**Directory & File Structure :**

**Project--|**

**|-CSR\_Requirements - Battery\_Balancing\_C++.pdf**

**|-Code**

**|-All\_Data**

**|-data.c**

**|-data.h**

**|-** **Battery\_processing**

**|-** **Battery\_Operation.c**

**|-** **BatteryOperation.h**

**|-** **application.c**

1. **Battery\_Balancing\_C++.pdf** – This document is Gram Power provided assessment document.
2. **application.c** – This .c file contains the main entry point of the code. There are three APIs in the file :

* **Function main:** Here the data is collected from the user, stored into the buffers and further simulation is started.
* **Function app\_processing\_routine** : This API check for 8v min. level and executes the API which simulates battery cells switch states.
* **Function app\_data\_routine** : This API prints the updated data(Voltage ,Current & Switch states) on console for User.

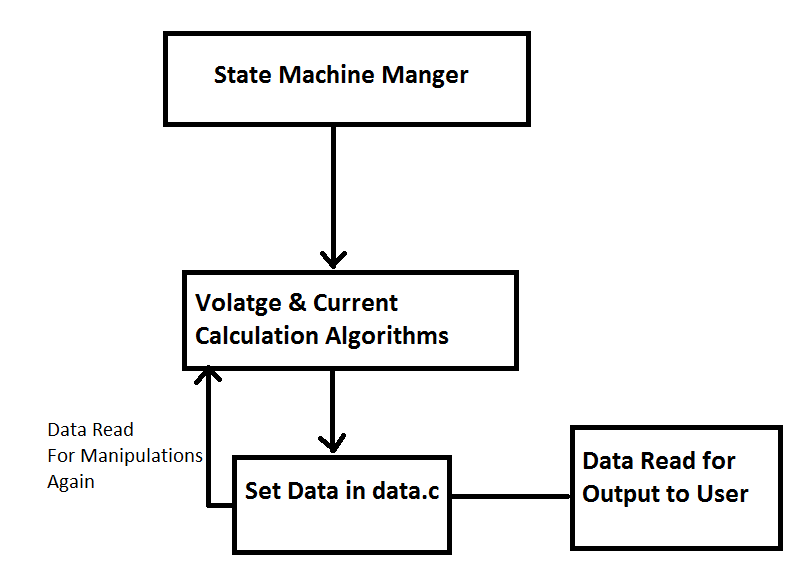
1. **Battery\_Operation.c** – This file contains the state machine for switch state sand calculation of currents & voltages.
   * **Function app\_SwitchStateManager** : This API is the state manager for all the switch states and is supported by other functions of this file.
2. **data.c** – Tis files contains the getter setter methods for all the buffers and variable to keep the data abstracted from user read and write.

**Implementation :**

1. High Level Design :

This big picture of the implementation contains a state machine which always keeps updating the state of the switches. After every 1sec in a state it recalculates the remaining voltages of the cells in order to understand the next probable state of the switches i.e. cell whose voltage is highest will have corresponding switch ON. Also further it calculates the voltage that Load will have in next state, currents for present states.

Data is abstracted in data.c from user and manipulated only through battery\_operation.c.



1. Low Level Design & Algo: