**Battery Balancing Algorithm and simulator**

**Contents**

1. REQUIREMENTS

2. DESIGN

2.1 The Battery Pack

2.1.1 Batteries

2.2 The Simulator

2.3 The driver program

3. IMPLEMENTATION

3.1 Batteries

3.1.1 The discharge curve

3.2 Battery Pack

3.3 Simulation

3.4 Command ProcessIP

3.5 Driver program

4. USAGE

4.1 Building

4.2 Running

4.2.1 Commands and Keywords

1. **REQUIREMENTS**

The requirement is to design a battery balancing algorithm for a battery pack for the discharge cycle and implement a simulator for testing it.

The simulator would provide an API for

1. Setting the initial value of the cell voltages

2. Setting the values of R1, R2, R3 and R\_Load

3. Getting the voltages of the system

4. Getting current being sourced by each cell

5. Getting the state of all switches

6. The implementation is thoroughly documented using Doxygen

7. Implemented with a Makefile to compile the program

8. The code is runnable on a Linux machine

2. **DESIGN**

The system has a simulator program and a battery pack present in the system. Also there is a driver function to run the simulator. So the system is divided into three parts:

1. The Battery Pack

2. The Simulator

3. The driver program

2.1 The Battery Pack

The battery pack is the main component of the system. A battery pack consists 3 different batteries and equal number of switches. A battery pack can be connected with a load and then it will run the circuit and update individual battries and switches status.

2.1.1 Batteries

The batteries is the fundamental element of a battery pack. A battery consists of a voltage source and an internal series resistance. The battery has its own characteristics like discharge curve, capacity and voltage.

2.2 The Simulator

The simulator is a component that binds the battery pack with load and provides API to start and stop simulation of the battery.

Start API resembles connecting the battery with a load making a closed circuit. Stop internally pauses and resets the battery.

2.3 The driver program

The driver program creates the environment to run the simulator. This program implements a command line interpreter that will take user input and perform necessary action. This program creates the cells and battery. connects them to a simulator, and passes user commands to simulator.

3. **IMPLEMENTATION**

The implementation of the above design consists of four classes, namely, Singlebatt, Battery, Simulation and ProcessIP. Singlebatt and Battery are the two most important components that actually implement the balancing algorithm and battery pack. The simulation and the processIP are used to demonstrate the user entered data.

3.1 Batteries

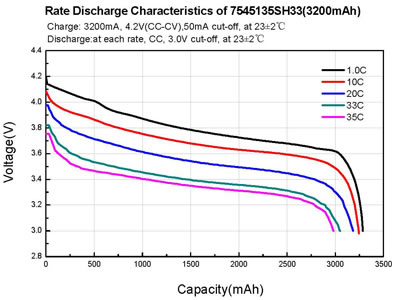
The battery is a voltage source with an internal series resistance. The voltage source has its characteristics discharge curve and capacity.

The battery provides method to set initial voltage, series resistance, capacity and other characteristics parameter which is predefined for this application. It also provides APIs to get sourced current and battery voltage at any time. The battery can update itself when serving a particular current for a specific duration.

The initial voltage, capacity and the discharge curve together defines the batteries characteristics. Capacity of the battery is measured in mAh and it signifies how long the cell can provide power. The discharge curve shows the relation between the output voltage of the cell with respect to its current capacity. The discharge curve is elaborated below.

3.1.1 The discharge curve

The discharge curve is the voltage vs capacity curve that determines the output voltage of the battery for its current capacity. A typical lithium-ion battery discharge curve looks like the following.



For simplicity of calculation, the implementation considers a two step linear approximation of the discharge curve.

This discharge curve incorporates four new quantities, namely, shift and drop. Shift and drop are the gradient shifting point on the capacity axis. These quantities are expressed as percentage of capacity discharged for convenient (one is top position and another for lower position). With these quantities, the gradients can easily be calculated and the discharge curve can be approximated using two linear equations.

Let the three gradients is m1 and then

m1 = ((initial voltage\*shift) - (initial voltage\*drop)) / ((capacity\*shift) - (Capacity\*drop))

This equation is used to calculate the voltage of the cell at any point.

3.2 Battery

The Battery resembles a battery pack with 3 batteries. This implementation assumes three batteries are present in a battery pack. Other than the batteries, the battery pack has switches for each battery to connect or disconnect it. The battery provides a output voltage and when connected to a load also the output current.

The design assumes that the output voltage is the voltage of the connected battery that has minimum battery voltage. It also assumes that the internal series resistance are negligible compared to the connected load.

The source current of the connected cells are calculated by dividing the output current in a ratio that is directly proportional to the potential difference and inversely proportional to the internal series resistance.

The battery pack provides APIs to set battery voltages, series resistance, load resistance and get switch states, output voltage and current, and run, stop and reset the battery.

The battery actually implements the balancing algorithm by operating the switches when the battery is connected to a load and running, i.e. closed circuit.

3.3 Simulation

The Simulator connects the Battery pack and the load. And it provides various APIs to operate the battery. It starts, stops and reset the simulation.

These operations are actually wrapper to the battery APIs. This give the user more option and flexibility to test the battery.

3.4 Command ProcessIP

The command ProcessIP is a very important part of the application as it takes user input and gives the command to execute. The command ProcessIP takes a valid set of commands and sub commands. It provides a command line prompt for user input.

When user gives any command, the parser matches them with the valid command set and returns the function number if there is a match. According to the function number appropriate actions can be taken. This is done by the Driver program.

3.5 Driver program

The driver program prepares the environment for testing the simulator. It creates cells, battery, and simulator, connects them and provides a command line interface for the user via the command ProcessIP to process user input.

The driver program is responsible for taking user input, processing it, taking appropriate actions and formatting the output to present to the user back.

4. **USAGE**

Using this application requires building it in a Linux machine and running. The battery pack and battery classes can also be used in other programs.

4.1 Building

To build the application, Open a terminal in Linux and change directory to the base directory of the application.

Then use 'make' to clean and build the application. It deletes any previous temporary files and binaries present and an executable named 'battbalancesim' will be created.

4.2 Running

To run the application, Open a terminal in Linux and change directory to the base directory of the application.

Then enter the executable name as follows

./battbalancesim

The application will provide with a prompt like Mybatsim>>

4.2.1 Commands and Keywords

The application currently supports 5 commands and 11 keywords. The following list describes them in details.

Commands

get, set, sim, help, exit

Keywords

initvoltage, seriesres, loadres, cvoltage, cutoff, sourcecurr, remaincap, capacity, start, stop, switch

The simulator will start a command line interface and accepts command to view and set various parameters

Generic command format is: MybatSim>> <command> <key> <value1> <value2> <value3>

COMMANDS AND KEYWORDS

set - Sets a value. Format: MybatSim>> <set> <key> <value1> <value2> <value3>

Unnecessary options/arguments are ignored. If required value is not provided, by default it takes 0.

Valid keys are: initvoltage, seriesres, and loadres (only loadres have one argument)

get - Returns a parameter. Format: MybatSim>> <get> <key>

Valid keys are: initvoltage, seriesres, loadres, cvoltage, cutoff, sourcecurr, remaincap, and switch

sim - Starts or stops the simulator. Format: MybatSim>> <sim> <start> / <stop>

help - Prints this help text.

exit - Exits the simulator. If the simulator is still running, tries to stop it first.\n";

DEFAULT VALUES

Initial voltages : 12.5 V, 14.1 V, 12.9 V

Series resistances : 20 Ohm, 30 Ohm, 40 Ohm

Load : 150 Ohm

Capacity : 800 mAH

Shift : 95 %

Drop : 10 %

Cutoff voltage : 8 V