so it blocks higher voltages than the reverse breakdown voltage. To get the reverse breakdown voltage we solve the equation for -BV.

$$\frac{V_{out} * R_3}{R_2 + R_3} = -BV \tag{3.5}$$

The output voltage required is 4.2 volts.

$$\frac{4.2V * 470}{1000 + 470} = 1.34V \tag{3.6}$$

This concludes that the required reverse breakdown voltage of the zener-diode is 1.34V. The following figure shows the circuit of the voltage regulator. It consists of three resistors, a NPN bipolar transistor, an op-amp and a zener-diode in n-form, which means that it needs to be connected to a common ground.

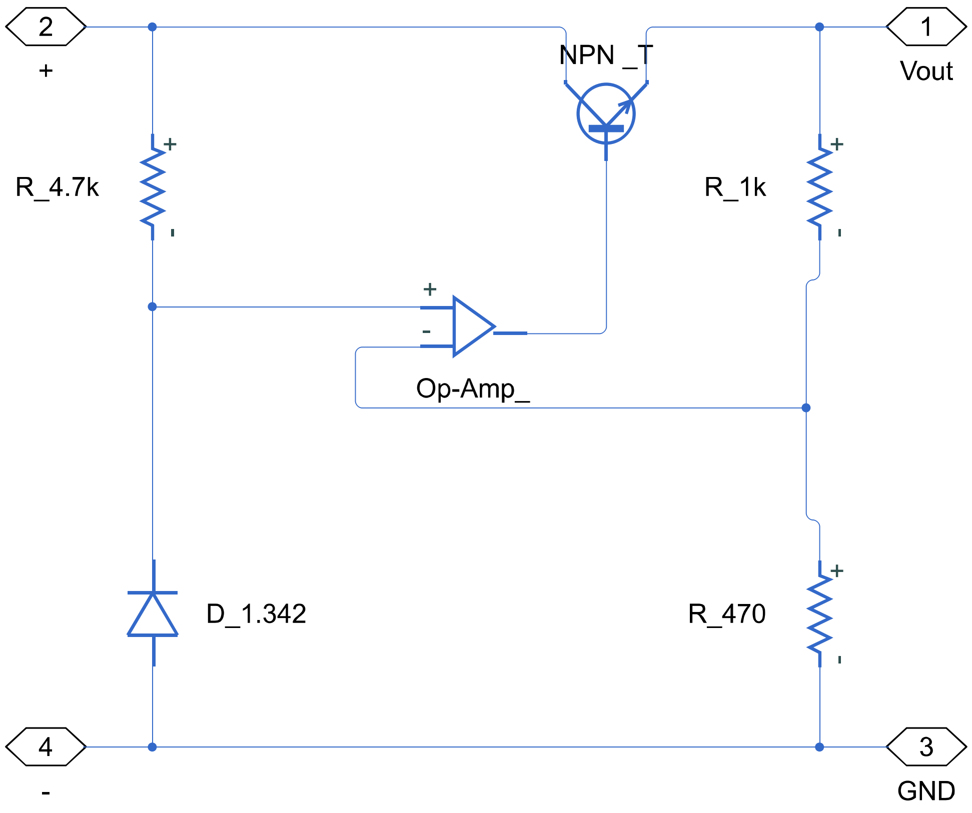

Figure 3.4: The figure shows a voltage regulator, where the terminals 2 and 4 are the voltage supply and 1 and 3 the output, where 3 and 4 are the grounds. V_0ut in this case terminal 1 should always be about 4.2 volts.

The NPN bipolar transistor is used as an amplifier in this circuit to achieve the right voltages for charging the batteries, since it is controlled by the current entering the base pin. It has three pins called collector, emitter and base. The base is the control pin. It is used to control the current flowing from collector to emitter.

3.3.3 Switching MOSFETs

MOSFET is short for Metal Oxide Semiconductor Field Effect Transistors. They are voltage controlled switches, with three pins the drain, gate and source. MOSFETs have two types n-type and p-type. The n-type MOSFETs source pin has to be connected to common ground, and the p-type MOSFETs drain pin has to be connected to common V_{cc}. For all MOSFETs the drain and source connect the circuit. The gate pin is controlled by the applied voltage, dependent on the MOSFET it will open or close the circuit. Every MOSFET has a threshold voltage (V_{th}), to open or close a MOSFET V_{gs} has to be bigger than V_{th}. V_{gs} is the difference in potential between the gate pin and the source pin. In conclusion to switch a MOSFET the following condition has to be fulfilled:

$$V_{gs} \geq V_{th} \tag{3.7}$$

To switch the MOSFETs transistors are used, which act comparable to MOSFETs, with the main difference that transistors are current controlled and MOSFETs voltage. The transistors base pin is controlled by the arduino. If the arduinos output is 5V, there is current flowing from the collector to the emitter pin. The emitter pin is connected to the MOSFET gate pin, which then switches the MOSFET.

Software

For MOSFET control, class "MOSFETcontroller" has been implemented.

```
void MOSFETController::InitOutPut(int pin, bool isDepletionMode =
    false)
{
  pinMode(pin, OUTPUT);
  _isDepletionMode = isDepletionMode;
  _pin = pin;
  if (!_isDepletionMode)
  {
    digitalWrite(_pin, LOW);
  }
  else
  {
    digitalWrite(_pin, HIGH);
  }
}

void MOSFETController::Open()
{
  if (!_isDepletionMode)
  {
    digitalWrite(_pin, LOW);
  }
  else
  {
    digitalWrite(_pin, HIGH);
  }
}

void MOSFETController::Close()
{
  if (!_isDepletionMode)
  {
    digitalWrite(_pin, HIGH);
  }
  else
  {
    digitalWrite(_pin, LOW);
  }
}
```

Listing 3.3: "MOSFETController" class

- "InitOutPut" named method initializes an output pin on a microcontroller and sets it to either a high or low value based on the "isDepletionMode" parameter. The method also sets the value of a class variable called "_pin" to the pin number specified in the parameter.
- Method named "Open" sets the output pin specified in the "_pin" variable to a high or low value, depending on the value of the "_isDepletionMode" variable.
- Method "Close" sets the output pin specified in the "_pin" variable to a low or high value, depending on the value of the "_isDepletionMode" variable. This is the opposite of the "Open" method.

3.4 Battery indicators

Below is a description of the main functions which comprise the battery management system.

3.4.1 State of Health

Measuring the state of health in a battery is an important part of many battery management systems. Data on the current state of the battery can be useful to know how much capacity it can currently provide, but over time data can also be gathered to track the rate of degradation of the battery, meaning a model of its behaviour over time can be made. The parameters which were decided on for measurement of the state of health were the following:

- Battery Capacity

- Open Circuit Voltage
- Internal Resistance

Measuring open circuit voltage and internal resistance was done in the lab using a multimeter. Measuring the batteries capacity was done using the technique of Coulomb Counting. This works by measuring and integrating the current drawn from the batteries over time. The measurements were done using an Arduino, while the data processing was done on a Raspberry Pi. Below is a flowchart describing the software which handled this operation.

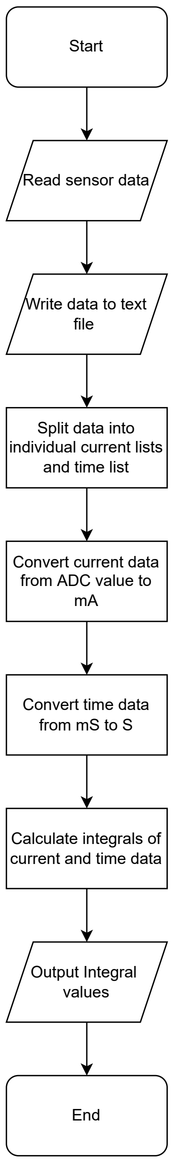
Start
Read sensor data
Write data to text file
Split data into individual current lists and time list
Convert current data from ADC value to mA
Convert time data from mS to S
Calculate integrals of current and time data
Output Integral values
End

Figure 3.5: A flowchart diagram of the current measurement.

This algorithm does three important things, namely saves the data provided by the Arduino, splits and converts said data, and integrates the current as is given by the Coulomb Counting method.

Since an Arduino is used to manage the sensors, and a Raspberry Pi for managing the data, a way of communicating between the two micro-controllers was needed. This was done using serial communication with Universal Asynchronous

Reception and Transmission (UART) protocol. The data was sent between the devices via USB cable. Utilizing the protocol was done using a library called Pyserial. The main code used from this library in managing the communication can be seen below:

```python
        #Set up serial communication for the Raspberry Pi
        if __name__ == '__main__':
            ser = serial.Serial('/dev/ttyACM0', 9600, timeout = 1)
            ser.reset_input_buffer()

        #Read data which is sent from the Arduino
        def get_raw_serial_data():
        serial_data_string = ''
        if ser.in_waiting > 0:
            read_line = ser.readline().decode('ascii').rstrip()
            serial_data_string += read_line
        return serial_data_string
```

Listing 3.4: Python example

In setting up the serial communication, the first thing that is done is using the built in python variable*__name__*, which checks what module of the program is currently being executed. Since *__main__* is the function in python which is always run first, this piece of code will be run at the beginning. It then sets up an object *ser*, where the USB port, baud rate and timeout rate are given. Baud rate is a term denoting the rate of change between each signal sent[35], and timeout denoting the time the program will await data to be received before proceeding[36]. A method named *reset_input_buffer()* is called to clear any data already sent and waiting to be read which may be incomplete, as the Arduino continuously outputs data. In the function *get_raw_serial_data()* data is read into the string *serial_data_string* when there is serial data available, and this is done until the *readline()* method detects a new line command, or until the serial timeout limit, which was set to 1 second in the initialization. The data received is appended to a .txt file for storing all data the data from the current session. This stored data is then split and processed giving information on current and capacity use. This data was collected using the current sensors designed in subsection 3.3.1.

Coulomb Counting makes use of integrating current over time, which gives the expended energy. For this the Scipy library function *cumulative_trapezoid()* was used, Scipy being a scientific computing library built for Python. The method uses the Trapezoidal rule for approximating the integral[37]. This is a mathematical technique used for integrating numerically. It functions by approximating the area underneath the curve as a trapezoid. The mathematical equation describing this is[38]:

$$\int_a^b f(x)\,dx = \sum_{n=1}^{M} \frac{f(_{x-1}) + f(_x)}{2} \Delta x_n \qquad (3.8)$$

In this equation the approximation between each point is linear, as f_{x-1} and f_x are added together. This can lead to small errors building up over time. The size of the error will also depend on how small Δx_n is for each summation. One way to mitigate such cumulative errors is to utilize the open circuit voltage of the battery as a calibration tool for the capacity algorithm[39]. This can bring its own set of challenges, such as highly variable voltage readings which result particularly from high current draw. During current draw the open circuit voltage can instead be estimated with the internal resistance of the battery[40]. Although test measurements of the open circuit voltage and internal resistance were measured on the four batteries used for the project, open circuit voltage and internal resistance were not implemented in this project for modelling the SOH over time, due to time restrictions.

3.5 Charging system

3.5.1 Hardware circuit

An important part of a battery management system is the ability to control the charging of the batteries. Control here meaning the ability to both connect and disconnect each individual battery to a supply, and controlling the amount of current being supplied to each battery. The purpose of having these two types of control over the charging are twofold. The ability to disconnect the battery is important for the purpose of not overcharging a battery, as this can damage batteries and potentially cause thermal runway reactions[41]. Controlling the magnitude of current is important for a batteries State of Health over time, as charging speed can affect the degradation of the battery[42]. Given the stated requirements, a circuit was developed with the capacity to control these.

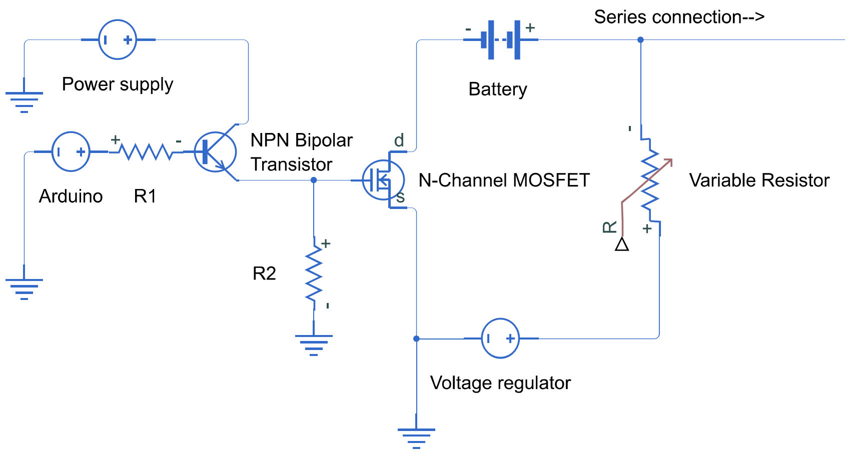

Figure 3.6: An overview of the battery charging circuit.

This diagram is an overview of how the circuit looks for one battery. The circuit is then connected in series to other batteries and their charge circuits. The charging of the battery is done using the voltage regulator developed for this project. The first part of controlling the charging; the ability to disconnect the battery entirely from the regulator, is done using several components. A N-channel MOSFET is used as a switch in order to create a short or open circuit to the voltage regulator. In order to drive the MOSFET it was decided to add additional components, as experiments in the laboratory with several MOSFETS showed that driving the gate strictly with voltages from an Arduino resulted in poor control of the gate, irrespective of the MOSFET being logic level or not. In the simulation an NPN Bipolar transistor is used as a gate driver, which itself is driven at its base using a voltage supply, which in a physical implementation would be the input from a micro-controller, here schematically represented as a voltage supply. The collector of the transistor is connected to a power supply, which ultimately is what provides the voltage at the MOSFETS gate when the transistor is conducting. For the secondary control part, regulating the charging rate of the battery, a variable resistor is connected in series between the regulator and battery. Being able to control the resistance here means that the current to the battery can be regulated. In a physical implementation this variable resistor would ideally be controlled using a micro controller, meaning it could be implemented as a software feature.

3.5.2 Simulation

The charging circuit was implemented as a simulation using Simulink. In implementing the simulation some changes were made with regards to the components compared to the schematic. In modelling the battery behaviour a resistor was put in place with a resistance of 100mΩ, to fit closely to the internal resistance measured of the physical batteries tested in section 4.3. The variable resistor was exchanged for a regular resistor, as values can be changed at will during each test cycle of the simulation.

3.5.3 Software

The main functions operating the charging system are put in the "ChargingService" class.

```cpp

void ChargingService::StartCharging()
{
  for (_iterator = _cells.begin(); _iterator != _cells.end(); ++
    _iterator)
  {
    Cell cell = *_iterator;

    if (cell.Voltage < 4.2)
    {
      cell.ChargeMOSFET.Open();
    }

    Serial.println(cell.Voltage);
  }
}
```

Listing 3.5: "StartCharging" function

Above method named "StartCharging" is a member of the "ChargingService" class. The function iterates through a list of "cells" and checks the voltage of each cell. If the voltage of a cell is less than 4.2V, the function closes a switch.

```cpp

void ChargingService::StopCharging()
{
  for (_iterator = _cells.begin(); _iterator != _cells.end(); ++
    _iterator)
  {
    Cell cell = *_iterator;

    cell.ChargeMOSFET.Close();

    Serial.println(cell.Voltage);
  }
}
```

Listing 3.6: "ChargingService:StopCharging" method

The above method named "StopCharging" is a member of the "ChargingService" class. The function iterates through a list of "cells" and closes the "ChargeMOSFET" for a cell which is called.

Function "CheckCharging" closes charging for cells which are fully charged.

```
void ChargingService::CheckCharging()
{
  for (_iterator = _cells.begin(); _iterator != _cells.end(); ++
    _iterator)
  {
    Cell cell = *_iterator;

    if (cell.Voltage >= 4.2)
    {
      cell.ChargeMOSFET.Close();
    }

    Serial.println(cell.Voltage);
  }
}
```

Listing 3.7: ChargingService::CheckCharging() method

Above method named "CheckCharging" is a member of a class called "ChargingService". The function iterates through a list of "cells" and checks the voltage of each cell. If the voltage of a cell is greater than or equal to 4.2V, the function closes a switch for that cell.

3.6 Cell balancing

As cell balancing method passive balancing was utilized.

3.6.1 Hardware circuit

The figure below is showing the circuit for two batteries. The prototype has four batteries and every battery has s similar discharging circuit for balancing.

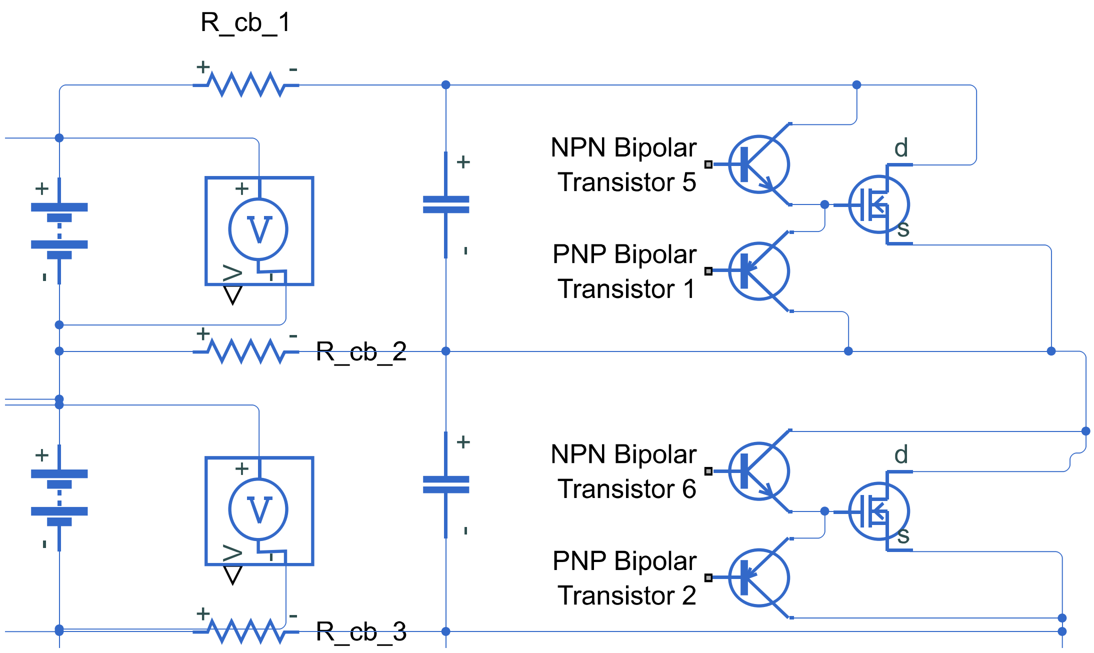

Figure 3.7: The circuit above shows two batteries in series and cell balancing circuit for these two cells.

The method of balancing used in this circuit is passive balancing, since one battery is not charged by another, but the batteries are discharged individually over resistors, in case of them not having the same SOC. The MOSFET on the right side of the figure is connected to the arduino. If the arduino sends a high signal the MOSFET conducts and the according battery is discharged. To have a more stable voltage a capacitor is used. The MOSFET gate is operated with two transistors, one is for charging up the gate and the second one to discharge it. Since the MOSFET is floating, which means the source is not connected to a common ground.

3.6.2 Software

Main methods used for cell balancing are in "BalancingService" class.

```
void BalancingService::StartBalancing()
{
  for (int i = 0; i < _cells.size() - 1; ++i)
  {
    if (Utilities::GetListItem(_cells, i).Voltage > Utilities::
   GetListItem(_cells, i + 1).Voltage)
    {
      Utilities::GetListItem(_cells, i).BalanceMOSFET.Open();
    }

    if (Utilities::GetListItem(_cells, i).Voltage < Utilities::GetListItem(_cells, i + 1).Voltage)
    {
      Utilities::GetListItem(_cells, i).BalanceMOSFET.Open();
    }

    if (abs(Utilities::GetListItem(_cells, i).Voltage - Utilities::GetListItem(_cells, i + 1).Voltage) < 0.02)
    {
      Utilities::GetListItem(_cells, i).BalanceMOSFET.Close();
    }
  }

  for (_iterator = _cells.begin(); _iterator != _cells.end(); ++_iterator)
  {
    Cell cell = *_iterator;


    Serial.println(cell.Voltage);
  }
}
```

Listing 3.8: BalancingService::CheckCharging() method

Above method named "StartBalancing" is a member of a class called "BalancingService". The function iterates through a list of "cells" and performs different actions based on the voltage of each cell. If the voltage of a particular cell is greater than the voltage of the next cell in the list, the method opens the MOSFET. If the voltage of a cell is less than the next cell, the function also opens the MOSFET. If the difference in voltage between the two cells is less than 0.02, the method closes the switch.

Chapter 4

Testing

Testing is important for validating the function of each part of the system. For this reason several tests were formulated and performed in order to gauge the function and performance of individual subsystem.

4.1 Current sensor

Hypothesis The current sensor should be able to measure current corresponding closely to the theoretical value calculated beforehand, and compared with a laboratory multi meter.

Method The test was split into two stages, to test different parts of the sensor. The two tests were as follows:

- Validate the Op-Amp amplification.
- Test measurement of current supplied from a power supply.

The current sensor was fitted with a TI-LM358P operational amplifier, two 5 watt tolerant resistors, $R_{load} = 4,7\Omega, R_{shunt} = 0,5\Omega$, and two resistors for the amplification, $R_f = 10k\Omega, R_1 = 1k\Omega$. This gives a theoretical amplification value of

$$V_{in}(1 + \frac{R_f}{R_1}) = 11V_{in} \tag{4.1}$$

Testing the amplification was done by supplying a certain voltage from a power supply, and using a multimeter to measure the input and output voltage of the Op-Amp.

In the second test, the current sensor was supplied by a power supply and connected to the Arduino to measure the voltage drop over R_{shunt}, which was converted to current in the code.

Results The first test was done with the goal of validating the theoretical amplification of the Op-Amp circuit used in the current sensor. The importance of this being any value over or under this will affect the readings done by the Arduino. The theoretical calculation yielded an amplification of eleven times the input voltage, while the experiment yielded a measurement that was 11,120 times the input. A constant (OP_AMP_MULTIPLIER = 11.120) was thus implemented in the code to compensate for this slight deviation. In the second test the output of the amplifier circuit was connected to the Arduino to test the reading of the current. The expected current at the 4V power supply output was around 0,77A. A measurement was taken with a multimeter which showed a current of 0,720A, while the Arduino measured a current of 0,666A. This measurement was done after having implemented the Op-Amp constant, thus another constant was implemented in the code named OP_AMP_CORRECTION_FACTOR in order to correct this, which in this case was

$$\frac{0,720}{0,666} = 1,081 \tag{4.2}$$

4.2 Voltage regulator

Hypothesis The voltage regulator should have a constant output voltage of 4.2V, independent of the input voltage.

Method For testing the hypothesis a simulation software is used called Simulink. Where the voltage regulator was modeled according to the description in chapter 3 and figure 3.4. It was tested starting at a supply DC voltage of 10V going up to 230V.

Results The Results are shown in the graph below. For 10V the output was 4.199V and and for a input voltage of 230V it was 4.243V. The increase was linear and dependent on the input voltage.

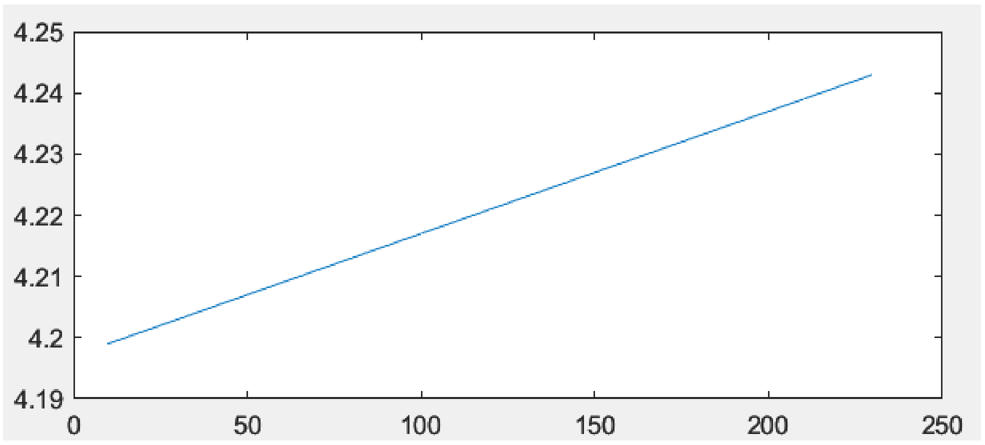

Figure 4.1: The x-axis of the plot is showing the input voltage and the y-axis is showing the output voltage.

The dependency of the input voltage and the output voltage can be described as following:

If

$$10V \leq V_{in} \leq 230V \tag{4.3}$$

Then

$$V_{out} = 0.0002 * V_{in} + 4.197 \tag{4.4}$$

In conclusion the voltage regulator is working as intended, even though it is for 230V about 50mV off, which is tolerable. The usage of a voltage supply of 20V seems ideal.

4.3 State of Health computation

Hypothesis To measure the capacity of the battery, its state of health, using the Coulomb Counting method required performing a full discharge of each battery while measuring current and integrating it. The expected outcome of this is that the final measured capacity is close to that of the stated battery capacity of 3400mAh, though slightly under as the batteries are not brand new, and some degradation is expected to have occurred over time.

Method The test was done using the four fully charged batteries, discharging two at a time. The open circuit voltage and internal resistance of each battery was measured before the experiment was commenced. This data can be used to create more sophisticated models of the batteries SOH, but this was not done in this project, as discussed in section 3.3.1. The measurements showed the following:

Table 4.1

	B2	B3	B4	B5
Open Circuit Voltage(V)	4,0425	4,0773	4,0626	4,0645
Internal Resistance(mΩ)	98,0	97,2	86,7	114

The batteries were connected to the current sensor circuit, and measurements were taken until the batteries were practically fully discharged, outputting less than 10% of the current output at full charge. This data was put through the Coulomb Counting algorithm giving the total capacity when the batteries had been fully discharged. Measurements were taken every 2 seconds by the Arduino.

Results The results from this experiment showed that the capacity for each battery was the following:

Table 4.2

	B2	B3	B4	B5
Capacity(mAh)	3000	3200	3203	3230

4.4 Charging circuit

Hypothesis Given the use of an NPN transistor controlling the gate of an N-type MOSFET, it is expected that the MOSFET will allow current to flow in the charging loop when sufficient voltage is supplied to the transistors base and the MOSFETS drain. It is further expected that changing the Gate to Source voltage (V_{gs}) value of the MOSFET will affect its drain to source resistance (R_{DS}), when the value exceeds the 5V gate input voltage, altering the current flow in the charging circuit.

Method The simulation was initially set up as in section 3.4 with the settings as shown in the diagram:

The MOSFET was initially set to have the following characteristics:

- $V_{gs} = 5V$
- $V_{th} = 1.7V$

- $R_{DS} = 0.025\Omega$

For the test the V_{gs} was further varied to 2V, 3V, 4V, 6V, 7V, 8V, 9V and 10V. A current sensor and scope were placed in series with the charging loop.

Result The current passing in the charging loop under these conditions were the following:

Table 4.3

V_{gs}(V)	2	3	4	5	6	7	8	9	10
Current(A)	3,772	3,772	3,684	3,639	3,592	3,545	3,496	3,445	3,393

The expected current in the charging loop is

$$I = \frac{V_{regulator}}{R_{battery} + R_{var} + R_{DS}} = \frac{4,2V}{0,1\Omega + 1\Omega + 0,025\Omega} = 3,733A \tag{4.5}$$

The results shown for $V_{gs} \leq 5$ were slightly under this value and not constant, which shows that the R_{DS} is not constant for values within the 5V gate input. For values above this it was expected that the resistance rose, which it did, lowering the current.

4.5 Battery cell balancing

Hypothesis Four batteries with different SOC balance during discharge.

Method For simulating simulink was used as before, but for this simulation the "Specialized Power Electronics" library was used, since it is able to read the sate of charge form the battery immediately. The simulation is for 250s. The batteries used in the simulation are Lithium-Ion batteries, with a nominal voltage of 4.0V and a capacity of 2.6Ah. The following circuit was simulated:

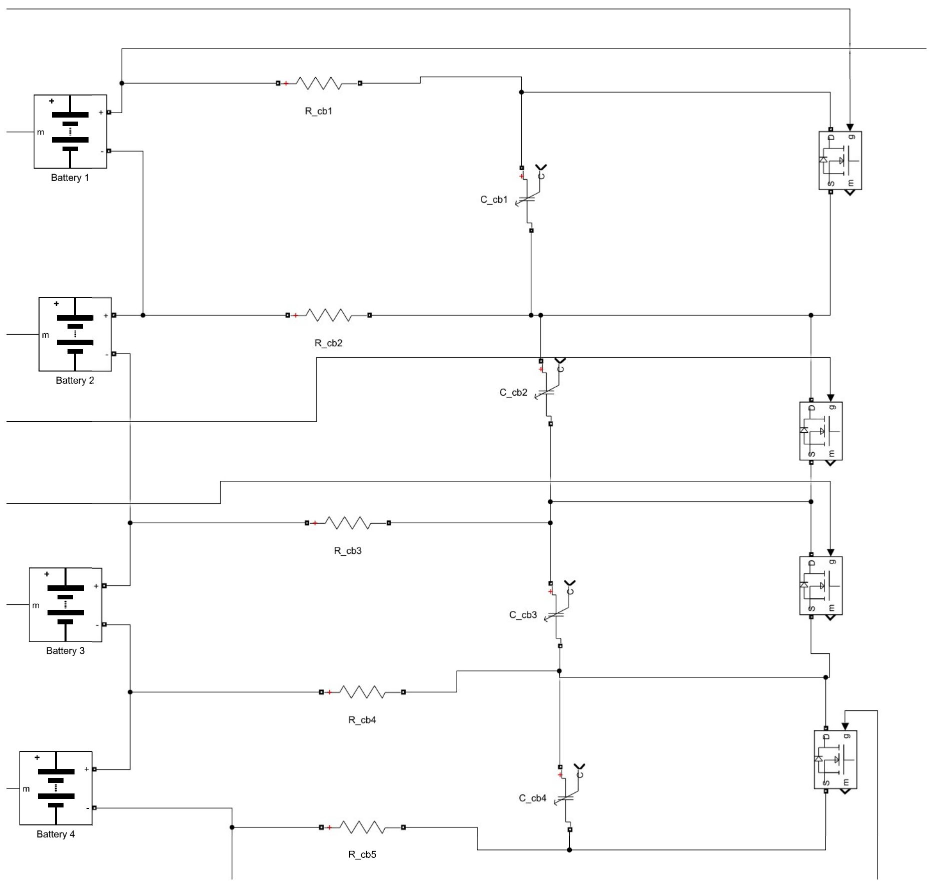

Figure 4.2: The figure shows the simulated circuit. The four batteries have different states of charge. The first one having a SOC of 64, the second one 68, the the third one 65 and the fourth one 67. All R cb resistors have a resistance of 1 Ohm and all capacitors a capacitance of 0.000001 Farad.

The Resistor on the right side of the figure is the load resistor it has a value of 2Ohm. The result that is aimed for is that, the batteries slowly discharge and during the discharging process the SOC of the different cells become equal to another.

Results The testing of the circuit was a success, since the SOC of the four batteries is approximately equal after 215s.

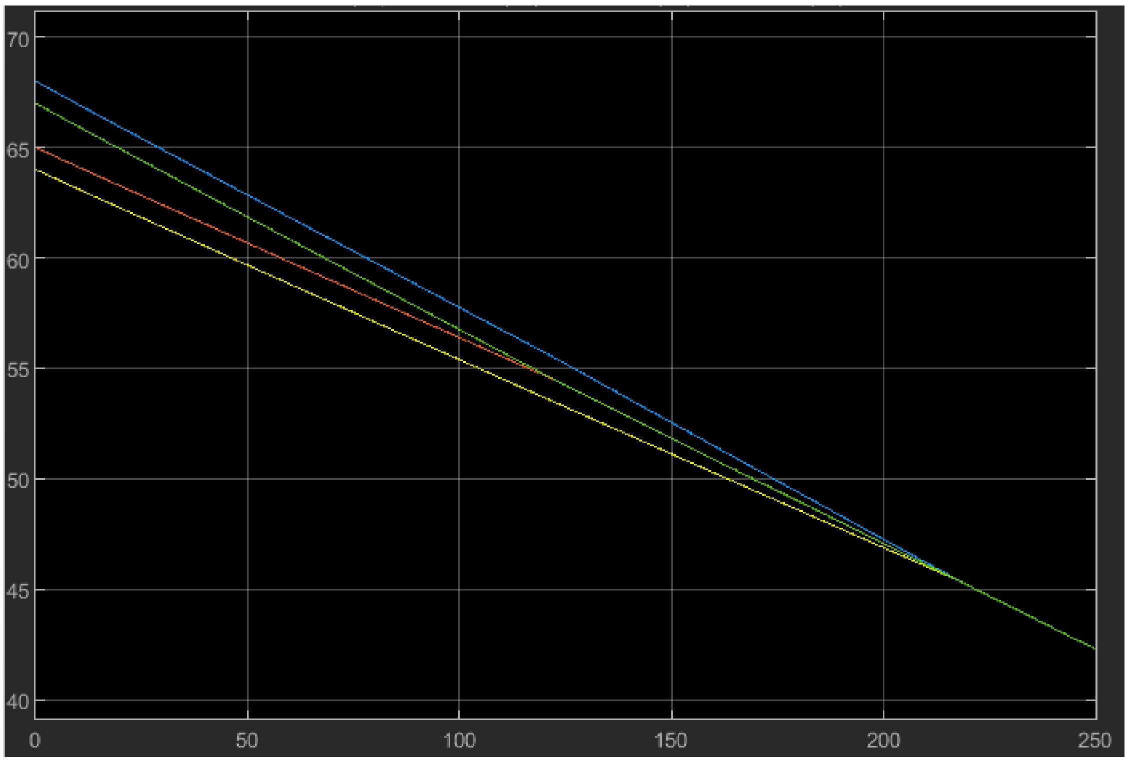

Figure 4.3: The figure shows the SOC of the different batteries. The x-axis is representing time and the y-axis is representing the state of charge. The blue line represents the second battery, the green line the fourth, the red line the third and the yellow line the first.

Chapter 5

Discussion

This section describes issues encountered with the developed models, aswell as other models of cell balancing systems which the group attempted to build.

5.1 Switching MOSFETs

The switching of MOSFETs was a big difficulty since there is a lot to take into account. First of all it was difficult to find the right MOSFET for the right application, since there are p-type and n-type MOSFETs, which can be depletion and enhancement MOSFETs. Depletion type MOSFETs are the MOSFETs used for switching, which is perpose of them in this project. A n-type MOSFET is needed in case the source is connected to common ground and a p-type MOSFET is needed in case there is a common voltage supply. After the necessary research the group made the MOSFETs work in simulation, by making sure that the gate to source voltage is above the threshold voltage and either connecting a resistor from gate to source or a transistor. A resistor only works when the source of the MOSFET is grounded.

5.2 Switching circuit

Another model of a balancing circuit developed by the group was a switching circuit, based on lossless balancing technique. It disconnects one particular cell from being connected in series. The principle to balance during charge and discharge by either:

- disconnecting the battery with the lowest SOC, so just the other batteries are discharged
- disconnecting the battery with the highest SOC during charge, so just the other batteries are charged

The above mentioned principles of the model can be implemented by controlling MOSFETs and can lead to SOC equalisation along all cells connected in series. The electronic circuit of the model is presented in figure 5.1.

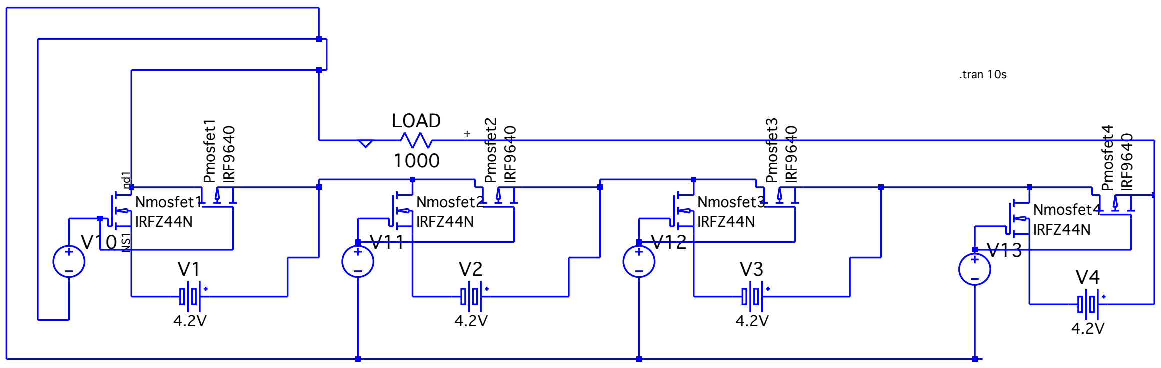

Figure 5.1: Schematics of a switching circuit used in the simulation. Voltage sources: V10; V11; V12; V13 imitate a signal going to a gate from a micro-controller. Resistance LOAD[Ohm] imitates a motor

5.2.1 Results

Voltage measurements have been conducted based on a simulation created in LT-Spice.

Disconnecting all of batteries

Table 5.1: All batteries are disconnected

	V1	V2	V3	V4
Voltage(V)	0	0	0	0

By applying the above values we receive a potential of 0.462V between the terminals of the load resistor.

Connecting 1 battery

Table 5.2: Input to MOSFETS gates

	V1	V2	V3	V4
Voltage(V)	5	0	0	0

By applying the above values, we receive a potential of 4.19V between the load resistor terminals, which corresponds to one battery connected.

Connecting all batteries

Table 5.3: Input to MOSFETS' gates

	V1	V2	V3	V4
Voltage(V)	5	5	5	5

By applying the above values we receive a potential of 5.33V between the terminals of the load resistor.

5.2.2 Problems

The simulation in LTspice software has shown that potential voltage between gate and source of a particular MOSFET is not equal to the signal applied to a gate and it resulted in a difficulty of predicting the value, as it was also depending on a voltage sources connected to other MOSFETs gates. Probably there were 2 main sources of the problem:

- Source pin of MOSFETs could not be grounded in this application.
- Connecting and disconnecting a battery from a series caused a variable voltage potential between the source and drain of particular MOSFETS.

5.2.3 Possible solutions

There might be included MOSFET driver to control gate to source voltage depending on a current state of other MOSFETS. Temporary simulated solution included variable input to MOSFET gates.

Connecting all of batteries

Table 5.4: Input to MOSFETS' gates

	V1	V2	V3	V4
Voltage(V)	5	10	15	18

By applying the above values we receive a potential of 16.79V between the terminals of the load resistor, what corresponds to 4 batteries in a system.

5.2.4 Conclusion

Simulation has shown that by applying a variable signal to the MOSFETS gates depending on the current state of the circuit - state of charge, a connection of particular batteries, it is possible to make the system work. There is a potential for an improvement in the future.

5.3 Discussion with a company about BMS

During testing of the switching balancing circuit, the group contacted and spoke with a company called Dansk IngeniørService A/S, which was at the time working on a hybrid performance car, and have worked on their own BMS. Their model was based on passive balancing because of reliability and the circuit complexity in case of active balancing which could cause other problems like limited capacity for regenerative breaking energy. They did however analyse our lossless solution, and the group received feedback about regarding issues of the floating MOSFET gate voltages. They proposed that isolated gate drivers may be a solution to this issue.

Chapter 6

Conclusion

During the development of the project important milestones have been accomplished, and further avenues for development have shown. The capacity to measure internal resistance, open circuit voltage and current and use that information to establish a batteries SOH is an aspect of a battery system which can be further developed to build more advanced models of a batteries state of health over time. Having constructed a current sensor for this purpose also gives more access to tuning the system to a specific application. Furthermore, the work done in the lab to develop a controllable charging circuit highlighted difficulties in working with different types of MOSFETS, leading to the implementation of using transistors to drive the gates in the final simulation. Lastly, the work on cell balancing has shown that an important issue to tackle when working with series circuits in these cases is that of grounding the MOSFET gates properly, a key issue in having worked on the lossless balancing circuit. The initiating problem for the project was:

Can energy management in electric vehicles be improved?

Although the initiating problem was not answered in full because of hinders with development and time, the work done in this project has laid important groundwork and identified key issues upon which further development can take place, both in extending simulation work and moving towards physical implementation of these simulations. Thus work on implementing this specifically for vehicles could be done in future works.

Bibliography

[1] Leonardo Paoli, Amrita Dasgupta, and Sarah McBain. *Electric Vehicles*. URL: https://www.iea.org/reports/electric-vehicles.

[2] European Environment Agency. *Electric Vehicles in Europe, EEA Report No 20/2016*. Tech. rep. European Environment Agency, 2016. URL: https://www.eea.europa.eu/publications/electric-vehicles-in-europe.

[3] Statista.com. *Estimated plug-in electric vehicle sales worldwide in 2021, by automaker*. URL: https://www.statista.com/statistics/977407/global-sales-of-plugin-electric-vehicles-by-brand/.

[4] Rebecca Matulka. *The History of the Electric Car*. Sept. 2014. URL: https://www.energy.gov/articles/history-electric-car.

[5] Thomas Gersdorf et al. *McKinsey Electric Vehicle Index: Europe cushions a global plunge in EV sales*. July 2020. URL: https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/mckinsey-electric-vehicle-index-europe-cushions-a-global-plunge-in-ev-sales.

[6] IEA. *Global EV Outlook 2021*. Tech. rep. International Energy Agency, Apr. 2021. URL: https://www.iea.org/reports/global-ev-outlook-2021.

[7] Dr. Bryn Walton, Dr. Jamie Hamilton, and Geneviève Alberts. *Electric vehicles: Setting a course for 2030*. July 2020. URL: https://www2.deloitte.com/uk/en/insights/focus/future-of-mobility/electric-vehicle-trends-2030.html.

[8] *AUTOMOTIVE SOFTWARE MARKET - GROWTH, TRENDS, COVID-19 IMPACT, AND FORECAST (2022 - 2027)*. URL: https://www.mordorintelligence.com/industry-reports/global-automotive-software-marketw.

[9] Ben Husch and Anne Teigen. *Regulating Autonomous Vehicles*. Apr. 2017. URL: https://www.ncsl.org/research/transportation/regulating-autonomous-vehicles.aspx.

[10] European Parliament. *Self-driving cars in the EU: from science fiction to reality*. Jan. 2019. URL: `https://www.europarl.europa.eu/news/en/headlines/economy/20190110STO23102/self-driving-cars-in-the-eu-from-science-fiction-to-reality`.

[11] S Bobba et al. *Critical Raw Materials for Strategic Technologies and Sectors in the EU, A Foresight Study*. Tech. rep. 2020. URL: `https://ec.europa.eu/docsroom/documents/42882`.

[12] Alexandre Beaudet et al. *Key Challenges and Opportunities for Recycling Electric Vehicle Battery Materials*. Tech. rep. July 2020. URL: `https://www.mdpi.com/2071-1050/12/14/5837`.

[13] Helena Berg. *Batteries for Electric Vehicles Materials and Electrochemistry*. Cambridge University Press, 2015, pp. 48–51.

[14] Helena Berg. *Batteries for Electric Vehicles Materials and Electrochemistry*. 2015, pp. 52–55.

[15] Helena Berg. *Batteries for Electric Vehicles Materials and Electrochemistry*. 2015, pp. 70–71.

[16] Helena Berg. *Batteries for Electric Vehicles Materials and Electrochemistry*. 2015, pp. 83–86.

[17] evexpert.eu. "Battery Management System". In: *evexpert.eu* (). URL: `https://www.evexpert.eu/eshop1/knowledge-center/bms1`.

[18] ionenergy.co. "How Does Cell Balancing Improve Battery Life". In: *https://www.ionenergy.co/resources/bl balancing-battery-life/* (). URL: `https://www.ionenergy.co/resources/blogs/cell-balancing-battery-life/`.

[19] batteryuniversity.com. *How does Internal Resistance affect Performance?* URL: `https://batteryuniversity.com/article/how-does-internal-resistance-affect-performance`.

[20] Aswinth Raj. *Cell Balancing Techniques and How to Use Them*. 2019. URL: `https://circuitdigest.com/article/cell-balancing-techniques-and-how-to-use-them`.

[21] Anushree Ramanath. *Active and Passive Battery Pack Balancing Methods*. URL: `https://eepower.com/technical-articles/active-and-passive-battery-pack-balancing-methods/#`.

[22] Kevin Scott and Sam Nork. *Passive Battery Cell Balancing*. URL: `https://www.analog.com/en/technical-articles/passive-battery-cell-balancing.html`.

[23] Aswinth Raj. "Cell Balancing Techniques and How to Use Them". In: (). URL: `https://circuitdigest.com/article/cell-balancing-techniques-and-how-to-use-them`.

[24] Kshitija A. Gaikwad and Vijaykumar Bhanuse. "State of Charge Estimation Techniques". In: *International Journal of Engineering Research & Technology (IJERT)* 8.7 (July 2019).

[25] Scienceabc.com. *ScienceABC*. URL: https://www.scienceabc.com/innovation/what-are-the-different-methods-to-estimate-the-state-of-charge-of-batteries.html.

[26] biologic.net. *How to measure SoC and/or SoH with a BioLogic potentiostat / galvanostat or battery cycler*. URL: https://www.biologic.net/topics/battery-states-state-of-charge-soc-state-of-health-soh/.

[27] Sravan Kumar Keerthi. *Battery Management System in Electric Vehicles*. URL: https://www.cyient.com/blog/battery-management-system-in-electric-vehicles.

[28] Mark Vaughn. *Electric Big Rigs Are Coming—and We Drive Four of Them*. May 2021. URL: https://www.autoweek.com/news/green-cars/a36506185/electric-big-rig-semi-trucks/.

[29] Robert C. Martin. *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Pearson Education, 2017.

[30] Microsoft.com. *C# Coding Conventions*. Sept. 2022. URL: https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/coding-style/coding-conventions.

[31] NI (formerly National Instruments). *Current Measurements Guide - How is Current Measured?* Sept. 2022. URL: https://www.ni.com/en-sg/support/documentation/supplemental/21/current-measurements-how-to-guide.html.

[32] Shrikrishna Yawale and Sangita Yawale. *Operational Amplifier Theory and Experiments*. Springer Nature Singapore Pte Ltd, 2022.

[33] Arduino.cc. *analogRead()*. URL: https://www.arduino.cc/reference/en/language/functions/analog-io/analogread/.

[34] Peter J. Vis. *Voltage Regulator using Op Amp and Transistor*. URL: https://www.petervis.com/electronics%20guides/voltage-regulator-using-op-amp-and-transistor/voltage-regulator-using-op-amp-and-transistor.html.

[35] Lou Frenzel. *What's The Difference Between Bit Rate And Baud Rate?* URL: https://www.electronicdesign.com/technologies/communications/article/21802272/whats-the-difference-between-bit-rate-and-baud-rate.

[36] https://github.com/pyserial/pyserial. *pySerial API*. URL: https://pyserial.readthedocs.io/en/latest/pyserial_api.html.

[37] The SciPy community. *scipy.integrate.cumulative_trapezoid*. URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.cumulative_trapezoid.html.

[38] Wikipedia. “Trapezoidal rule”. In: *Wikipedia* (). URL: https://en.wikipedia.org/wiki/Trapezoidal_rule.

[39] Ines Baccouche et al. “Implementation of an Improved Coulomb-Counting Algorithm Based on a Piecewise SOC-OCV Relationship for SOC Estimation of Li-IonBattery”. In: *NTERNATIONAL JOURNAL of RENEWABLE ENERGY RESEARCH* 8.1 (Mar. 2018).

[40] Nigel. *SOC Estimation Techniques*. 2022. URL: https://www.batterydesign.net/soc-estimation-techniques/.

[41] Takahisa Ohsaki et al. “Overcharge reaction of lithium-ion batteries”. In: *Journal of Power Sources* 146.1-2 (2005), pp. 97–100. URL: https://www.sciencedirect.com/science/article/pii/S0378775305005112?casa_token=jzT4HBvn9ZgAAAAA:tcdB9s-LccX7qy9fG1IGhYXthagsSkaC1ycvhq4CeAfP1moGqnLMFWZXwQHr7Z-5lY_OCRA.

[42] Juhyun Song et al. “Pathways towards managing cost and degradation risk of fast charging cells with electrical and thermal controls”. In: *Energy & Environmental Science* 14.12 (2021), pp. 6564–6573. ISSN: 1754-5692. DOI: 10.1039/D1EE02286E.

Appendix A

GitHub link containing code and simulations: https://github.com/TeslaAIE/AIEProject