Battery Management System with Individual Cell Controller for a High-Power Battery

By

Shoko Koga

Thesis
Submitted to Flinders University
for the degree of MEE

Shoko Koga

Flinders University
30th October 2023

TABLE OF CONTENTS

TABLE OF CONTENTS I
ABSTRACT III
DECLARATION IV
ACKNOWLEDGEMENTS V
LIST OF FIGURES VI
LIST OF TABLES VII
INTRODUCTION 1
  Background 1
    Requirement of the secondary battery for our lives 1
    Risks of using Lithium-ion batteries 1
  Overview of the existing literature 1
    Battery Management System (BMS) 1
    Protecting Function 1
    Monitoring Function 1
  Scope of the project 2
    Multi-cell connection 2
  Gap between the current and the ideal BMS 2
    The current practical BMS 2
    The required BMS 3
  Problem and Aims for the project 3
    Monitoring individual cells 3
    Isolating each cell 3
  Hypotheses for the project 3
    Continuous running without the replacement of faulty cells 3
    Easy expansion of the battery pack 3
  Experimental methodology 4
    Individual Cell Controller (CC) 4
    Connection and Communication of the CCs 4
  Outline for the thesis 4
LETERATURE REVIEW 5
  Battery Management System (BMS) 5
    OCV model 6
    Coulomb Counting 7
    General BMS for a battery pack 10
    Communication Network – I2C 11
METHODOLOGY 14
  Proposed BMS 14
    Lithium-ion battery 14

Hierarchy structure of proposed BMS 14
Individual Cell Controller 15
Hardware 15
Charging 17
Discharging 18
Software 18
Multi-Cell Connection 22
Cells in Parallel Connection 22
Cells in Series Connection 23
RESULT 24
Result for single cell 24
Charging Mode 24
Discharging Mode 25
Changing and discharging for two cells in parallel 27
Charging Mode 27
DISCUSSION 29
Accuracy of an individual cell controller 29
SoC estimation 29
MOSFET switching 29
CONCLUSIONS 31
REFERENCES 32
APPENDICES 34

ABSTRACT

The more popular the Lithium-ion battery is, the more need for a Battery Management System (BMS) for it is. This is because Lithium-ion batteries cannot manage itself. The general BMS for a single cell has a monitoring function and protection function. In addition, the BMS for cells in series connected has a cell balancing function to prevent over-charging and over-discharging, but the BMS for cells in parallel connected does not because the unbalance in parallel connection is not critical. However, it is not a small problem for high-power batteries. Therefore, the BMS with individual cell controller is proposed to isolate cells with problems. It can monitor and control all cells individually. An individual cell controller has similar functions as general BMS, but it is for a single cell, and it is communicated with a high-level controller through the I2C bus isolating each controller. The Cell Controller (CC) is proposed to consist of a Fuel Gauge IC and three MOSFETs. A Fuel Gauge IC can report cell status based on the Open-Circuit Voltage (OCV) model. Arduino, as the main controller, communicates with this IC and three MOSFETs. As a result, the cell controller can stop charging or discharging once the terminal voltage reaches its threshold. However, the accuracy of State of Charge (SoC) estimation is not enough. Two reasons for this problem can be considered. One is the common GND between Arduino and a CC. The other is the deterioration of cells. In addition, two cells in series or parallel could not be monitored and controlled because the reference point (GND) was common. Therefore, in the next step, each GND for an Arduino and CCs has to be isolated.

DECLARATION

I certify that this thesis:

1. does not incorporate without acknowledgment any material previously submitted for a degree or diploma in any university
2. and the research within will not be submitted for any other future degree or diploma without the permission of Flinders University; and
3. to the best of my knowledge and belief, does not contain any material previously published or written by another person except where due reference is made in the text.

Shoko Koga

30th October 2023

ACKNOWLEDGEMENTS

I want to acknowledge the following for their assistance in producing this report:

- I thank Professor Nasser Asgari, my supervisor, for valuable discussions and advice.
- I am grateful to Mr. Damian Kleiss, Ms. Fiona Cramer, Mr. Crag Dawson, and David Cole for support with the PCB ordering, assembly, and solutions.

LIST OF FIGURES

Figure 1. Comparison of specific energy and power for different EES technologies 6
Figure 2. Method of Voltage Response Test 7
Figure 3. RC Equivalent Circuit for a Lithium-ion Battery........................ 8
Figure 4. General BMS for a battery pack 11
Figure 5. General I2C network 13
Figure 6. General I2C signal 13
Figure 7. Proposed BMS for a battery pack........................ 15
Figure 8. The Diagram for a Cell Controller........................ 16
Figure 9. OCV model for a cell........................ 16
Figure 10. PCB design for a Cell Controller........................ 17
Figure 11. Current flow to charge a cell on a Cell Controller 17
Figure 12. Current flow to discharge a cell on a Cell Controller 18
Figure 13. Control Flow on a Cell Controller........................ 19
Figure 14. Signal of SCL and SDA on I2C line from Arduino 20
Figure 15. Time for one bit related to setting Baud Rate at 9770bps and 38460bps........................ 20
Figure 16. Circuit diagram and I2C network for parallel cell connection 23
Figure 17. Circuit diagram for series cell connection 23
Figure 18. Terminal Voltage and Current in Charge Mode 24
Figure 19. SoC in Charge Mode........................ 25
Figure 20. Terminal Voltage and Current in Discharge Mode........................ 26
Figure 21. SoC in Discharge Mode 26
Figure 22. Terminal Voltage and Current in Charge Mode for two parallel cells 27
Figure 23. The improved prototype cell controller in series connection 30

LIST OF TABLES

Table 1. Decision Matrix of Serial Communication for the BMS.. 12
Table 2. The Detail of NCR18650B.. 14
Table 3. Display of Cell Status reaching Voltage to Charge threshold.. 25
Table 4. Display of Cell Status reaching Voltage to Discharge Threshold 27
Table 5. Display of Cell Status in stopping charging at 3.8V for two parallel cells.......................... 28

INTRODUCTION

Background

Requirement of the secondary battery for our lives

The demand for secondary batteries has grown significantly in recent years due to the increasing popularity of electric vehicles, smart devices and IoT industries across various industries [1]. Especially, Lithium-ion battery is frequently used because it has strengths in high energy density and high efficiency [2].

Risks of using Lithium-ion batteries

The use of Lithium-ion batteries, while prevalent in various applications, raises safety concerns due to inherent risks associated with overcharging and over-discharging. Overcharging can significantly degrade battery performance, ultimately rendering it inoperative. Even more critically, over-discharging poses an even greater risk, potentially leading to catastrophic outcomes, such as thermal runaway, fire, or explosion. Battery Management Systems (BMS) are employed in conjunction with Lithium-ion batteries to mitigate these risks.

Overview of the existing literature

Battery Management System (BMS)

As previously emphasized, the Battery Management System plays an important role in ensuring the safe and efficient utilization of Lithium-ion batteries. The BMS serves several essential functions, including the following:

- Protection from Over-current, Over-temperature, Over-discharge and Over-charge
- Indication in State of Charge (SoC), State of Health (SoH) and State of Power (SoP)

Protecting Function

This function is to prevent over-current, over-temperature, over-discharge and over-charge. Therefore, the system has to look at some parameters continuously during charging and discharging mode. Generally, the system controls the switching device such as a contactor, relay or power FET, to stop charging or discharging.

Monitoring Function

The current, terminal voltage and temperature are measured to monitor the cell status. These parameters cannot show actual cell status during charging and discharging. Therefore, these are converted to State of Charge, State of Health and State of Power to judge the cell status. SoC is the most important parameter in particular [8]. There are some methods to estimate SoC. The simple way is just referring to the cell characteristics after measuring the Open-Circuit Voltage (OCV) of the cell, but this is just for the cell, which is not connected to any charger or load. Thus, the system has to use the other methods during charging or discharging. For example, Coulomb-counting is a popular method. At first, the system captures the open circuit voltage (OCV). Then, once it starts charging or discharging, it counts the flowing current and sums up to estimate consumed or charged charges. The remaining SoC is estimated using this accumulated charge and initial OCV. The other developing method is the internal resistance tracking method, which is employed to BMS IC from Texas Instruments, but it would be out of my project and the detail of it is omitted here.

Scope of the project

Multi-cell connection

The BMS has to be suitable for any batteries, not only single-cell batteries but also multi-connection cells. We can call the battery, which is assembled with some cells, "a battery pack." The battery pack comprises parallel and series cells to generate large amounts of power and energy. Those cells have differences in the SoC, SoH and SoP and it is getting larger in using them because they are affected by different environments with their temperature, charging or discharging current and initial condition. This difference loses the balance among the cells. As this unbalance of the cells, the weakest cell is in over-charge and over-discharge. Therefore, the BMS needs to monitor each cell to keep them balanced. This is called cell balancing and it is an important function for BMSs. The current BMS has a cell balancing function for the series connection cells, but the parallel cell connection does not have this function because it is not critical for them. This means the current BMS cannot manage cells individually. Especially if a large-scale battery pack has a weak or faulty cell, it is difficult to find it, then it remains in the battery pack and makes it a serious problem for the battery pack. Thus, the BMS for the battery packs needs to eliminate this problem by managing all cells individually. [17]

Gap between the current and the ideal BMS

The current practical BMS

As mentioned in the previous section, the current BMS manages only the series connection cells. This consists of some group of cells and they are connected in series. A group is regarded as a single cell and the status of the group is reported to BMS [11]. Thus, there is one weakest cell group, and the system regards that group as the weakest one to prevent over-charging and over-discharging even if all cells are in good condition except one faulty cell. Moreover, the users cannot find a particular cell that has the smallest capacity in the battery and it is difficult to maintain the cell balance when mixing the new cells and old cells. Then, they would replace the whole group of cells with the new one. Some of the cells that are still okay would be wasted.

The required BMS

If the proposed BMS can manage all cells individually, even in parallel connection, all cells would be used until the end of their life. Not only this, it can find the weakest cell and it is possible to add the function to isolate it without stopping the current flow.

Problem and Aims for the project

Monitoring individual cells

The proposed BMS has the same number of cell controllers as the cells in the battery pack. Therefore, there is a lot of data. The system has to estimate each cell's status. In addition, this information should be summarised and displayed on the interface device. Meanwhile, on each cell, the CCs have to charge and discharge their own cell correctly when a charger or a load is connected to the battery pack.

Isolating each cell

When the cell is in an irregular condition, the system has to isolate it from the other cells and continue the safety charging and discharging of the other cells.

Hypotheses for the project

Continuous running without the replacement of faulty cells

When the individual cell controller is used for the high-power system, the battery pack would be running without the replacement of the faulty cell. For example, there are some spare cells in the battery pack and if there are some faults in running cells, those faulty cells are isolated by the cell controller and those spare cells can work instead of them without any physical replacement of cells.

Easy expansion of the battery pack

Generally, the scale of the battery pack is limited by the BMS. Thus, if there is no BMS for a specific battery pack and if the users would like to make the battery pack power or voltage higher, they need to prepare a special BMS for the battery pack. However, the proposed system can expand the battery pack and BMS easily by just adding more cell controllers and middle-level controllers if needed.

Experimental methodology

Individual Cell Controller (CC)

The low-level controller for each cell needs protection and the indicator of cell status to control each cell individually. Therefore, Fuel Gauge IC and power MOSFETs are used. Fuel Gauge IC can estimate the State of Charge in a cell frequently, and three power MOSFETs can switch current flow for isolating, charging or discharging.

Connection and Communication of the CCs

A battery pack has multiple cells in parallel and series connection. Therefore, I2C is used to communicate with multiple CCs, and each CC has to be isolated on the communication bus. Therefore, the I2C isolator has to be included in the CC.

Outline for the thesis

The paper describes the key components and features of the proposed BMS, as well as the role and functionality of the Individual Cell Controller (CC). It also mentions the use of smaller-scale prototypes for evaluation. The methodology section provides details about the methods used, including the scale of prototypes for testing. The results section explains how the Individual Cell Controller operates and indicates the cell status. The discussion section analyses the results and their implications for improving BMS technology. It also identifies potential areas for further research and development. Ultimately, the paper reiterates the significance of the research and its contributions to BMS technology, concluding with a list of references.

LETERATURE REVIEW

Battery Management System (BMS)

Utilization of Lithium-Ion Batteries in Energy Storage

The choice of lithium-ion batteries for energy storage is driven by their exceptional characteristics, including high energy density and long-lasting performance [5]. Figure 1 illustrates the notable advantage of lithium-ion batteries, which can store twice the weight of conventional secondary batteries such as lead-acid and nickel-cadmium (NiCd) batteries. This leap in energy storage capacity has greatly facilitated the miniaturization and weight reduction of battery systems, making them more versatile and adaptable, largely owing to the capabilities of lithium-ion batteries.

However, it's crucial to recognize that lithium-ion batteries cannot be used in the same manner as primary batteries. Their charging and discharging processes require careful control to ensure they operate within their specified capacity. Several critical parameters, including typical capacity, threshold voltage, and typical charging conditions, are pivotal for the efficient management of these batteries.

For instance, let's consider the widely utilized NCR18650B battery, which boasts a typical capacity of 3350mAh, a rated voltage of 3.6V, and adheres to standard charging procedures involving Constant Current Constant Voltage (CC-CV) techniques. This entails a maximum charging voltage of 4.2V and a current rate of 1625mA, as detailed in the manufacturer's datasheet. In this case, the BMS has to stop charging when the terminal voltage reaches 4.2V and discharging when it reaches 2.5V to prevent over-charging and over-discharging.

Meanwhile, the indicator of cell capacity is an important function for the BMS because the user can diagnose the cell health and the remaining capacity. Moreover, battery management extends to encompass three vital estimations. The first, the State of Charge (SOC), is a critical metric used to monitor the remaining charge within the battery. Secondly, the State of Health (SOH) is employed to gauge the level of degradation a battery has experienced over time. Lastly, the State of Power (SOP) indicates the battery's ability to deliver power to a load at a given moment. It's essential to highlight that the accuracy of the system for managing batteries, often referred to as a Battery Management System (BMS), hinges on the precision of SOC estimation. SOC serves as a linchpin in ensuring efficient battery operation and preservation.

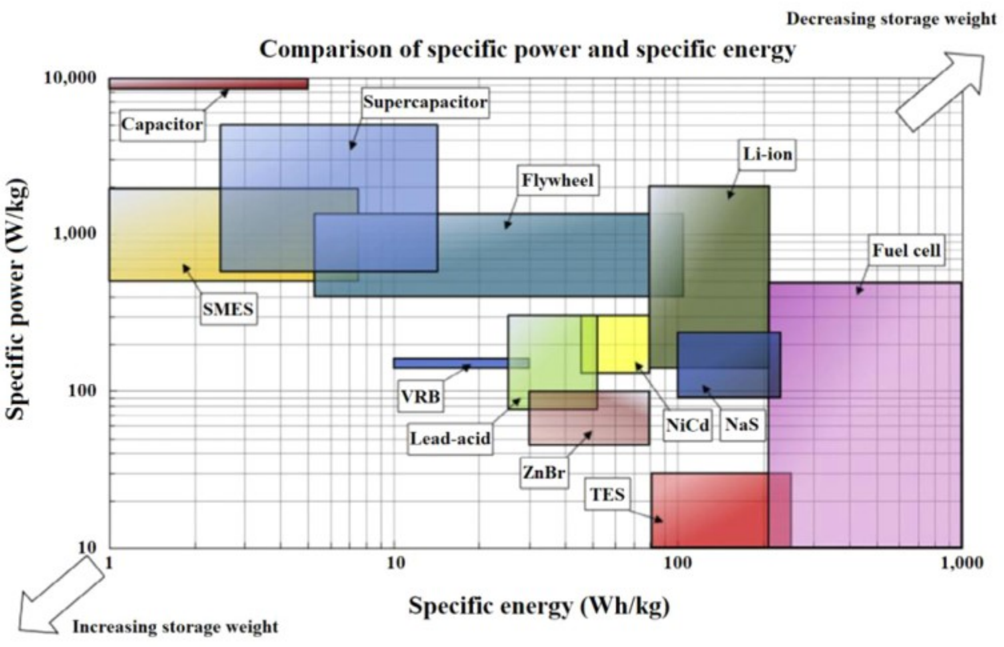


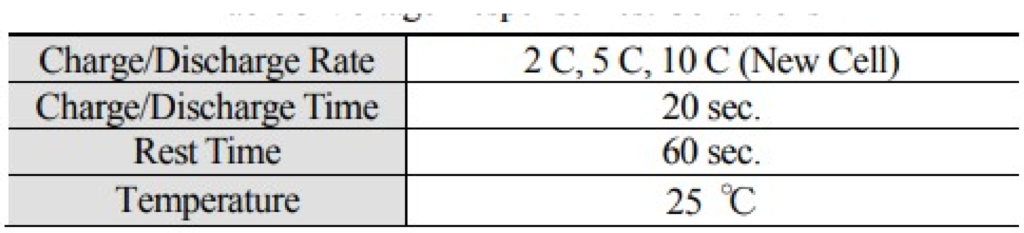
Figure 1. Comparison of specific energy and power for different EES technologies

Acronyms: SMES (superconducting magnetic energy storage), VRB (vanadium redox battery), ZnBr (zinc-bromine battery), NaS (sodium-sulphur), TES (thermal energy storage).

OCV model

The simplest way to estimate the cell's State of Charge (SOC) is the generalized SOC-OCV model. [3] The system with the SOC-OCV model can inform the exact cell status because the Lithium-ion battery has a particular characteristic between the SOC and OCV called the SOC-OCV curve. SOC is an important parameter in maintaining cell condition. The system reports the cell's remaining charge and the other important parameters based on it. Nevertheless, the SOC-OCV curve relies on the cell temperature or its deterioration. Thus, when the system reports the actual SOC, it has to use the SOC estimation model considered with them to get the accurate cell status. As mentioned, the generalized model has been proposed by [3] and would estimate SOC with measured OCV regardless of cell temperature and degradation. Mainly, this model is used for portable devices such as smartphones, cameras, laptops, and so on.

However, the SOC-OCV method is not enough to estimate the SOC continuously and accurately because it is impossible to measure the OCV during charging and discharging. It is made clear that the time constant is 10 sec, and the time for 90% response is about 20 sec with the condition shown in Figure 2 in the response after running and stopping. [17] Thus, only when there is enough rest to make the battery inside stable, OCV can be measured directly. Otherwise, it means that OCV cannot be measured with charging or discharging batteries.



Charge/Discharge Rate	2 C, 5 C, 10 C (New Cell)
Charge/Discharge Time	20 sec.
Rest Time	60 sec.
Temperature	25 ℃

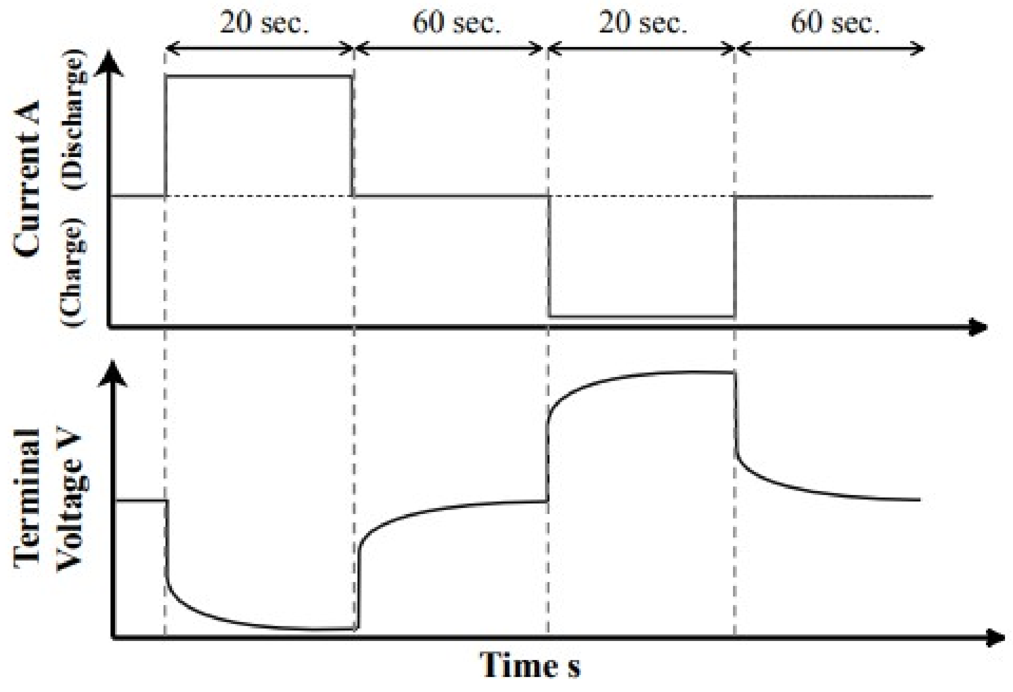


Figure 2. Method of Voltage Response Test

Coulomb Counting

Another method to solve this problem is required to estimate the accurate SOC, which is called the Coulomb Counting method. The cell current is measured constantly during charging or discharging and the measured currents are accumulated with initial OCV, which is measured before starting the charging or discharging process. As a result, the more accurate SOC, which contains the estimated SOC by OCV and the accumulated charge by column counter, is obtained.

Some studies of the estimation of SOC use this basic method, but it is not enough to acquire the exact cell SOC. That is why they explore the ideal cell model and algorithms. Some of them use the RC equivalent circuit shown in Figure 3 for a battery to propose their ideas [8,11,13]. In one of them, the accurate SOC can be estimated by determining the following parameters and battery capacity with the measurement data of an actual battery.

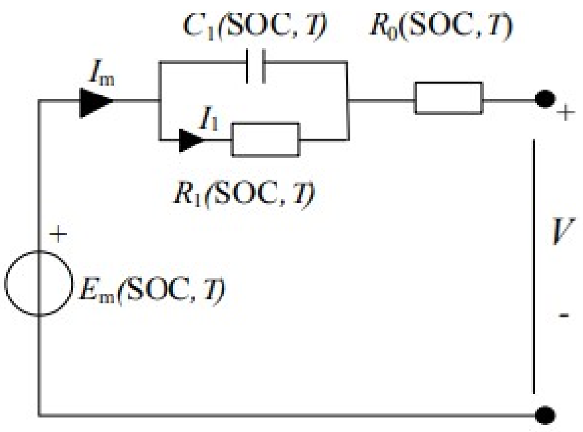


Figure 3. RC Equivalent Circuit for a Lithium-ion Battery

The equation for the SOC estimation is represented by (1).

$$SOC = 1 - \frac{Q_e}{C}, \; Q_e = \int i_m(t)dt \tag{1}$$

Where Q_e is the delivered charge, and C is the battery capacity. Battery capacity C is determined by examining the discharging with constant current and reading the time at the cut-off voltage. This result would be given by the battery datasheet, but it is varied by cell temperature and deterioration. This study uses Simulink to create the model for a particular battery with its measured data. Meanwhile, the determination of the equivalent circuit parameters uses the measurement data with constant current I_m. As a result, C1, R1, and R0 are determined. These parameters represent OCV in E_m, the resistance related to instantaneous voltage in R0, and the resistance and capacitance related to delay in R1 and C1. These parameters function in terms of SOC and temperature. To complete this model, this process needs to imply at some temperature. Once this model is created, this battery SOC can be monitored by measuring the terminal voltage V and the current I_m.

The typical BMS monitors SOC using this model, and if it is out of the safety region of the battery, the battery pack has to be stopped in charging or discharging. This is because the misusage of Lithium-ion batteries will result in the degradation of their performance. In the worst case, the battery will explode when it degrades too much. Degradation in Lithium-ion batteries has three modes. [2] Namely, Loss of Lithium-ion inventory (LLI), Loss of active anode material, and Loss of active cathode material. These are induced by chemical mechanisms such as growth, decomposition, exfoliation, formation, and dissolution of the materials. In the end, these can be diagnosed by their open circuit voltage (OCV) and internal resistance at a specific point. Using this relationship, the method to identify the degradation modes and causes is made sure. Generally, the level of deterioration can be indicated using SOH to make the system simpler. SOH is determined by the internal resistance and the internal resistance is obtained using the previous model with measured temperature and SOC. At the end, the SOH can be obtained using the following equation.

$$SOH = \frac{Current\ battery\ capacity}{Initial\ battery\ capacity}$$

When the battery can be fully charged at initial capacity, its SOH is 100%. When SOH is 50%, the battery can be charged to only half of its initial capacity. Generally, the battery industry defines the end of a lifetime for the lithium-ion battery as 50% of SOH because the speed of the deterioration is accelerated at this point.

One more important estimation is SOP. When a battery delivers or receives power at an arbitrary SOC, the maximum power the battery can do is SOP. SOP uses different equations in discharging and charging, and it is expressed as (2) and (3).

$$SOP_{in} = I_{in}V,\ I_{in} = \frac{V_{max} - OCV}{R} \tag{2}$$

$$SOP_{out} = I_{out}V,\ I_{out} = \frac{OCV - V_{min}}{R} \tag{3}$$

SOP_{in}*: maximum power in charging*

SOP_{out}*: maximum power in discharging*

I_{in}*: maximum current in charging*

I_{out}*:maximum current in charging*

R*: total internal resistance*

From this equation, SOP depends on measured voltage, OCV, and internal resistance because the maximum and minimum voltage is given by the type of battery. All of the estimations are related to OCV or SOC. Therefore, the estimation of SOC or OCV decides the accuracy of the system.

Using these introduced techniques, BMS monitors cell temperature, cell terminal voltage, charging and discharging current and, from these parameters, estimates the state of charge (SOC), the state of health (SOH), and the state of power (SOP), and control power supply in charging and discharging. Basically, BMS reports the battery condition with the cell specification and these estimated parameters, and if it is out of the usable condition, it stops charging and discharging. Additionally, some BMS controls charging conditions instead of chargers.

General BMS for a battery pack

As mentioned in the introduction, Lithium-ion Battery has more advantages than other secondary battery. Thanks to this, the application for the secondary battery has been expanded from portable devices to electric vehicles (EVs). For such large applications, the battery pack contains hundreds of lithium-ion cells. These cells are connected in series and parallel. The series connection is employed to increase the battery voltage. On the other hand, the parallel connection can gain battery power by increasing the battery current. [8,11,13] The cells used for the battery pack are chosen considering cost, productivity, availability, and so on. There are some types of Lithium-ion batteries, but some studies used NCR18650B, which is one of them. For example, using two NCR18650Bs, it is assumed to assemble them in series. It has 3.6V of rated voltage, and then the battery voltage becomes 7.2V. Likewise, when those cells are assembled in parallel, the battery capacity becomes 6700mAh because NCR18650 has 3350mAh of rated capacity. Ideally, the battery pack contains n cells in series or parallel, and the battery pack can supply n times the cell voltage or n times the cell capacity. However, the actual battery pack has a smaller voltage and power than that. Some study clarifies this fact [17]. The causes of the gap between ideal power and actual power are external resistance and individual cell parameter differences. Basically, these cannot be solved completely. The individual cell parameter difference includes the difference in each cell's capacity, internal resistance, and deterioration level and the manufacturing companies control the tolerance of them, and it has some range. Therefore, those cell parameters cannot be made the same completely. When designing the battery pack with such cells, the needs for this product are given as total voltage and total capacity, but it does not mind individual cell parameters. However, the requirement to build the products includes safety constraints and every single component has to follow them and prevent dangerous events. As a result, the weakest cell is dominant and it affects total capacity and, in the end, it is getting smaller than the needs. The required total capacity has to be increased to meet the needs.

This reduction of the total capacity happens no matter which cells in series or in parallel are configured, but the characteristics are different in each connection. In series connected cells, their current is the same, but their voltage fluctuates. If these cells are charged or discharged without any control, the partial cells are getting over-charged or over-discharged.

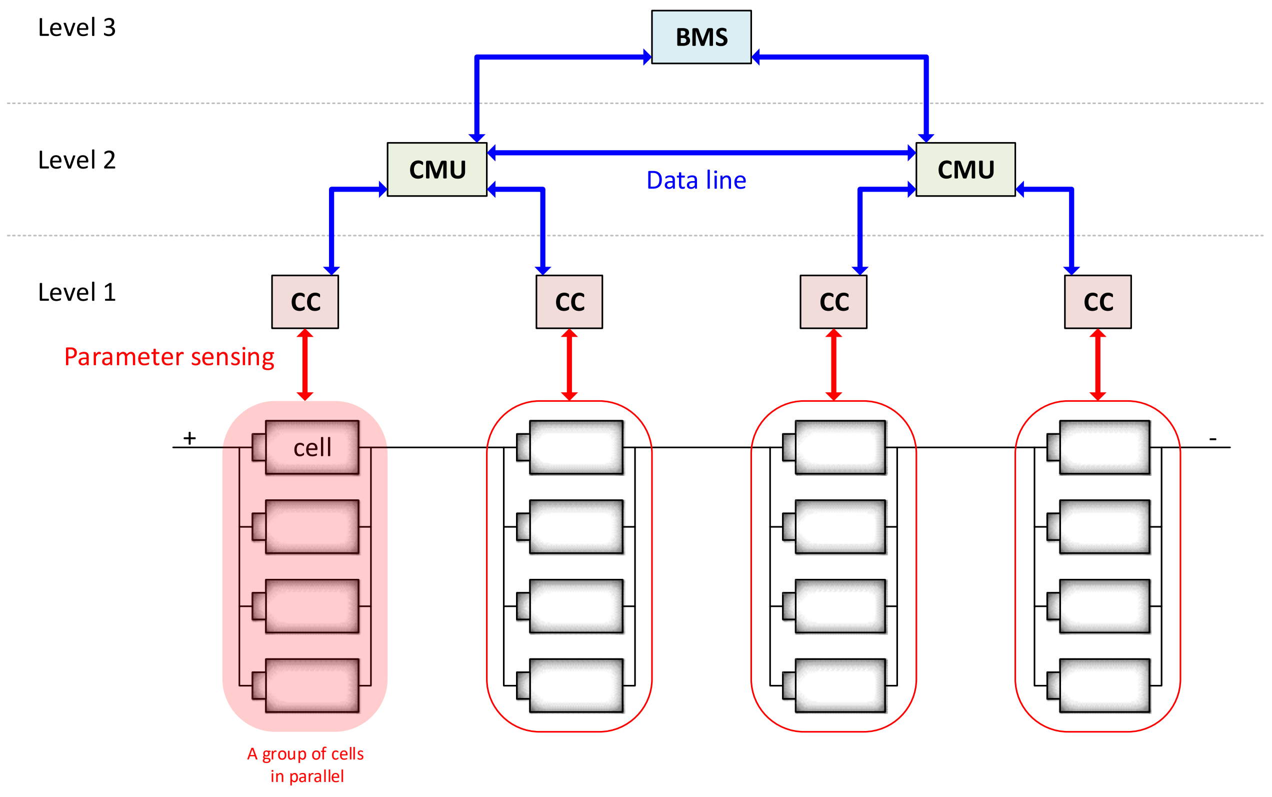


Figure 4. General BMS for a battery pack

Communication Network – I2C

The BMS can use some serial communication methods between a main controller and some other controllers. Namely, SCI, CAN and I2C. Each method has advantages for various applications, such as the following lists, and I2C is better to use individual cell controllers according to the decision matrix shown in Table 1.

- 1-Wire serial communication:

Its concept is for a single master and a single slave. It has a single wire physically and can handle up to 300m distance depending on the master circuit and 2.8 to 6.0V voltage.

- I2C:

It can consist of multiple masters and multiple salves. It has two signal lines for Serial Clock and Serial Data. It can handle up to 5m distance and network voltage from 1.8 to 5.5V.

- SPI:

It can consist of a single master and multiple slaves. It requires four wires physically. Its scale is limited to the circuit board level.

- CAN:

It can consist of multiple masters and multiple slaves and has two wires like I2C, but it has a different network interface, which is a differential open drain/source and requires more electrical components. The network distance is 40m at 1Mbps and 1000m at 50kbps.

Table 1. Decision Matrix of Serial Communication for the BMS

	Weight	1-wire	I2C	SPI	CAN
Multi-device communication	**5**	2	5	2	5
Speed	**2**	1	3	4	4
Network Size	**4**	1	4	2	5
Cost	**3**	4	5	4	1
Complexicity	**4**	1	5	5	1
Total	**90**	32	82	58	60
rank		4	1	3	2

As shown above, there are some types of serial communication methods, such as 1-wire, SPI, CAN and I2C. The purpose of this project is that the system controls individual cells on a battery. It needs to use the communication method to communicate between multiple devices (individual controllers) and one master controller. The master controller observes all cells' conditions via individual controllers and controls the MOSFET to turn a charger and a load on/off. Therefore, the prototype will use I2C, which is advantageous for multiple-device communication.

Figure 5 shows an example of an I2C network. There is one master device and multiple slave devices. Those slave devices have slave addresses individually, and the slave address consists of 7 bits. Generally, the Serial Clock and Serial Data are put to high by a pull-up resistor. One packet of data is composed of 8 bits of data, as shown in Figure 6. General I2C signal. A master device can send a start condition, slave address and a write or read bit first to identify which slave device is communicated with the master. Once a particular slave device gets its address from the main device, the slave device will send low as an acknowledge bit. Otherwise, there is no slave address that a master talks to. The acknowledge bit is to stay high on SDA. After sending a slave address to communicate with a master and receiving an acknowledge bit, the master sends 8 bits of data and continues sending data as long as the whole data is sent. The data can be as long as you need. At the end of sending data, the master sends a stop condition. Meanwhile, if a master wants a slave to send data, the first step is the same, but after the slave gets the slave address, the slave starts sending data in the same way as a master.

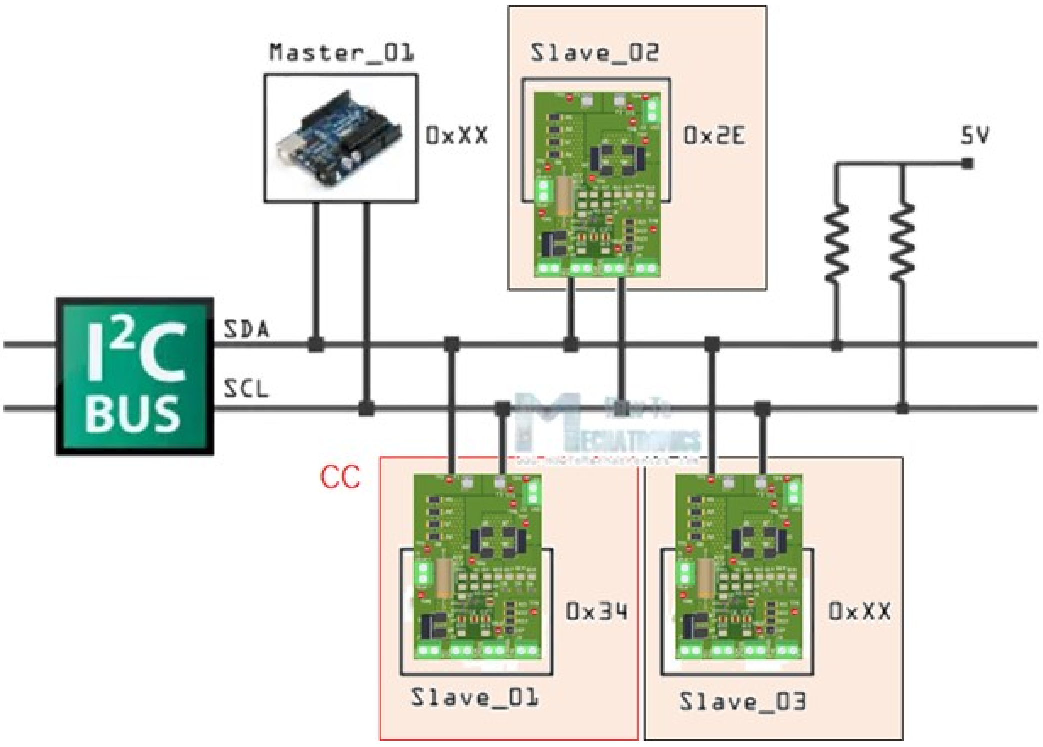


Figure 5. General I2C network

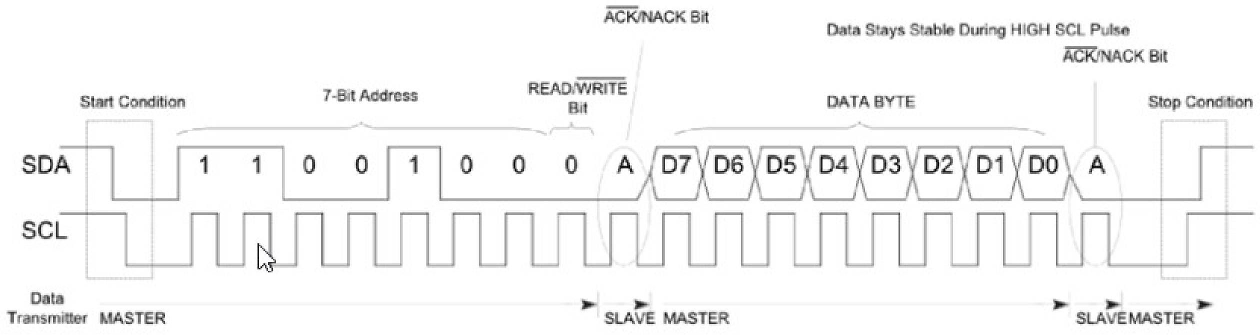


Figure 6. General I2C signal

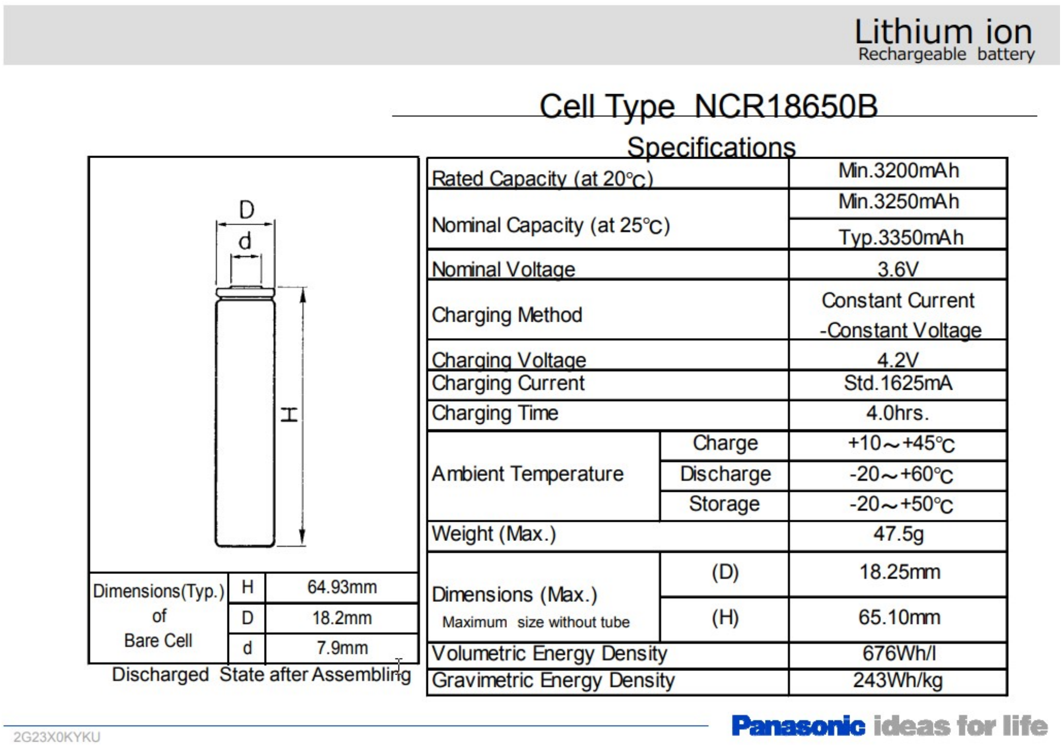
METHODOLOGY

Proposed BMS

Lithium-ion battery

There are various types of Lithium-ion batteries. NCR18650B is used because it is popular and accessible. The part of its detail is shown in Table 2.

Table 2. The Detail of NCR18650B



Lithium ion
Rechargeable battery

Cell Type NCR18650B

Specifications

Dimensions(Typ.) of Bare Cell	H	64.93mm
	D	18.2mm
	d	7.9mm

Discharged State after Assembling

Rated Capacity (at 20℃)		Min.3200mAh
Nominal Capacity (at 25℃)		Min.3250mAh
		Typ.3350mAh
Nominal Voltage		3.6V
Charging Method		Constant Current -Constant Voltage
Charging Voltage		4.2V
Charging Current		Std.1625mA
Charging Time		4.0hrs.
Ambient Temperature	Charge	+10～+45℃
	Discharge	-20～+60℃
	Storage	-20～+50℃
Weight (Max.)		47.5g
Dimensions (Max.) Maximum size without tube	(D)	18.25mm
	(H)	65.10mm
Volumetric Energy Density		676Wh/l
Gravimetric Energy Density		243Wh/kg

2G23X0KYKU

Panasonic ideas for life

According to its datasheet, the preferable charging current is 1625mA. Its charging voltage is 4.2V, and the minimum voltage is 2.5V. Therefore, in this project, the maximum charging voltage of 4.2V, the minimum discharging voltage of 2.8V and the charging current of 0.67A are used considering the safety use.

Hierarchy structure of proposed BMS

A BMS to control the high-power and high-voltage battery safely and efficiently is modeled and refers to the general BMS, which has managed the cells in series connection, and it is improved by managing the individual cells even in parallel connection, as shown in Figure 7. By this, the system can charge and discharge cells following the individual cell's condition, detect a faulty cell and isolate it from others. Hence, the battery can keep working without stopping the system even if some cells fail.

The proposed system has four levels of controllers. CC (Cell Controller), as a low-level controller, has some sensors to measure parameters and estimate the desired parameters such as OCV, SOC, SOH, and SOP and three switches for changing the mode. CMU (Cell Management Unit) and CPU (Central Processing Unit) as middle-level controllers have a microprocessor to gather the data to send it to BMS (Battery Management System). BMS displays the battery pack status and sends a control command to each CC. CC is connected to a cell and CMU via the data line. The parameters of each cell are sent to BMS through CMU and CPU. Then, the BMS judges the total battery status. Additionally, the switches are controlled based on the measured voltage by CC and the thresholds given by the initial setup.

As mentioned earlier, the proposed BMS refers to the current BMS. However, it is too large and too much cost for the study. Thus, the prototype, which has a smaller system, such as for a 3p3s assembled battery, is used in this project.

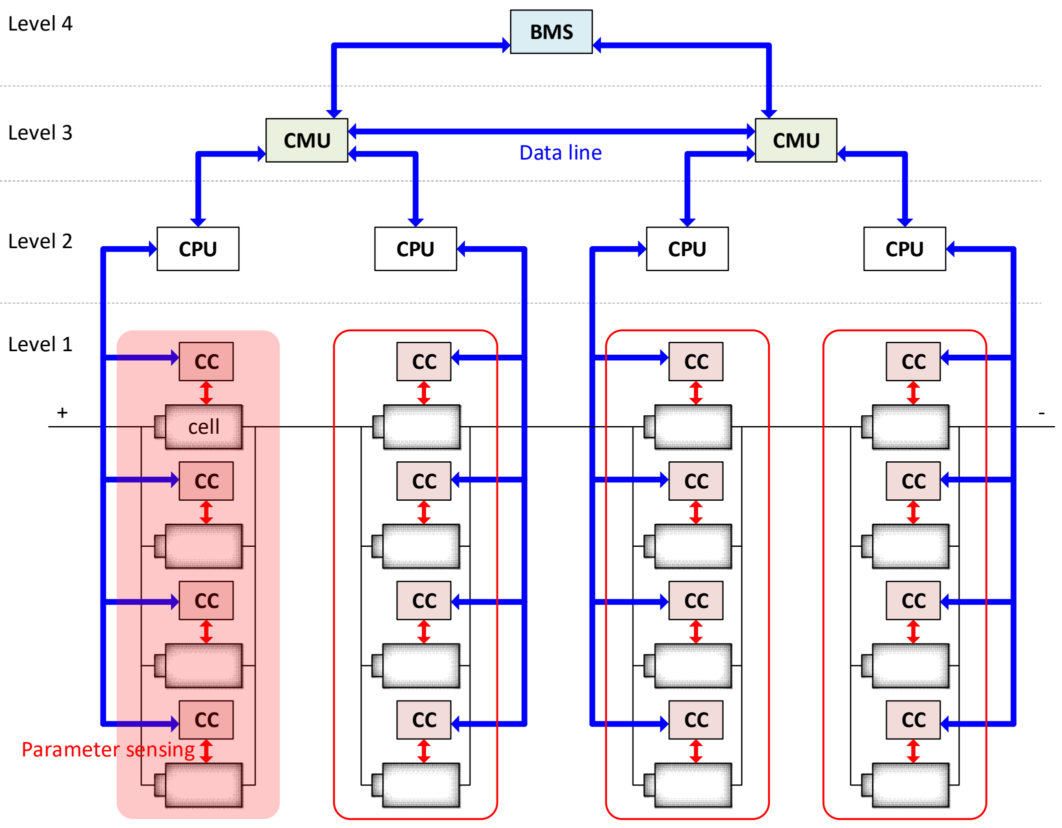


Figure 7. Proposed BMS for a battery pack

Individual Cell Controller

Hardware

To control a single cell and monitor its status, the CC has a Fuel Gauge IC and three MOSFETs. Stand–alone OCV-Based Fuel Gauge IC (model no. DS2786B) is used for this prototype mode. This IC can measure terminal voltage, cell temperature with internal or external sensors, and charging or discharging current with a shunt resistor of 6mΩ. These measurements are used to estimate SoC based on the OCV model. Figure 9 shows the OCV model used for this cell. Three

MOSFETs to switch modes are n-channel power MOSFET (CSD16321Q5) are used. Its threshold voltage is 1.4V because the signal level of 3.3V from Arduino as a main controller is used.

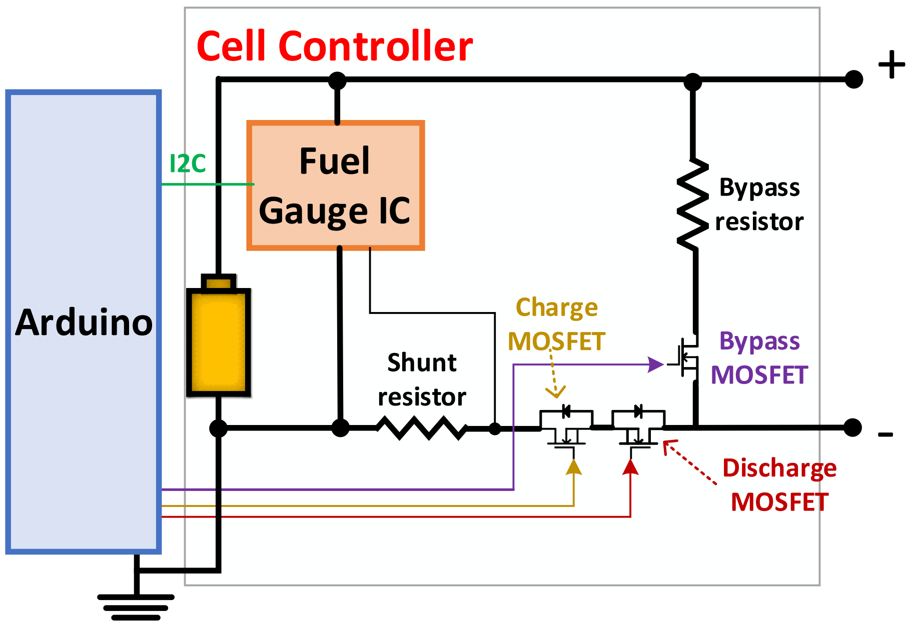


Figure 8. The Diagram for a Cell Controller

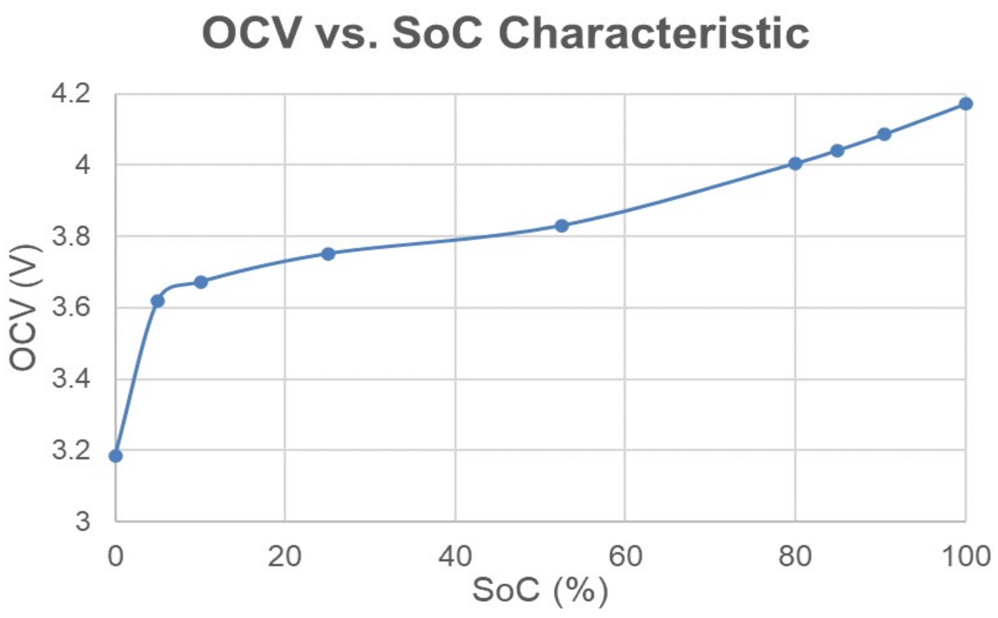


Figure 9. OCV model for a cell

Using Altium, which is PCB design software, the above circuit is designed as shown in Figure 10. VDD and Vbat+ bus are positive sides for a charger / a load and for a cell, respectively and, likewise, VSS and Vbat- are negative sides. Those buses would handle up to 8A according to the estimation, so the metal pattern for them should be more than 8mm in width. In addition, the metal area that handles high current and heat sink function has additional vias.

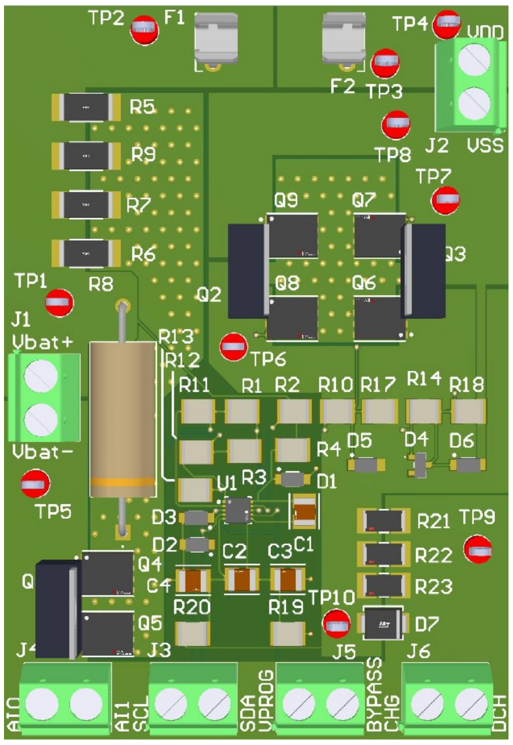


Figure 10. PCB design for a Cell Controller

Charging

When a cell is charging, Charging MOSFET is turned on, and the current flows, as shown with a dark green line in Figure 11. If there are multiple cells in charging and it has to be isolated with problems, Bypass MOSFET is turned on, and the current flows through the bypass resistor instead, as shown with a thin green line.

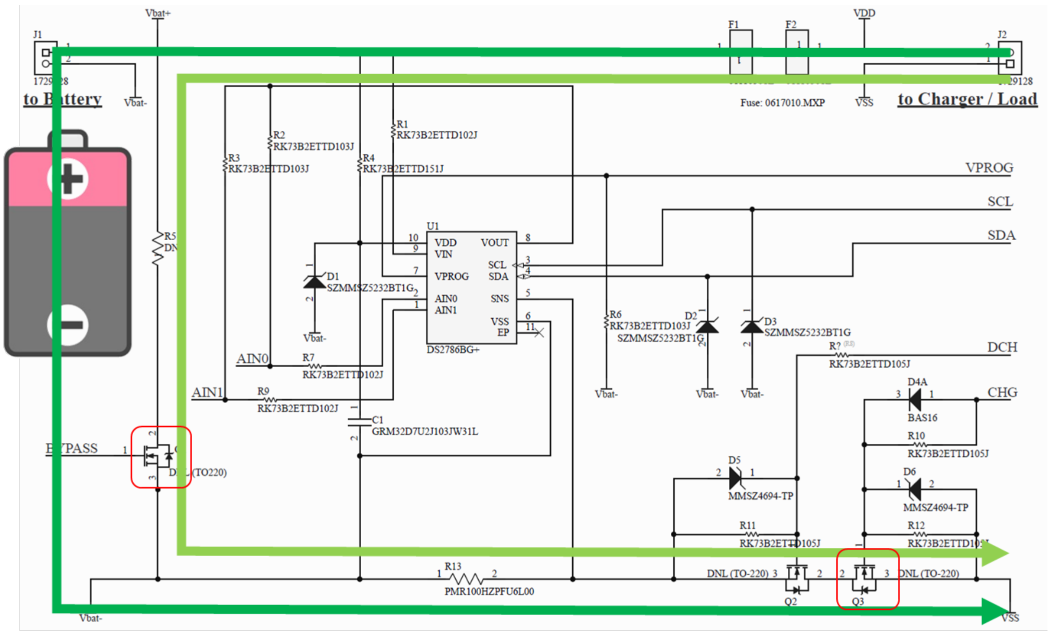


Figure 11. Current flow to charge a cell on a Cell Controller

Discharging

When a cell is discharging, Discharging MOSFET is turned on, and the current flows, as shown with a dark red line in Figure 12. If there are multiple parallel-connected cells in discharging and this cell is having problems, Discharging MOSFET is turned off, and the total battery power is reduced due to this.

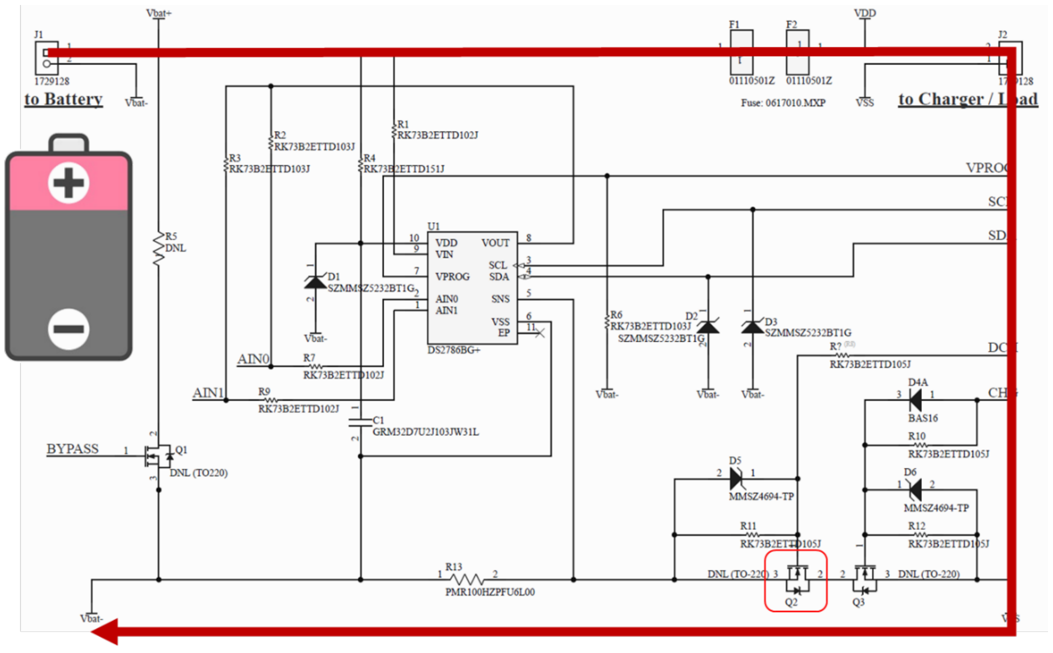


Figure 12. Current flow to discharge a cell on a Cell Controller

Software

The control flow of a cell controller is shown in Figure 13. A CC is powered on when a cell is inserted into the system. Then, the Fuel Gauge IC starts to measure terminal voltage, the current through the cell and ambient temperature and estimate SoC. It is reported to Arduino frequently, and Arduino displays the cell parameters on a PC. After this process, the CC checks if a cell is charging or discharging. When it is in charging mode, the CC checks if the terminal voltage of the cell is less than the charge threshold voltage and continues to charge the cell. Once the terminal voltage exceeds the threshold voltage, the CC stops charging. Meanwhile, in discharging mode, the CC checks if the terminal voltage of the cell is over-discharge threshold voltage. Like charging mode, the CC stops discharging as soon as the terminal voltage reaches the threshold voltage. It takes about 3 sec to implement one cycle of this process.

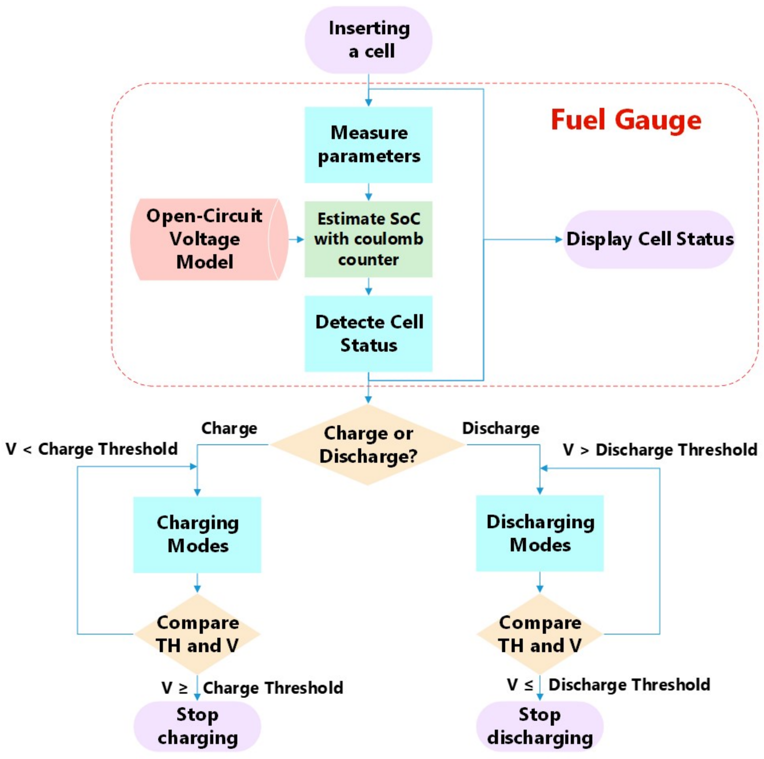


Figure 13. Control Flow on a Cell Controller

To do this, Arduino has to communicate with a CC through I2C. Figure 14 checks signals on SCL and SDA to successfully send data from Arduino. In the Test, the first data is 18 for a 7-bit slave address and a consecutive number are sent continuously. The blue line is the start condition, which is the falling edge of SDA and high on SCL, and then slave address (0b00110000) and write bit (0) are sent. Right after them, 1 is sent as a non-acknowledge bit because there is no slave device. After this, 0xB5 and a non-acknowledge bit are shown up. Finally, stop condition, which is the rising edge of SDA and high on SCL.

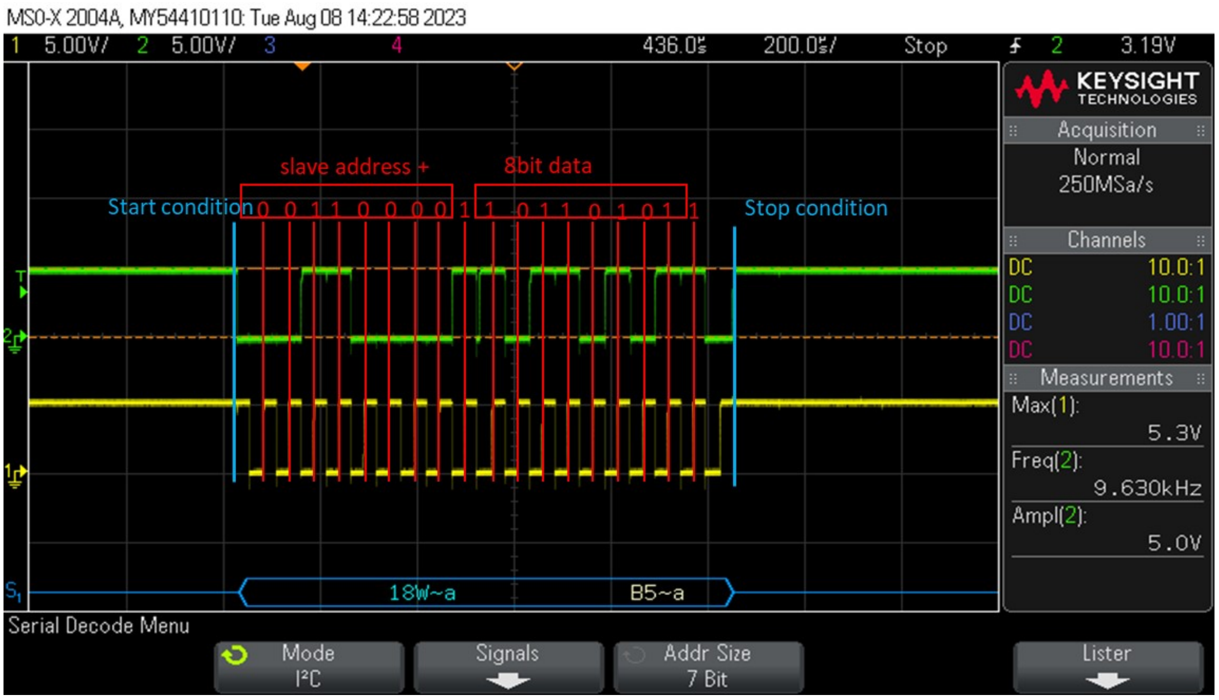


Figure 14. Signal of SCL and SDA on I2C line from Arduino

The SSPADD register has to be changed to change the baud rate of these signals. This is only for a master device. SSPADD register for a slave device is configured to set its slave address. At 172, the baud rate is 9770bps and 47us at one bit. At 64, the baud rate is 38460bps and 47us at one bit.

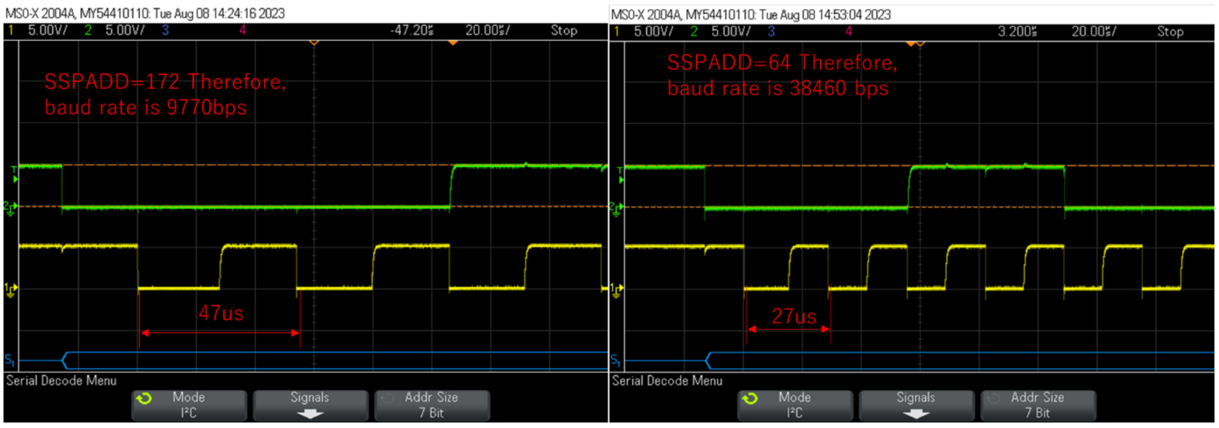

SSPADD=172 Therefore, baud rate is 9770bps
47us
SSPADD=64 Therefore, baud rate is 38460 bps
27us

Figure 15. Time for one bit related to setting Baud Rate at 9770bps and 38460bps

The following code is used to read data from the Fuel Gauge IC. This code sends the start condition first, and then it sends a slave address (Sadd). If Arduino receives an acknowledge bit (0), after 50ms waiting, Arduino receives 1 byte of data from a responded CC. When using this subroutine, Sadd is the slave address, which has to send cell status, and MAdd is the memory address, which the parameters are put in. Finally, Arduino sends a stop condition after receiving data. The memory addresses for each parameter are shown in the IC datasheet (attached in the appendices).

```
/* Subroutine for reading Memorys which are one byte */
byte ReadIC(byte SAdd, byte MAdd){
  int ReadData;

  Wire.beginTransmission(SAdd); // transmit to a slave device
  Wire.write(MAdd);             // write the memory address byte
  Wire.endTransmission(false);  // send the restart condition
  delay(50);

  Wire.requestFrom(SAdd, 1);    // request 1 bytes from slave device
  while(Wire.available()){      // read 1byte data
    ReadData = Wire.read();
  }
  return ReadData;              // return 1byte data
}
```

The voltage, current and temperature are 2 bytes of data, and two's complement form. After taking 2 bytes of data using a similar code to the previous one, the other subroutine converts them following each defined resolution and units by its datasheet. The following are the conversions for the main parameters.

Current (A) =(data * 25u) / 6m

Terminal Voltage (mV) = data * 1.22

Temperature (℃) = data * 0.125

The following code has to be used to change Status / Command to Cell Controllers. This code sends the start condition first, and then it sends a slave address (Sadd). If Arduino receives an acknowledge bit (0), after 50ms waiting, Arduino sends 1-byte data to a responded CC. When using this subroutine, Sadd is the slave address, which has to send cell status, and MAdd is the memory address in which the parameters are put. And then, Data is command/ status. Finally, Arduino sends a stop condition after sending data. This IC has EEPROM, so some important parameters or statuses can be saved in the IC and recalled when the IC restarts. To do this, after changing the data to save, 15V supply to V_{prog} PIN and Arduino sends a copy command. After waiting more than 80ms, the copy command has to be cleared by the software. The slave address can be changed in the same way, but the new slave address has to be used after changing the slave address.

```
/* Subroutine for writting Data which are one byte */
void SendIC(byte SAdd, byte MAdd, byte Data){
  Wire.beginTransmission(SAdd); // transmit to a slave device
  Wire.write(MAdd);        // write the memory address byte
  Wire.write(Data);        // write the data
  Wire.endTransmission();  // send the restart condition
  delay(50);
}
```

To control three MOSFETs, the IO pin on Arduino is connected to the Gate of the MOSFET, and Arduino sends a low signal at first to set all of them off. The signal to switch them on/off is sent from the serial monitor by entering '1' for charging mode, '2' for discharging mode, '3' for open-circuit mode and '4' for short-circuit mode. In addition, once charging or discharging starts, the terminal voltage is monitored, and it stops when the voltage reaches its threshold. The code to do this is the following.

```
/* function to prevent overdischarge */
  if(readvolt1<UVth1){      // comparing terminal voltage and threshold
    digitalWrite(dch,LOW); // diacharging is turned off
    digitalWrite(chg,LOW);  // charging is turned off
    Serial.println("Open-circuit mode for Cell1");
  }

  if(readvolt2<UVth2){
    digitalWrite(dch,LOW); // diacharging is turned off
    digitalWrite(chg,LOW);  // charging is turned on
    Serial.println("Open-circuit mode2");
  }

/* function to prevent overcharge */
  if(readvolt1>OVth1){      // comparing terminal voltage and threshold
    digitalWrite(dch,LOW); // diacharging is turned off
    digitalWrite(chg,LOW);  // charging is turned off
    Serial.println("Open-circuit mode for Cell1");
  }

  if(readvolt2>OVth2){
    digitalWrite(dch1,LOW);
    digitalWrite(chg1,LOW);
    Serial.println("Open-circuit mode for Cell2");
  }
```

Multi-Cell Connection

Cells in Parallel Connection

Figure 16 shows the parallel connection of cells. The right side of Figure 16 shows the I2C network among each controller. Each CC as a slave device has an individual slave address. According to Fuel Gauge IC, up to 16 slave devices can communicate with a master device on the same I2C bus. The left side of Figure 16 is shown as the parallel connected cells with CCs. In this phase, all CCs and Arduino use the same reference point (GND).

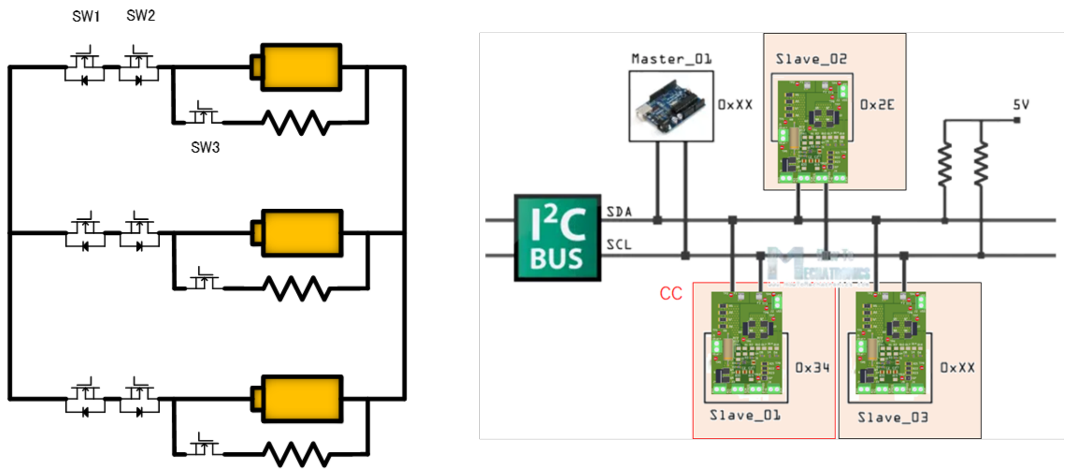


Figure 16. Circuit diagram and I2C network for parallel cell connection

Cells in Series Connection

Figure 17 shows the series of connected cells. However, there is an isolation problem to communicate with Arduino using I2C.

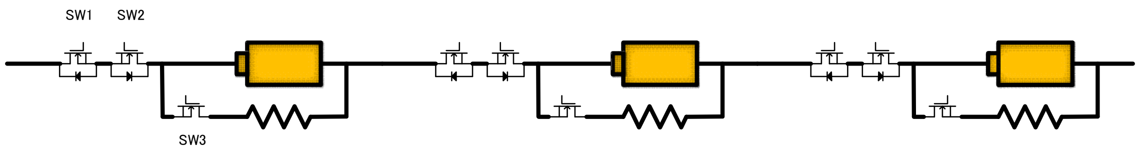


Figure 17. Circuit diagram for series cell connection

RESULT

Result for single cell

Charging Mode

A single cell was fully charged from empty at CC-CV charging using a stabilized power supply. The constant current is 0.67A, and the Constant Voltage is 4.2V. Figure 18 shows the changing of the terminal voltage and the current across a cell. Firstly, the voltage was increased during constant current charging. Then, the terminal voltage reached about 4.2V, and the charging current decreased until the current became 0.2mA, and CC stopped charging. The full capacity of this cell is 3350mAh, and the charging current is 0.67A, so it has to take five hours to charge this cell. However, the result is about 4.7h. Therefore, it is deduced that this cell has deteriorated.

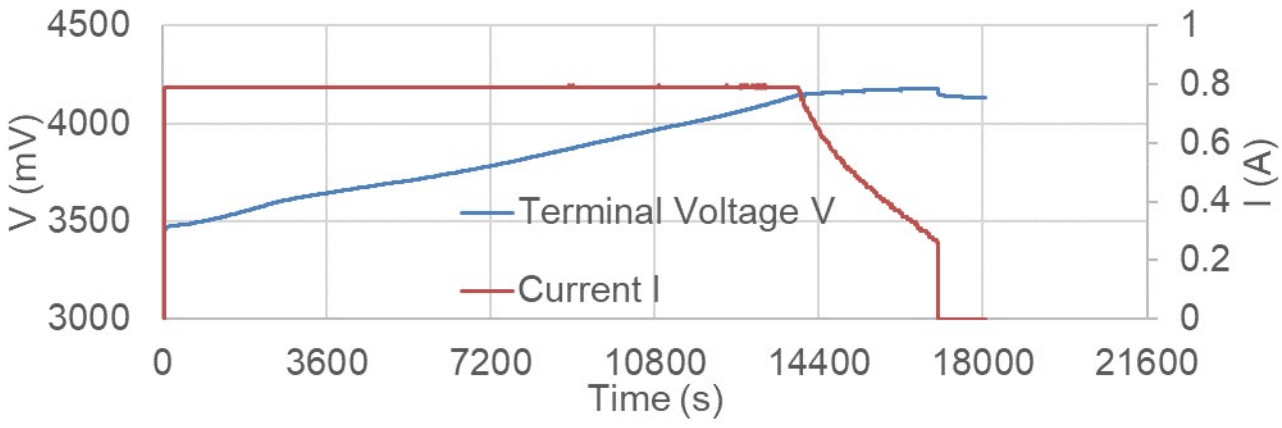


Figure 18. Terminal Voltage and Current in Charge Mode

The estimation of SoC is shown in Figure 19. During Constant Current (CC) charging, SoC is constantly increasing because the charging current is constant, and it is charged at 670mA/h. After changing the charging method to Constant Voltage (CV) charging, the SoC is increasing slowly compared with CC charging. 7.5 minutes after charging, the system measured the OCV of the cell and updated SoC to 98%. There is a difference between estimation by coulomb counter and the OCV model.

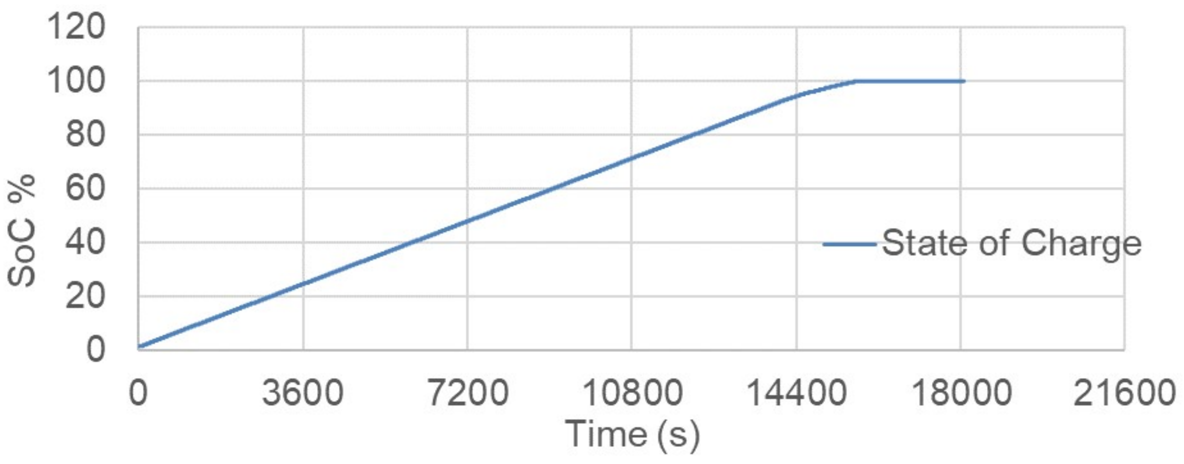
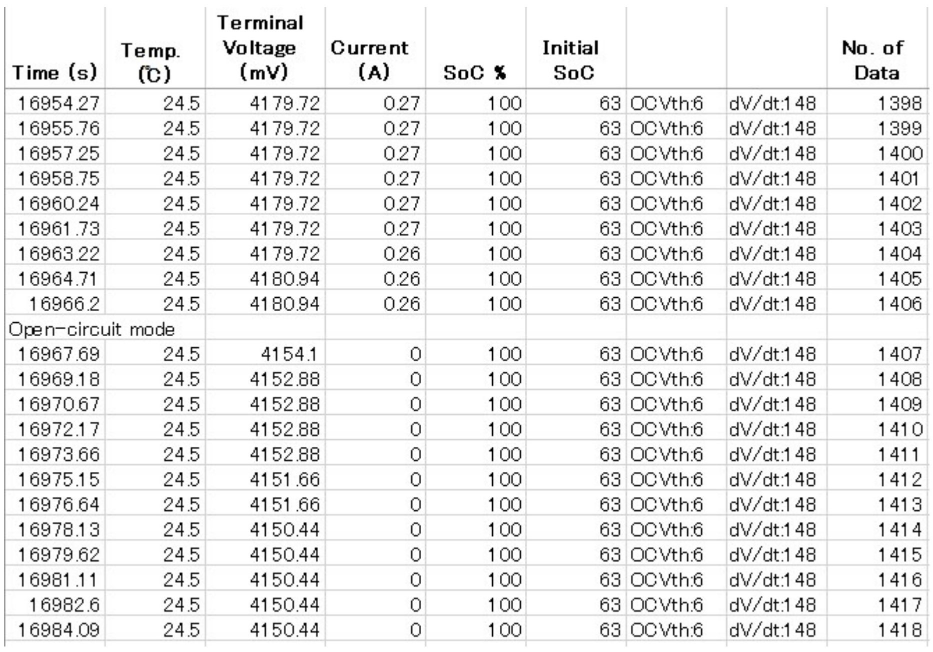


Figure 19. SoC in Charge Mode

In this experiment, the threshold of the cell voltage and current is set to 4.18V and 0.2A. therefore,

Table 3 shows that the system stops charging when the terminal voltage reaches the threshold.

Table 3. Display of Cell Status reaching Voltage to Charge threshold



Time (s)	Temp. (℃)	Terminal Voltage (mV)	Current (A)	SoC %	Initial SoC			No. of Data
16954.27	24.5	4179.72	0.27	100	63	OCVth:6	dV/dt:148	1398
16955.76	24.5	4179.72	0.27	100	63	OCVth:6	dV/dt:148	1399
16957.25	24.5	4179.72	0.27	100	63	OCVth:6	dV/dt:148	1400
16958.75	24.5	4179.72	0.27	100	63	OCVth:6	dV/dt:148	1401
16960.24	24.5	4179.72	0.27	100	63	OCVth:6	dV/dt:148	1402
16961.73	24.5	4179.72	0.27	100	63	OCVth:6	dV/dt:148	1403
16963.22	24.5	4179.72	0.26	100	63	OCVth:6	dV/dt:148	1404
16964.71	24.5	4180.94	0.26	100	63	OCVth:6	dV/dt:148	1405
16966.2	24.5	4180.94	0.26	100	63	OCVth:6	dV/dt:148	1406
Open-circuit mode								
16967.69	24.5	4154.1	0	100	63	OCVth:6	dV/dt:148	1407
16969.18	24.5	4152.88	0	100	63	OCVth:6	dV/dt:148	1408
16970.67	24.5	4152.88	0	100	63	OCVth:6	dV/dt:148	1409
16972.17	24.5	4152.88	0	100	63	OCVth:6	dV/dt:148	1410
16973.66	24.5	4152.88	0	100	63	OCVth:6	dV/dt:148	1411
16975.15	24.5	4151.66	0	100	63	OCVth:6	dV/dt:148	1412
16976.64	24.5	4151.66	0	100	63	OCVth:6	dV/dt:148	1413
16978.13	24.5	4150.44	0	100	63	OCVth:6	dV/dt:148	1414
16979.62	24.5	4150.44	0	100	63	OCVth:6	dV/dt:148	1415
16981.11	24.5	4150.44	0	100	63	OCVth:6	dV/dt:148	1416
16982.6	24.5	4150.44	0	100	63	OCVth:6	dV/dt:148	1417
16984.09	24.5	4150.44	0	100	63	OCVth:6	dV/dt:148	1418

Discharging Mode

A single cell was discharged from a fully charged condition. A 4.7Ω resistor (20W rating) is used as a load. When discharging, the smaller the terminal voltage and the current of the cell are, the smaller the remaining capacity is. In the end, the terminal voltage reached 2.8V of the discharge threshold, and CC stopped discharging. At this time, SoC is decreasing following the amount of current flow. Therefore, it could stop discharging at the correct time. However, when SoC reaches 0%, it is earlier than when the terminal voltage reaches 2.8V.

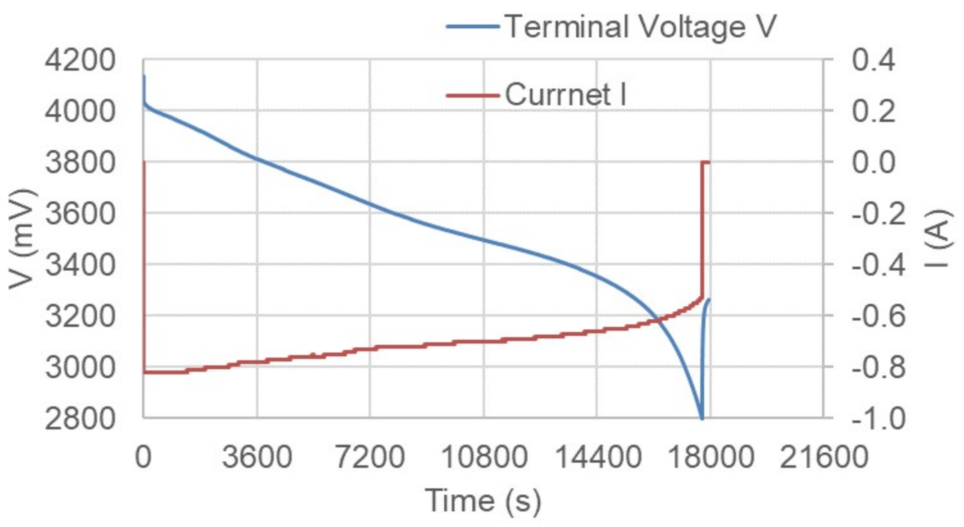


Figure 20. Terminal Voltage and Current in Discharge Mode

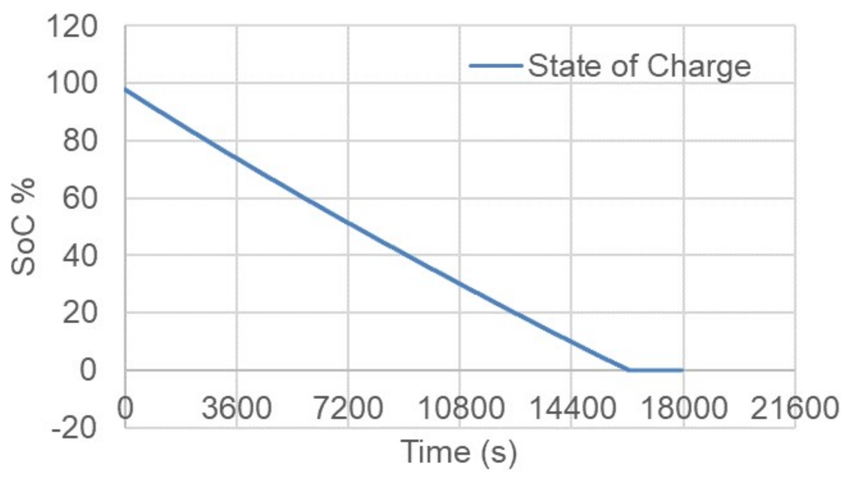
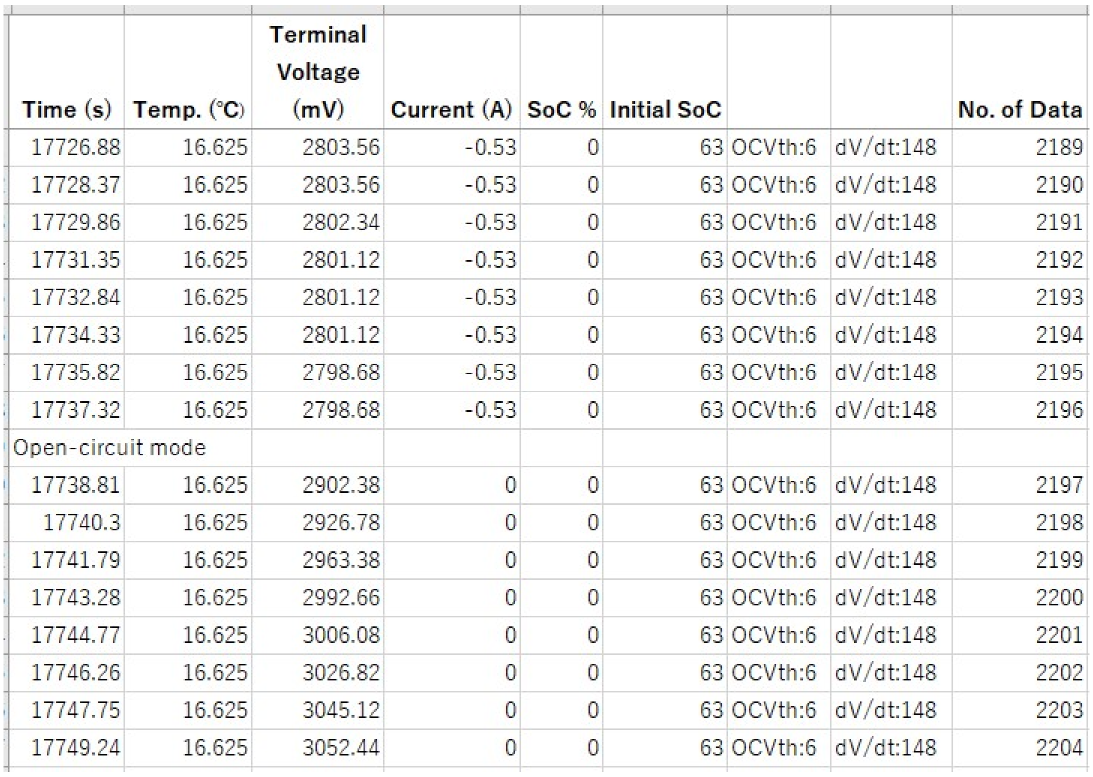


Figure 21. SoC in Discharge Mode

In this experiment, the threshold of the cell voltage and current is set to 2.8V and -0.2A. Therefore, Table 4 shows that the system stops discharging when the terminal voltage reaches the threshold.

Table 4. Display of Cell Status reaching Voltage to Discharge Threshold



Time (s)	Temp. (°C)	Terminal Voltage (mV)	Current (A)	SoC %	Initial SoC			No. of Data
17726.88	16.625	2803.56	-0.53	0	63	OCVth:6	dV/dt:148	2189
17728.37	16.625	2803.56	-0.53	0	63	OCVth:6	dV/dt:148	2190
17729.86	16.625	2802.34	-0.53	0	63	OCVth:6	dV/dt:148	2191
17731.35	16.625	2801.12	-0.53	0	63	OCVth:6	dV/dt:148	2192
17732.84	16.625	2801.12	-0.53	0	63	OCVth:6	dV/dt:148	2193
17734.33	16.625	2801.12	-0.53	0	63	OCVth:6	dV/dt:148	2194
17735.82	16.625	2798.68	-0.53	0	63	OCVth:6	dV/dt:148	2195
17737.32	16.625	2798.68	-0.53	0	63	OCVth:6	dV/dt:148	2196
Open-circuit mode								
17738.81	16.625	2902.38	0	0	63	OCVth:6	dV/dt:148	2197
17740.3	16.625	2926.78	0	0	63	OCVth:6	dV/dt:148	2198
17741.79	16.625	2963.38	0	0	63	OCVth:6	dV/dt:148	2199
17743.28	16.625	2992.66	0	0	63	OCVth:6	dV/dt:148	2200
17744.77	16.625	3006.08	0	0	63	OCVth:6	dV/dt:148	2201
17746.26	16.625	3026.82	0	0	63	OCVth:6	dV/dt:148	2202
17747.75	16.625	3045.12	0	0	63	OCVth:6	dV/dt:148	2203
17749.24	16.625	3052.44	0	0	63	OCVth:6	dV/dt:148	2204

Changing and discharging for two cells in parallel

Charging Mode

Two cells in parallel connection are charged at CC-CV charging. The Constant Current is 1.34A, and the Constant Voltage is 4.2V. One cell has 3.83V of OCV, and the other has 3.2V. Because of this, there was a difference between them in voltage and current, but in the end, both voltage and current became equal.

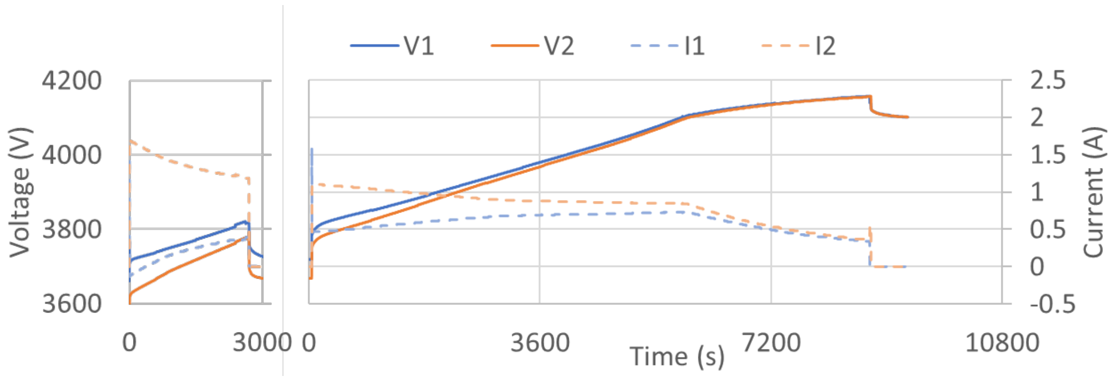


Figure 22. Terminal Voltage and Current in Charge Mode for two parallel cells

However, the estimated SoC for them did not correspond to the current because those cells were discharging and charging each other.

Table 5. Display of Cell Status in stopping charging at 3.8V for two parallel cells

Cell 1							Cell 2						
Time (s)	Temp. (℃)	Terminal Voltage (mV)	Current (A)	SoC %			Temp. (℃)	Terminal Voltage (mV)	Current (A)	SoC %			No. of Data
2657.24	24.625	3817.38	0.39	54	OCVth:6	dV/dt:148	24.875	3775.9	1.19	80	OCVth:6	dV/dt:148	993
2660.12	24.625	3817.38	0.39	54	OCVth:6	dV/dt:148	24.875	3775.9	1.19	80	OCVth:6	dV/dt:148	994
2663.01	24.625	3817.38	0.39	54	OCVth:6	dV/dt:148	24.875	3777.12	1.19	80	OCVth:6	dV/dt:148	995
2665.89	24.625	3817.38	0.39	54	OCVth:6	dV/dt:148	24.875	3777.12	1.19	80	OCVth:6	dV/dt:148	996
2668.77	24.625	3817.38	0.39	54	OCVth:6	dV/dt:148	24.875	3777.12	1.19	80	OCVth:6	dV/dt:148	997
2671.65	24.625	3817.38	0.39	54	OCVth:6	dV/dt:148	24.875	3777.12	1.19	80	OCVth:6	dV/dt:148	998
2674.53	24.625	3817.38	0.39	54	OCVth:6	dV/dt:148	24.875	3777.12	1.19	80	OCVth:6	dV/dt:148	999
2677.42	24.625	3817.38	0.39	54	OCVth:6	dV/dt:148	24.875	3777.12	1.19	80	OCVth:6	dV/dt:148	1000
2680.3	24.625	3817.38	0.39	54	OCVth:6	dV/dt:148	24.875	3777.12	1.19	80	OCVth:6	dV/dt:148	1001
2683.18	24.625	3817.38	0.39	54	OCVth:6	dV/dt:148	24.875	3777.12	1.19	80	OCVth:6	dV/dt:148	1002
2686.06	24.625	3817.38	0.39	54	OCVth:6	dV/dt:148	24.875	3777.12	1.19	80	OCVth:6	dV/dt:148	1003
2688.94	24.625	3817.38	0.39	54	OCVth:6	dV/dt:148	24.875	3777.12	1.19	80	OCVth:6	dV/dt:148	1004
2691.83	24.625	3817.38	0.39	54	OCVth:6	dV/dt:148	24.875	3777.12	1.19	80	OCVth:6	dV/dt:148	1005
2694.71	24.625	3817.38	0.39	54	OCVth:6	dV/dt:148	24.875	3778.34	1.19	80	OCVth:6	dV/dt:148	1006
2697.59	24.625	3817.38	0.39	54	OCVth:6	dV/dt:148	24.875	3778.34	1.19	80	OCVth:6	dV/dt:148	1007
Open-circuit mode for cell1													
2700.47	24.625	3817.38	0.08	54	OCVth:6	dV/dt:148	24.875	3778.34	1.19	80	OCVth:6	dV/dt:148	1008
Open-circuit mode for cell2													
2703.35	24.625	3802.74	0	54	OCVth:6	dV/dt:148	24.875	3791.76	0.42	80	OCVth:6	dV/dt:148	1009
2706.24	24.625	3757.6	0	54	OCVth:6	dV/dt:148	24.875	3696.6	0	80	OCVth:6	dV/dt:148	1010
2709.12	24.625	3756.38	0	54	OCVth:6	dV/dt:148	24.875	3694.16	0	80	OCVth:6	dV/dt:148	1011
2712	24.625	3755.16	0	54	OCVth:6	dV/dt:148	24.875	3691.72	0	80	OCVth:6	dV/dt:148	1012
2714.88	24.625	3753.94	0	54	OCVth:6	dV/dt:148	24.875	3691.72	0	80	OCVth:6	dV/dt:148	1013
2717.77	24.625	3752.72	0	54	OCVth:6	dV/dt:148	24.875	3689.28	0	80	OCVth:6	dV/dt:148	1014
2720.65	24.625	3752.72	0	54	OCVth:6	dV/dt:148	24.875	3689.28	0	80	OCVth:6	dV/dt:148	1015
2723.53	24.625	3751.5	0	54	OCVth:6	dV/dt:148	24.875	3688.06	0	80	OCVth:6	dV/dt:148	1016
2726.41	24.625	3750.28	0	54	OCVth:6	dV/dt:148	24.875	3686.84	0	80	OCVth:6	dV/dt:148	1017
2729.29	24.625	3750.28	0	54	OCVth:6	dV/dt:148	24.875	3686.84	0	80	OCVth:6	dV/dt:148	1018
2732.18	24.625	3749.06	0	54	OCVth:6	dV/dt:148	24.875	3686.84	0	80	OCVth:6	dV/dt:148	1019
2735.06	24.625	3749.06	0	54	OCVth:6	dV/dt:148	24.875	3685.62	0	80	OCVth:6	dV/dt:148	1020
2737.94	24.625	3747.84	0	54	OCVth:6	dV/dt:148	24.875	3684.4	0	80	OCVth:6	dV/dt:148	1021
2740.82	24.625	3747.84	0	54	OCVth:6	dV/dt:148	24.875	3684.4	0	80	OCVth:6	dV/dt:148	1022
2743.7	24.625	3747.84	0	54	OCVth:6	dV/dt:148	24.875	3684.4	0	80	OCVth:6	dV/dt:148	1023

DISCUSSION

Accuracy of an individual cell controller

SoC estimation

Firstly, the measurements are not accurate. The voltage has a 0.01V difference between the IC and manual measuring result. The current has about a 0.06A difference between them. The reference point was common with the GND of Arduino. It would be one of the reasons for this problem.

This Fuel Gauge did not consider the cell deterioration. Therefore, the result of the estimated SoC would be incorrect a little further on. To solve this problem, an additional estimation for SoH is required to adjust SoC considering the cell deterioration. In addition, the OCV model almost fits the current cell condition as an initial condition, but it should have been calculated using OCV model simulation tools such as Simulink of MATLAB.

MOSFET switching

The designed prototype sends the signal to MOSFETs from Arduino. Thus, CC cannot control MOSFETs when cells are combined in series and in parallel. To control all MOSFETs, no matter which cell combination, the cell controller has to be isolated and send the signal to MOSFETs.

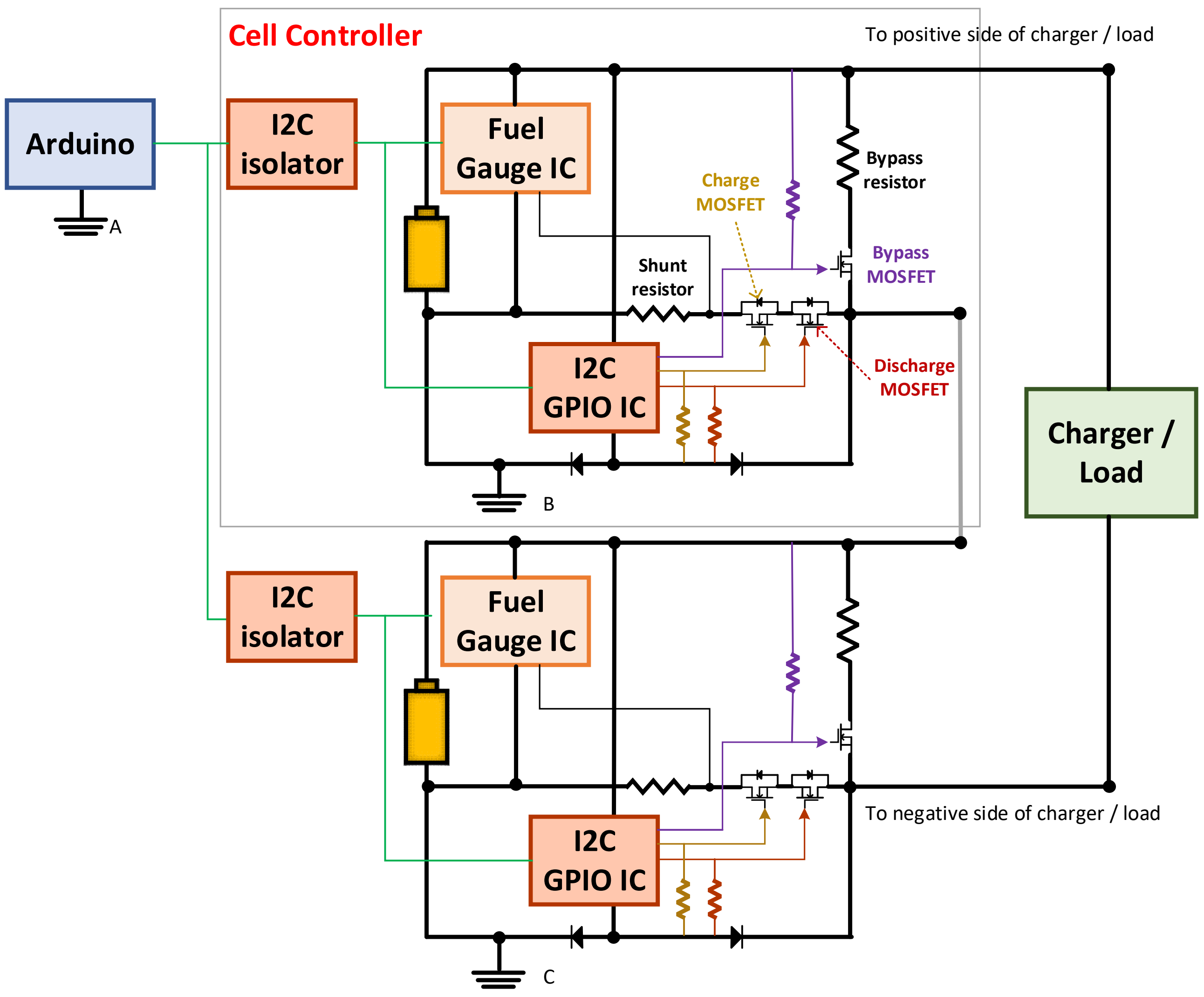


Figure 23. The improved prototype cell controller in series connection

CONCLUSIONS

The purpose of this project was to connect controllers to each cell of the assembled battery that could individually monitor and control them. These controllers would report the battery's status to the main controller through I2C communication. For each individual controller, on/off control of charging and discharging was made possible through multiple MOSFETs. However, the accuracy of State of Charge (SoC) estimation was reduced due to the configuration of the Fuel Gauge IC and ground insulation. Challenges include finding accurate methods for estimating Open Circuit Voltage (OCV) and providing feedback on cell degradation to SoC, as well as addressing the issue of common ground in CC.

Furthermore, when two or more cells are connected in a battery assembly:

- In the case of parallel connections, a voltage measurement is possible, but an unexpected common ground leads to current flowing, bypassing the sensors and making accurate current measurement impossible for each cell.
- In the case of series connections, to measure each cell's voltage, separate reference points must be established, and as of the current stage without insulation, this cannot be confirmed.

However, it is believed that the concerns mentioned above can be addressed in the next step by incorporating I2C isolation ICs and I2C GPIO ICs. This would allow MOSFETs to receive signals from I2C, and each controller can operate on its own ground, thereby resolving the issues mentioned above.

REFERENCES

1. A. Zhang, S. Song, C. Wang, J. Zhang, K. Wang, L. Li, *Research of Battery Management System for Integrated Power Supply*. IEEE 2017, 3178-3181.

2. C. R. Birkl, M. R. Roberts, E. McTurk, P. G. Bruce, D. A. Howey, *Degradation diagnostics for lithium ion cells*. Journal of Power Source. 341(2017) 373-386.

3. C. Zhang, J. Jiang, L. Zhang, S. Liu, L. Wang, P. C. Loh, *A Generalized SOC-OCV Model for Lithium-Ion Batteries and the SOC Estimation for LNMCO Battery*. Energies 2016, 9(11), 900; https://doi.org/10.3390/en9110900

4. E. Gotaas and A. Nettum, *Single cell battery management systems (BMS)*. INTELEC. Twenty-Second International Telecommunications Energy Conference (Cat. No.00CH37131), Phoenix, AZ, USA, 2000, pp. 695-702, doi: 10.1109/INTLEC.2000.884324.

5. G. Zubi, R. Dufo-López, M. Carvalho, G. Pasaoglu, *The lithium-ion battery: State of the art and future perspectives. Renewable and Sustainable Energy Reviews*, Volume 89, June 2018, Pages 292-308.

6. H. Niu, L. Wang, P. Guan, N. Zhang, C. Yan, M. Ding, X. Guo, T. Huang, X. Hu, *Recent Advances in Application of Ionic Liquids in Electrolyte of Lithium Ion Batteries*. Journal of Energy Storage, 40 (2021) 102659.

7. K. C. Chiu, C. H. Lin, S. F. Yeh, Y. H. Lin, C. S. Huang, and K. C. Chen, *Cycle life analysis of series connected lithium-ion batteries with temperature difference*. Journal of Power Sources, vol.263, no.1, pp.75-84, Oct. 2014.

8. M. Kim, K. Kim, J. Kim, J. Yu, S. Han, *State of Charge Estimation for Lithium Ion Battery Based on Reinforcement Learning*. IFAC-PapersOnLine,Volume 51, Issue 28, 2018, Pages 404-408

9. M. Wakihara, Recent developments in lithium ion batteries. *Materials Science and Engineering*: R: Reports,Volume 33, Issue 4, 1st June 2001, Pages 109-134

10. N. yang, X. Zhang, B. Shang, and G. Li, *Unbalanced discharging and aging due to temperature difference among the cells in a lithium-ion battery pack with parallel combination.* Journal of Power Sources, vol.306, no.29, pp.733-741, Feb. 2016.

11. N. Yang, X. Zhang, B. Shang, G. Li, *Unbalanced discharging and aging due to temperature differences among the cells in a lithium-ion battery pack with parallel combination*. Journal of Power Source, Volume 306, 29th February 2016, Pages 733-741

12. P. Weicker, A Systems Approach to Lithium-Ion Battery Management , Artech, 2013

13. R. Gogoana, M. B. Pinson, M. Z. bazant, S. E. Sarma, *Internal resistance matching for parallel-connected lithium-ion cells and impacts on battery pack cycle life*. Journal of Power Sources, vol.252, no.15, pp.8-13, Apr. 2014.

14. T. Cai, P. Mohtat, A. G. Stefanopoulou, J. B. Siegel, *Li-ion Battery Fault Detection in Large Packs Using Force and Gas Sensor*. IFAC PapersOnLine 53-2 (2020) 12491-12496.

15. X. Gong, R. Xiong and C. C. Mi, *Study of the characteristics of battery packs in electric vehicles with parallel-connected lithium-ion battery cells*. 2014 IEEE Applied Power Electronics Conference and Exposition - APEC 2014, Fort Worth, TX, USA, 2014, pp. 3218-3224, doi: 10.1109/APEC.2014.6803766.

16. Y. Chen, Y. Kang, Y. Zhao, L. Wang, J. Liu, Y. Li, Z. Liang, X. He, X. Li, N. Tavajohi, B. Li, *A review of lithium-ion battery safety concerns: The issues, strategies, and testing standards*. Journal of Energy Chemistry, 59 (2021)

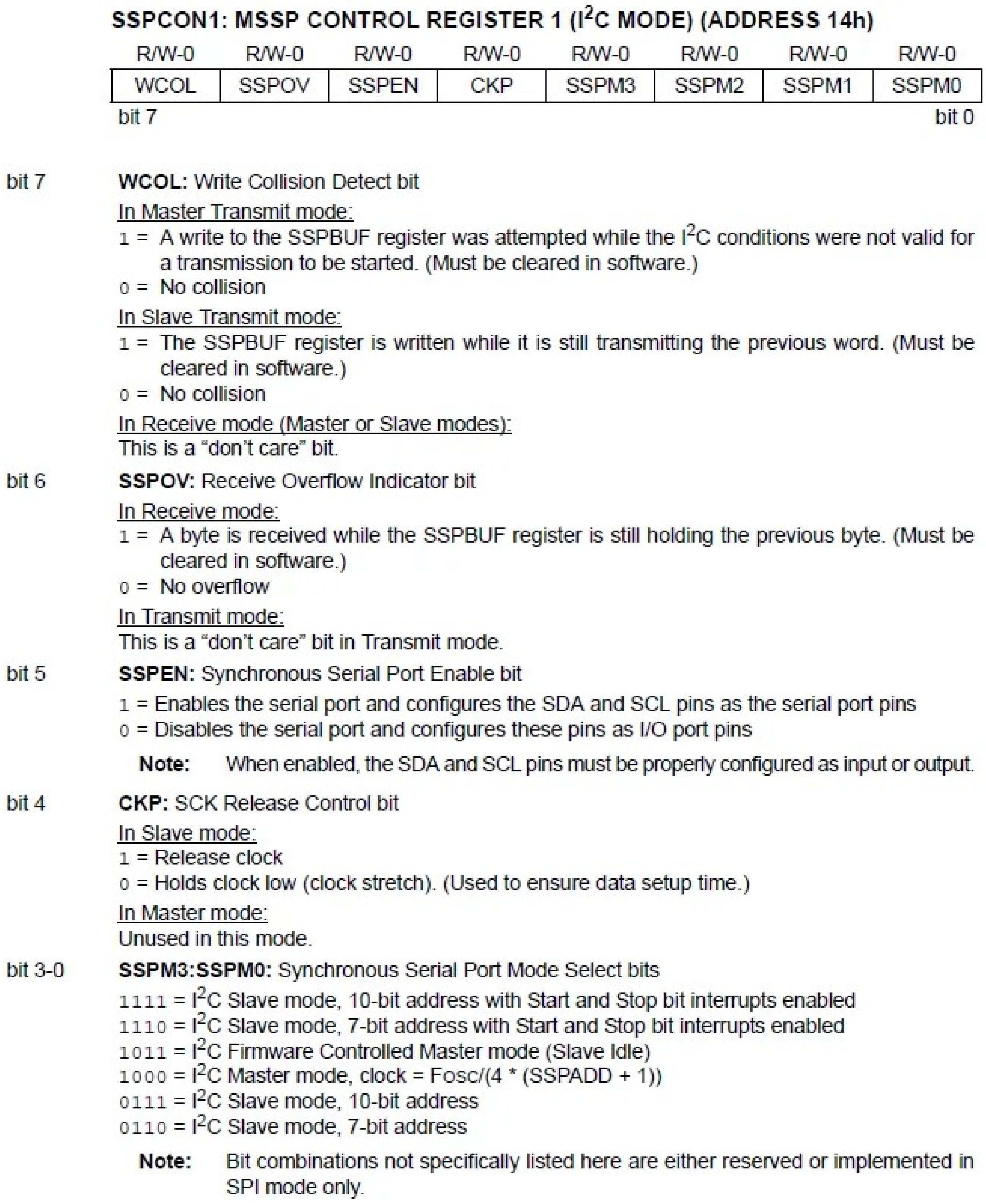
17. Y. Hato, J. Kato, T. Hirota, Y. Kamiya, Y. Daisho, *Improvement of Open Circuit Voltage Estimation Method for Lithium Ion Battery*. Jidosha Gijutukai Ronbunsyu, Vol.47, No2, March 2016, pp419-424.

18. Y. Wang, Y. Zhao, S. Zhou, Q. Yan, H. Zhan, Y. Cheng, W. Yim, *Impact of Individual Cell Parameter Difference on the Performance of Series-Parallel Battery Packs*. ACS Omega 2023, 8, 10512-10524.

19. Z. Wang, G. Feng, X. Liu, F. Gu, A. Ball, *A novel method of parameter identification and state of charge estimation for lithium-ion battery energy storage system*. Journal of Energy Storage, Volume 49, May 2022, 104124

APPENDICES

- Configuration and Status Register for I2C on PIC



SSPCON1: MSSP CONTROL REGISTER 1 (I^2C MODE) (ADDRESS 14h)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
bit 7							bit 0

bit 7 **WCOL:** Write Collision Detect bit

In Master Transmit mode:
1 = A write to the SSPBUF register was attempted while the I^2C conditions were not valid for a transmission to be started. (Must be cleared in software.)
0 = No collision

In Slave Transmit mode:
1 = The SSPBUF register is written while it is still transmitting the previous word. (Must be cleared in software.)
0 = No collision

In Receive mode (Master or Slave modes):
This is a "don't care" bit.

bit 6 **SSPOV:** Receive Overflow Indicator bit

In Receive mode:
1 = A byte is received while the SSPBUF register is still holding the previous byte. (Must be cleared in software.)
0 = No overflow

In Transmit mode:
This is a "don't care" bit in Transmit mode.

bit 5 **SSPEN:** Synchronous Serial Port Enable bit

1 = Enables the serial port and configures the SDA and SCL pins as the serial port pins
0 = Disables the serial port and configures these pins as I/O port pins

Note: When enabled, the SDA and SCL pins must be properly configured as input or output.

bit 4 **CKP:** SCK Release Control bit

In Slave mode:
1 = Release clock
0 = Holds clock low (clock stretch). (Used to ensure data setup time.)

In Master mode:
Unused in this mode.

bit 3-0 **SSPM3:SSPM0:** Synchronous Serial Port Mode Select bits

1111 = I^2C Slave mode, 10-bit address with Start and Stop bit interrupts enabled
1110 = I^2C Slave mode, 7-bit address with Start and Stop bit interrupts enabled
1011 = I^2C Firmware Controlled Master mode (Slave Idle)
1000 = I^2C Master mode, clock = Fosc/(4 * (SSPADD + 1))
0111 = I^2C Slave mode, 10-bit address
0110 = I^2C Slave mode, 7-bit address

Note: Bit combinations not specifically listed here are either reserved or implemented in SPI mode only.

SSPCON2: MSSP CONTROL REGISTER 2 (I^2C MODE) (ADDRESS 91h)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
bit 7							bit 0

bit 7 **GCEN:** General Call Enable bit (Slave mode only)
1 = Enable interrupt when a general call address (0000h) is received in the SSPSR
0 = General call address disabled

bit 6 **ACKSTAT:** Acknowledge Status bit (Master Transmit mode only)
1 = Acknowledge was not received from slave
0 = Acknowledge was received from slave

bit 5 **ACKDT:** Acknowledge Data bit (Master Receive mode only)
1 = Not Acknowledge
0 = Acknowledge

Note: Value that will be transmitted when the user initiates an Acknowledge sequence at the end of a receive.

bit 4 **ACKEN:** Acknowledge Sequence Enable bit (Master Receive mode only)
1 = Initiate Acknowledge sequence on SDA and SCL pins and transmit ACKDT data bit. Automatically cleared by hardware.
0 = Acknowledge sequence Idle

bit 3 **RCEN:** Receive Enable bit (Master mode only)
1 = Enables Receive mode for I^2C
0 = Receive Idle

bit 2 **PEN:** Stop Condition Enable bit (Master mode only)
1 = Initiate Stop condition on SDA and SCL pins. Automatically cleared by hardware.
0 = Stop condition Idle

bit 1 **RSEN:** Repeated Start Condition Enabled bit (Master mode only)
1 = Initiate Repeated Start condition on SDA and SCL pins. Automatically cleared by hardware.
0 = Repeated Start condition Idle

bit 0 **SEN:** Start Condition Enabled/Stretch Enabled bit
In Master mode:
1 = Initiate Start condition on SDA and SCL pins. Automatically cleared by hardware.
0 = Start condition Idle
In Slave mode:
1 = Clock stretching is enabled for both slave transmit and slave receive (stretch enabled)
0 = Clock stretching is enabled for slave transmit only (PIC16F87X compatibility)

SSPSTAT: MSSP STATUS REGISTER (I^2C MODE) (ADDRESS 94h)

R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
SMP	CKE	D/$\overline{\text{A}}$	P	S	R/$\overline{\text{W}}$	UA	BF
bit 7							bit 0

bit 7 **SMP:** Slew Rate Control bit

In Master or Slave mode:
1 = Slew rate control disabled for standard speed mode (100 kHz and 1 MHz)
0 = Slew rate control enabled for high-speed mode (400 kHz)

bit 6 **CKE:** SMBus Select bit

In Master or Slave mode:
1 = Enable SMBus specific inputs
0 = Disable SMBus specific inputs

bit 5 **D/$\overline{\text{A}}$:** Data/$\overline{\text{Address}}$ bit

In Master mode:
Reserved.

In Slave mode:
1 = Indicates that the last byte received or transmitted was data
0 = Indicates that the last byte received or transmitted was address

bit 4 **P:** Stop bit

1 = Indicates that a Stop bit has been detected last
0 = Stop bit was not detected last

Note: This bit is cleared on Reset and when SSPEN is cleared.

bit 3 **S:** Start bit

1 = Indicates that a Start bit has been detected last
0 = Start bit was not detected last

Note: This bit is cleared on Reset and when SSPEN is cleared.

bit 2 **R/$\overline{\text{W}}$:** Read/$\overline{\text{Write}}$ bit information (I^2C mode only)

In Slave mode:
1 = Read
0 = Write

Note: This bit holds the R/$\overline{\text{W}}$ bit information following the last address match. This bit is only valid from the address match to the next Start bit, Stop bit or not $\overline{\text{ACK}}$ bit.

In Master mode:
1 = Transmit is in progress
0 = Transmit is not in progress

Note: ORing this bit with SEN, RSEN, PEN, RCEN or ACKEN will indicate if the MSSP is in Idle mode.

bit 1 **UA:** Update Address (10-bit Slave mode only)

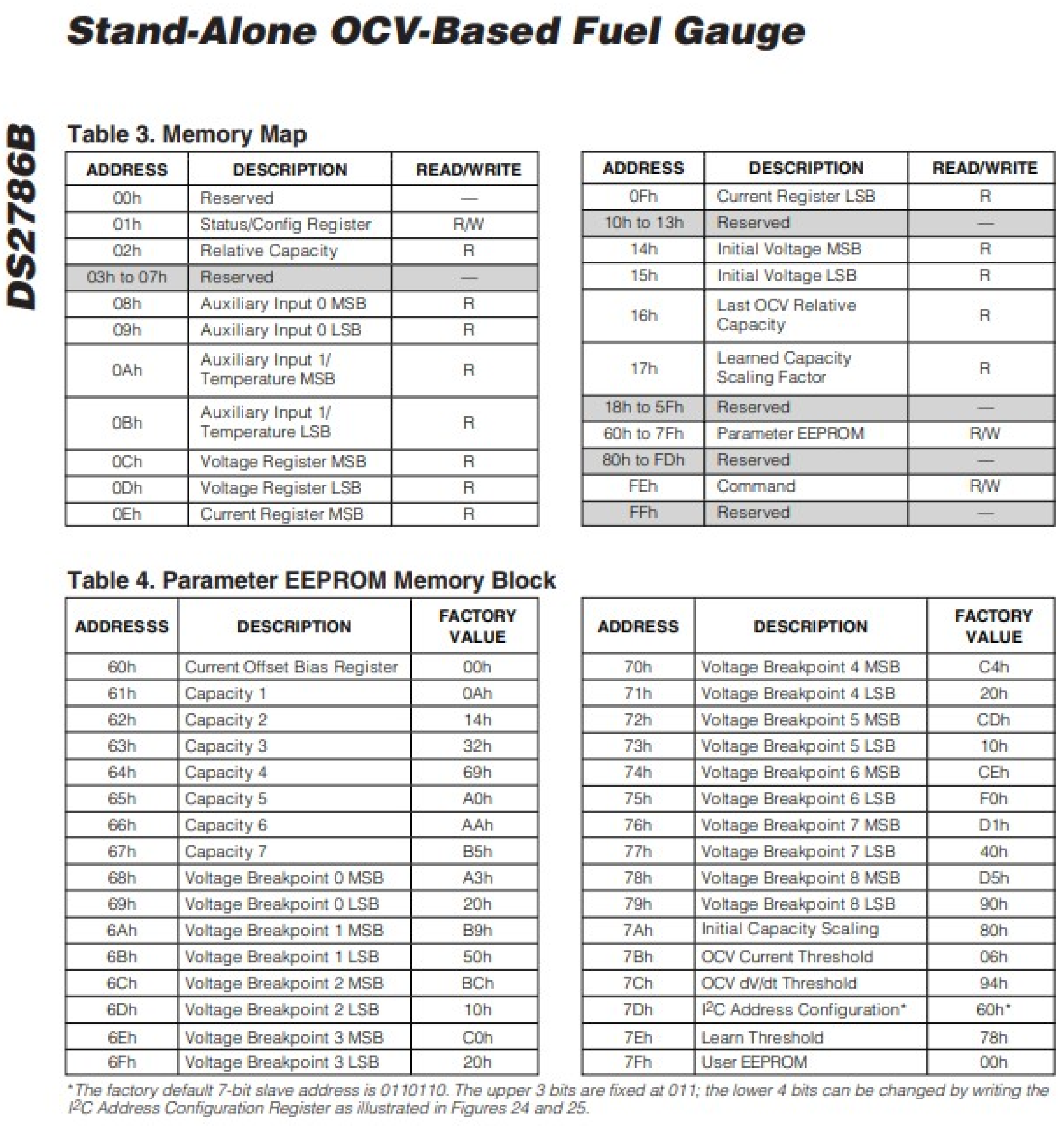
1 = Indicates that the user needs to update the address in the SSPADD register
0 = Address does not need to be updated

bit 0 **BF:** Buffer Full Status bit

In Transmit mode:
1 = Receive complete, SSPBUF is full
0 = Receive not complete, SSPBUF is empty

In Receive mode:
1 = Data Transmit in progress (does not include the $\overline{\text{ACK}}$ and Stop bits), SSPBUF is full
0 = Data Transmit complete (does not include the $\overline{\text{ACK}}$ and Stop bits), SSPBUF is empty

- Memory Map for DS2786B



DS2786B

Stand-Alone OCV-Based Fuel Gauge

Table 3. Memory Map

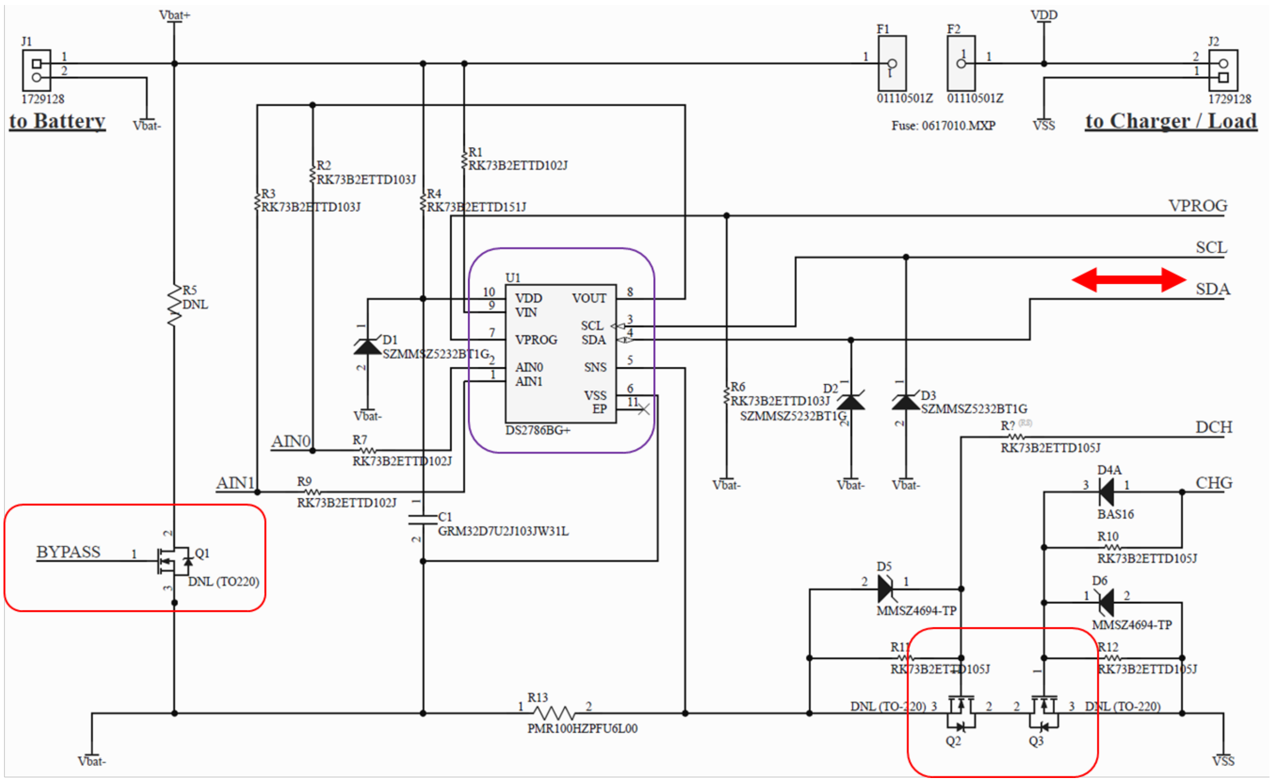
ADDRESS	DESCRIPTION	READ/WRITE
00h	Reserved	—
01h	Status/Config Register	R/W
02h	Relative Capacity	R
03h to 07h	Reserved	—
08h	Auxiliary Input 0 MSB	R
09h	Auxiliary Input 0 LSB	R
0Ah	Auxiliary Input 1/ Temperature MSB	R
0Bh	Auxiliary Input 1/ Temperature LSB	R
0Ch	Voltage Register MSB	R
0Dh	Voltage Register LSB	R
0Eh	Current Register MSB	R
0Fh	Current Register LSB	R
10h to 13h	Reserved	—
14h	Initial Voltage MSB	R
15h	Initial Voltage LSB	R
16h	Last OCV Relative Capacity	R
17h	Learned Capacity Scaling Factor	R
18h to 5Fh	Reserved	—
60h to 7Fh	Parameter EEPROM	R/W
80h to FDh	Reserved	—
FEh	Command	R/W
FFh	Reserved	—

Table 4. Parameter EEPROM Memory Block

ADDRESSS	DESCRIPTION	FACTORY VALUE
60h	Current Offset Bias Register	00h
61h	Capacity 1	0Ah
62h	Capacity 2	14h
63h	Capacity 3	32h
64h	Capacity 4	69h
65h	Capacity 5	A0h
66h	Capacity 6	AAh
67h	Capacity 7	B5h
68h	Voltage Breakpoint 0 MSB	A3h
69h	Voltage Breakpoint 0 LSB	20h
6Ah	Voltage Breakpoint 1 MSB	B9h
6Bh	Voltage Breakpoint 1 LSB	50h
6Ch	Voltage Breakpoint 2 MSB	BCh
6Dh	Voltage Breakpoint 2 LSB	10h
6Eh	Voltage Breakpoint 3 MSB	C0h
6Fh	Voltage Breakpoint 3 LSB	20h
70h	Voltage Breakpoint 4 MSB	C4h
71h	Voltage Breakpoint 4 LSB	20h
72h	Voltage Breakpoint 5 MSB	CDh
73h	Voltage Breakpoint 5 LSB	10h
74h	Voltage Breakpoint 6 MSB	CEh
75h	Voltage Breakpoint 6 LSB	F0h
76h	Voltage Breakpoint 7 MSB	D1h
77h	Voltage Breakpoint 7 LSB	40h
78h	Voltage Breakpoint 8 MSB	D5h
79h	Voltage Breakpoint 8 LSB	90h
7Ah	Initial Capacity Scaling	80h
7Bh	OCV Current Threshold	06h
7Ch	OCV dV/dt Threshold	94h
7Dh	I^{2}C Address Configuration*	60h*
7Eh	Learn Threshold	78h
7Fh	User EEPROM	00h

**The factory default 7-bit slave address is 0110110. The upper 3 bits are fixed at 011; the lower 4 bits can be changed by writing the I^{2}C Address Configuration Register as illustrated in Figures 24 and 25.*

- Circuit Diagram for a Cell Controller



- Code into Arduino communicated with two Cell Controllers

```
#include <Wire.h>


/* Constant */
const int dch = 8;      // discharging control signal
const int chg = 9;      // charging control signal
const int by = 10;
const int dch1 = 11;      // discharging control signal
const int chg1 = 12;      // charging control signal
const int by1 = 13;
const float OVth1 = 4190, OVth2 = 4190;  // threshold to prevent overcharge(mV)
const float UVth1 = 2800, UVth2 = 2800;  // threshold to prevent overdischarge(mV)
int t=750;
byte SAdd1 = 54, SAdd2 = 56;          // Slave address for CC
```

```
/* Constant to display data on excel */
int row_excel = 0; // number of lines
int test = 123; //test variable to be passed to Excel
int test_2 = 456; // the second test variable that will be passed to excel
byte rx_byte;

/* Subroutines */
byte ReadIC(byte SAdd, byte MAdd);
long int ReadIC2(byte SAdd, byte MAdd, int i);
float ConvT(long int Data);
long int ConvC(long int Data);
float ConvV(long int Data);
float ConvVNS(long int Data);
char sign(long int data,int i);
void SendIC(byte SAdd, byte MAdd, byte Data);

void setup(){
  Serial.println("CLEARDATA"); // clear excel sheet
  Serial.println("LABEL,Time,Test 1, Test 2, Num Rows"); // column headers
  Wire.setClock(10000);     // set clock frequency at 10kHz
  Wire.begin();        // join i2c bus (address optional for master)
  Serial.begin(9600);  // start serial for output
  /* for cell 1 */
  pinMode(dch,OUTPUT);    // discharging signal set as output
  pinMode(chg,OUTPUT);    // charging signal set as output
  pinMode(by,OUTPUT);    // bypass signal set as output
  digitalWrite(dch,LOW); // discharging is turned off
  digitalWrite(chg,LOW);  // charging is turned off
  digitalWrite(by,LOW);  // bypass is turned off
  /* for cell 2 */
  pinMode(dch1,OUTPUT);    // discharging signal set as output
  pinMode(chg1,OUTPUT);    // charging signal set as output
  pinMode(by1,OUTPUT);    // bypass signal set as output
  digitalWrite(dch1,LOW); // discharging is turned off
  digitalWrite(chg1,LOW);  // charging is turned off
  digitalWrite(by1,LOW);  // bypass is turned off
//  SendIC(0x7D,0b01110000);  // changing slave address


}


void loop(){


/* check the mode with serial input */
  if (Serial.available() > 0) {    // is a character available?
  rx_byte = Serial.read();       // get the character


    switch(rx_byte){
      case '1':
        digitalWrite(dch,LOW); // discharging is turned off
        digitalWrite(chg,HIGH);  // charging is turned on
        Serial.println("Charging mode for cell1");
        break;
      case '2':
        digitalWrite(dch,HIGH); // discharging is turned off
        digitalWrite(chg,LOW);  // charging is turned on
        Serial.println("Discharging mode for cell1");
        break;
      case '3':
        digitalWrite(dch,LOW); // discharging is turned off
        digitalWrite(chg,LOW);  // charging is turned on
        Serial.println("Open-circuit mode for cell1");
        break;
      case '4':
        digitalWrite(dch,HIGH); // discharging is turned off
        digitalWrite(chg,HIGH);  // charging is turned on
        Serial.println("Short-circuit mode for cell1");
        break;
      case '5':
        digitalWrite(dch1,LOW); // discharging is turned off
        digitalWrite(chg1,HIGH);  // charging is turned on
        Serial.println("Charging mode for cell2");
        break;
      case '6':
        digitalWrite(dch1,HIGH); // discharging is turned off
        digitalWrite(chg1,LOW);  // charging is turned on
        Serial.println("Discharging mode for cell2");
        break;
      case '7':
        digitalWrite(dch1,LOW); // discharging is turned off
        digitalWrite(chg1,LOW);  // charging is turned on
        Serial.println("Open-circuit mode for cell2");
        break;
      case '8':
        digitalWrite(dch1,HIGH); // discharging is turned off
        digitalWrite(chg1,HIGH);  // charging is turned on
        Serial.println("Short-circuit mode for cell2");
        break;
      default:        // remaining the previous mode
        break;
```

```
   }

  }


/* variables */


  int recapa1, initSF1, learnedSF1, OCVth1, dVdtth1, capa11, capa12, capa13, capa14, capa15,
capa16, capa17, LastOCV1;

  float readinitvolt1, v10, v11, v12, v13, v14, v15, v16, v17, v18;

  int recapa2, initSF2, learnedSF2, OCVth2, dVdtth2, capa21, capa22, capa23, capa24, capa25,
capa26, capa27, LastOCV2;

  float readinitvolt2, v20, v21, v22, v23, v24, v25, v26, v27, v28;




/* this is to read temperature with a internal sensor */


  long int temp1, temp2;

  float readtemp1, readtemp2;


  temp1 = ReadIC2(SAdd1,0x0A,5);

  temp2 = ReadIC2(SAdd2,0x0A,5);

  readtemp1 = ConvT(temp1);

  readtemp2 = ConvT(temp2);




/* this is to read voltage measurement */


  long int volt1, volt2;

  float readvolt1, readvolt2;

  char sv1,sv2;


  volt1 = ReadIC2(SAdd1,0x0C,3);
  sv1 = sign(volt1,13);

  readvolt1 = ConvV(volt1);

  volt2 = ReadIC2(SAdd2,0x0C,3);

  sv2 = sign(volt2,13);

  readvolt2 = ConvV(volt2);


/* this is to read currnt measurement */


  long int amp1, amp2;

  long int readamp1, readamp2;

  char sa1, sa2;


  amp1 = ReadIC2(SAdd1,0x0E,4);

  sa1 = sign(amp1,12);

  readamp1 = ConvC(amp1)*25;

  amp2 = ReadIC2(SAdd2,0x0E,4);

  sa2 = sign(amp2,12);

  readamp2 = ConvC(amp2)*25;


/* This is to read each register */


  recapa1 = ReadIC(SAdd1,0x02);

  recapa2 = ReadIC(SAdd2,0x02);


  initSF1 = ReadIC(SAdd1,0x7A);

  learnedSF1 = ReadIC(SAdd1,0x17);

  OCVth1 = ReadIC(SAdd1,0x7B);

  dVdtth1 = ReadIC(SAdd1,0x7C);

  LastOCV1 = ReadIC(SAdd1,0x16);

  capa11 = ReadIC(SAdd1,0x61);
```

```
capa12 = ReadIC(SAdd1,0x62);
capa13 = ReadIC(SAdd1,0x63);
capa14 = ReadIC(SAdd1,0x64);
capa15 = ReadIC(SAdd1,0x65);
capa16 = ReadIC(SAdd1,0x66);
capa17 = ReadIC(SAdd1,0x67);
v10 = ConvVNS(ReadIC2(SAdd1,0x68,4));
v11 = ConvVNS(ReadIC2(SAdd1,0x6A,4));
v12 = ConvVNS(ReadIC2(SAdd1,0x6C,4));
v13 = ConvVNS(ReadIC2(SAdd1,0x6E,4));
v14 = ConvVNS(ReadIC2(SAdd1,0x70,4));
v15 = ConvVNS(ReadIC2(SAdd1,0x72,4));
v16 = ConvVNS(ReadIC2(SAdd1,0x74,4));
v17 = ConvVNS(ReadIC2(SAdd1,0x76,4));
v18 = ConvVNS(ReadIC2(SAdd1,0x78,4));
readinitvolt1 = ConvV(ReadIC2(SAdd1,0x14,3));        // read initial voltage


initSF2 = ReadIC(SAdd2,0x7A);
learnedSF2 = ReadIC(SAdd2,0x17);
OCVth2 = ReadIC(SAdd2,0x7B);
dVdtth2 = ReadIC(SAdd2,0x7C);
LastOCV2 = ReadIC(SAdd2,0x16);
capa21 = ReadIC(SAdd2,0x61);
capa22 = ReadIC(SAdd2,0x62);
capa23 = ReadIC(SAdd2,0x63);
capa24 = ReadIC(SAdd2,0x64);
capa25 = ReadIC(SAdd2,0x65);
capa26 = ReadIC(SAdd2,0x66);
capa27 = ReadIC(SAdd2,0x67);
v20 = ConvVNS(ReadIC2(SAdd2,0x68,4));
```

```
  v21 = ConvVNS(ReadIC2(SAdd2,0x6A,4));
  v22 = ConvVNS(ReadIC2(SAdd2,0x6C,4));
  v23 = ConvVNS(ReadIC2(SAdd2,0x6E,4));
  v24 = ConvVNS(ReadIC2(SAdd2,0x70,4));
  v25 = ConvVNS(ReadIC2(SAdd2,0x72,4));
  v26 = ConvVNS(ReadIC2(SAdd2,0x74,4));
  v27 = ConvVNS(ReadIC2(SAdd2,0x76,4));
  v28 = ConvVNS(ReadIC2(SAdd2,0x78,4));
  readinitvolt2 = ConvV(ReadIC2(SAdd2,0x14,3));        // read initial voltage



/* display measurements on EXCEL sheet */


  row_excel++; // row number + 1
//  Serial.print("DATA,TIME,"); // excel record current date and time
  Serial.print(float(millis())/1000);
  Serial.print(",Cell1,");
  Serial.print(readtemp1,3);
  Serial.print(",");
  Serial.print(sv1);
  Serial.print(readvolt1);
  Serial.print(",");
  Serial.print(sa1);
  Serial.print((float)readamp1/6000.0);
  Serial.print(",");
  Serial.print((float)recapa1/2.0);
//  Serial.print(",");
//  Serial.print(readinitvolt1);
//  Serial.print(",");
//  Serial.print((float)LastOCV1/2.0);
```

```
  Serial.print(",");
  Serial.print(initSF1);
//  Serial.print(",");
//  Serial.print(learnedSF1);
  Serial.print(",OCVth:");
  Serial.print(OCVth1);
  Serial.print(",dV/dt:");
  Serial.print(dVdtth1);
//  Serial.print(",");
//  Serial.println(t);
//  Serial.print(",");
//  Serial.print(0,1);
//  Serial.print("% / ");
//  Serial.print(v10);
//  Serial.print(",");
//  Serial.print((float)capa11/2.0,1);
//  Serial.print("% / ");
//  Serial.print(v11);
//  Serial.print(",");
//  Serial.print((float)capa12/2.0,1);
//  Serial.print("% / ");
//  Serial.print(v12);
//  Serial.print(",");
//  Serial.print((float)capa13/2.0,1);
//  Serial.print("% / ");
//  Serial.print(v13);
//  Serial.print(",");
//  Serial.print((float)capa14/2.0,1);
//  Serial.print("% / ");
//  Serial.print(v14);
//  Serial.print(",");
//  Serial.print((float)capa15/2.0,1);
//  Serial.print("% / ");
//  Serial.print(v15);
//  Serial.print(",");
//  Serial.print((float)capa16/2.0,1);
//  Serial.print("% / ");
//  Serial.print(v16);
//  Serial.print(",");
//  Serial.print((float)capa17/2.0,1);
//  Serial.print("% / ");
//  Serial.print(v17);
//  Serial.print(",");
//  Serial.print(100,1);
//  Serial.print("% / ");
//  Serial.print(v18);
//  Serial.print(",");
//  Serial.print("Cell2,");
//  Serial.print(readtemp2,3);
//  Serial.print(",");
//  Serial.print(sv2);
//  Serial.print(readvolt2);
//  Serial.print(",");
//  Serial.print(sa2);
//  Serial.print((float)readamp2/6000.0);
//  Serial.print(",");
//  Serial.print((float)recapa2/2.0);
//  Serial.print(",");
//  Serial.print(readinitvolt2);
//  Serial.print(",");
//  Serial.print((float)LastOCV2/2.0);

//  Serial.print(",");

//  Serial.print(initSF2);

//  Serial.print(",");

//  Serial.print(learnedSF2);

//  Serial.print(",OCVth:");

//  Serial.print(OCVth2);

//  Serial.print(",dV/dt:");

//  Serial.print(dVdtth2);

//  Serial.print(",");

//  Serial.print(0,1);

//  Serial.print("% / ");

//  Serial.print(v20);

//  Serial.print(",");

//  Serial.print((float)capa21/2.0,1);

//  Serial.print("% / ");

//  Serial.print(v21);

//  Serial.print(",");

//  Serial.print((float)capa22/2.0,1);

//  Serial.print("% / ");

//  Serial.print(v22);

//  Serial.print(",");

//  Serial.print((float)capa23/2.0,1);

//  Serial.print("% / ");

//  Serial.print(v23);

//  Serial.print(",");

//  Serial.print((float)capa24/2.0,1);

//  Serial.print("% / ");

//  Serial.print(v24);

//  Serial.print(",");
//  Serial.print((float)capa25/2.0,1);
//  Serial.print("% / ");
//  Serial.print(v25);
//  Serial.print(",");
//  Serial.print((float)capa26/2.0,1);
//  Serial.print("% / ");
//  Serial.print(v26);
//  Serial.print(",");
//  Serial.print((float)capa27/2.0,1);
//  Serial.print("% / ");
//  Serial.print(v27);
//  Serial.print(",");
//  Serial.print(100,1);
//  Serial.print("% / ");
//  Serial.print(v28);
  Serial.print(",");
  Serial.println(row_excel);


/* if rows are more than 5000, then start filling the rows again */
  if (row_excel > 5000){
    row_excel = 0;
    Serial.println("ROW,SET,2");
  }


/* function to prevent overdischarge */
  if(readvolt1<UVth1){      // comparing terminal voltage and threshold
    digitalWrite(dch,LOW); // discharging is turned off
    digitalWrite(chg,LOW);  // charging is turned off
    Serial.println("Open-circuit mode for Cell1");
  }
```

```
 if(readvolt2<UVth2){
   digitalWrite(dch,LOW); // discharging is turned off
   digitalWrite(chg,LOW);  // charging is turned on
   Serial.println("Open-circuit mode2");
 }

/* function to prevent overcharge */
 if(readvolt1>OVth1){      // comparing terminal voltage and threshold
   digitalWrite(dch,LOW); // discharging is turned off
   digitalWrite(chg,LOW);  // charging is turned off
   Serial.println("Open-circuit mode for Cell1");
 }

 if(readvolt2>OVth2){
   digitalWrite(dch1,LOW);
   digitalWrite(chg1,LOW);
   Serial.println("Open-circuit mode for Cell2");
 }

 delay(100); //delay
}



/**********************SUBROUTINS**********************************/

/* Subroutine for reading Memorys which are one byte */
byte ReadIC(byte SAdd, byte MAdd){
 int ReadData;
```

```
  Wire.beginTransmission(SAdd); // transmit to a slave device
  Wire.write(MAdd);             // write the memory address byte
  Wire.endTransmission(false);  // send the restart condition
  delay(50);

  Wire.requestFrom(SAdd, 1);    // request 1 bytes from slave device
  while(Wire.available()){      // read 1byte data
    ReadData = Wire.read();
  }
  return ReadData;              // return 1byte data
}

/* Subroutine for reading Memories which are 2 bytes */
long int ReadIC2(byte SAdd, byte MAdd, int i){
  long int ReadData;
  long int Data[2];

  Wire.beginTransmission(SAdd); // transmit to a slave address
  Wire.write(MAdd);             // write the memory address byte
  Wire.endTransmission(false);  // send the restart condition
  delay(50);

  Wire.requestFrom(SAdd, 2);    // request 2 bytes from slave device #54
  int n = 0;
  while(Wire.available()){    // read 2 bytes data
    Data[n] = Wire.read();
    n++;
  }
  Data[0] = Data[0] << (8-i);
  Data[1] = Data[1] >> i;
  ReadData = Data[0] | Data[1];
  return ReadData;            // return 2 bytes data
}



/* This is to read temperature */

float ConvT(long int Data){

  long int mask, t;
  float readt;
  int m;

  mask = 1;
  t = 0;
  if(Data & 0b10000000000){       // convert data for two's complement form
    for(m=0;m<11;m++){
      if(!(Data & mask)) t = t + mask;
        mask = mask << 1;
    }
  }
  else t = Data;
  t = t & 0b1111111111;           // mask 10 bits
  readt = (float)t * 0.125;       // convert temerature at 0.125 degree per bit

  return readt;
}

/* This is to read current */
```

```
long int ConvC(long int Data){

  long int mask, c;
  int m;

  mask = 1;
  c = 0;
  if(Data & 0b100000000000){    // convert data for two's complement form
    for(m=0;m<12;m++){
      if(!(Data & mask)) c = c + mask;
        mask = mask << 1;
    }
  }
  else c = Data;
  c = c & 0b11111111111;       // mask 11 bits

  return c;
}

/* This is to read voltage */

float ConvV(long int Data){

  long int mask, v;
  float readv;
  int m;

  mask = 1;
  v = 0;
  if(Data & 0b1000000000000){   // convert data for twe's complement form
```

```
    for(m=0;m<13;m++){
      if(!(Data & mask)) v = v + mask;
        mask = mask << 1;
    }
  }
  else v = Data;
  v = v & 0b111111111111;        // mask 12 bits
  readv = (float)v*1.22;         // convert voltage at 1.22mV per bit

  return readv;
}

/* This is to read voltage without sign */

float ConvVNS(long int Data){

  long int v;
  float v1;

  v = Data;       // conbine MSB and LSB

  v = v & 0b111111111111;        // mask 12 bits
  v1 = (float)v*1.22;         // convert voltage at 1.22mV per bit

  return v1;
}



/* check sign */
char sign(long int data,int i){   // i is the bit number of digits
```

```
  char fc;
  data = data >>(i-1);           // shift sign bit to first bit
  if(data & 1){                  // check sign bit
    fc = '-';                    // if sign bit is 1, sign is negative
    return fc;
  }
  else{
    fc = '+';                    // if sign bit is 0, sign is positive
    return fc;
  }
}


/* Subroutine for writting Data which are one byte */
void SendIC(byte SAdd, byte MAdd, byte Data){
  Wire.beginTransmission(SAdd); // transmit to a slave device
  Wire.write(MAdd);       // write the memory address byte
  Wire.write(Data);       // write the data
  Wire.endTransmission();  // send the restart condition
  delay(50);
}
```

[**Notes on the template:** Please note that this template is a guide only and may be modified to suit the individual requirements of different disciplines and theses. You may modify the fonts and formatting of the headings to suit your personal style, taking note that this is a professional document.

This template has been set up for single sided printing. If you wish to print double sided, you will need to adjust the margins to mirrored margins. Digital theses may not require the same margin settings.]