A Prototype for Estimating State-of-Charge in the Battery-powered Mobile System

Minsu Oh[1] and HyunJin Kim[1a]
[1]School of Electronics and Electrical Engineering, Dankook University
[a]E-mail: hyunjin2.kim@gmail.com

Abstract - **This paper proposes a method for estimating state-of-charge (SOC) using the current error compensation technique in the battery-powered mobile system. In this paper, with the initial information for the lithium-ion battery, the measured current is compensated. In the target battery powered system, SOC is calculated using the Coulomb counting. Based on the experimental results, the estimated SOCs using open-circuit voltage (OCV) and the Coulomb counting are compared. By referencing the SOC with the OCV, the error compensation equation can be provided by adopting the regression analysis. Experimental results show that the error is under 2% on average. Compared to the traditional Coulomb counting with 7% error, the proposed error compensation technique can enhance the quality of the SOC estimation.**

I. INTRODUCTION

Accurate SOC estimation system is essential to the battery powered mobile system. SOC is related to the battery life or capacity; therefore, inaccurate SOC estimation makes a battery over-charged or over-discharged. In these cases, the battery reliability or healthy condition can be damaged [1]. Therefore, more accurate SOC estimation should be designed.

This paper proposes a method for estimating SOC using the current error compensation technique in the battery-powered mobile system. The value of SOC is an essential information in the battery management system (BMS). For the battery-powered mobile system, the traditional Coulomb counting method can be easily implemented by calculating the amount of charges consumed in a battery. When measuring current from the battery, if the measured current is not correct, the accumulated error degrades the accuracy for estimating SOC [1].

In this paper, with the initial information for the lithium-ion battery, the measured current is compensated. In the target battery powered system, SOC is calculated using the Coulomb counting. Based on the experimental results, the estimated SOCs using open-circuit voltage and the Coulomb counting are compared. By referencing the SOC with the OCV, the error compensation equation can be provided by adopting the regression analysis. In order to show practical e

xperimental environments, the proposed method is implemented with Dongbu 0.11μm semiconductor process. In the implemented chip, ARM Cortex-M0 DesignStart [2] and many bus IPs are implemented in a chip. Experimental results show that the error is under 2% on average. Compared to the traditional Coulomb counting with 7% error, the proposed error compensation technique can enhance the quality of the SOC estimation.

II. ESTIMATION OF SOC IN A BATTERY

In this section, the general characteristics of Lithium-ion battery and well-known techniques for estimating SOC are explained. For high powered mobile system, Lithium ion or Lithium Polymer batteries are being adopted prevalently. Therefore, the basic characteristics of Lithium ion battery are shown in the following subsection.

A. Lithium ion battery

From 1990s, Lithium ion battery is invented and commercially produced. The Lithium ion battery is one of the rechargeable batteries, where several chemical materials are used in anode and cathode. In charging, Lithium ions are transferred from anode to cathode; when discharging, the transfer from cathode and anode happens. According to the materials used in electrolyte, anode, and cathode, the output voltage, capacity, and lifetime of a battery can be influenced.

Especially, because the Lithium ion battery has great energy density and high electromotive force, the size of battery can be reduced, compared to other kinds of batteries.

B. Estimation of SOC

Because it is impossible to extract the information of the internal state of a battery directly, the residual charge is estimated using the output voltage, current, and temperature in the battery. The ratio of the residual charge to the rating charge in a battery is defined as SOC. When the battery is fully charged, SOC is considered as 100%; when the battery is fully discharged, SOC can be 0%. In order to estimate SOC, there is need to adopt several sensors. For example of [1], voltage, current, and temperature sensors are equipped

a. Corresponding author; hyunjin2.kim@gmail.com

Copyright ©2017 IDEC All rights reserved.
This is an Open-Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (http://creativecommons.org/licenses/by-nc/3.0) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.

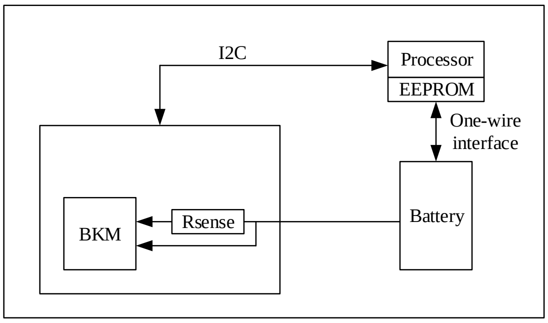

Fig. 1. Diagram of a book-keeping system.

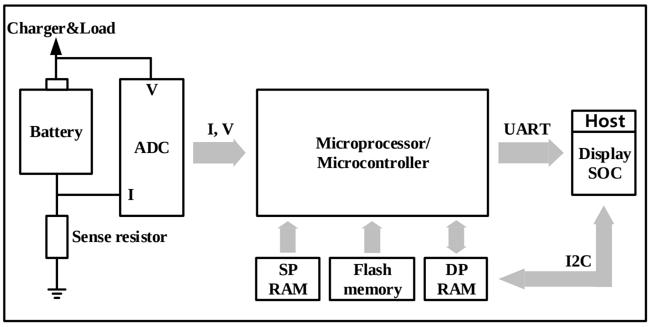

Fig. 2. Proposed architecture for estimating SOC.

in the estimation. In addition, after converting the analog values using the analog-to-digital converters, the calculation is performed in the microprocessor. Considering the equipped sensors and applied algorithms, several methods have been developed. Especially, there are two basic methods called the Coulomb counting and OCV.

The Coulomb counting gathers the information of current flow in a battery [1] [3]. The accumulation of the current flow from the battery can be the amount of the consumed charge. The system can be modeled as the book-keeping system. Figure 1 shows the diagram of a book-keeping system. In Fig. 1, the book-keeping machine (BKM) gathers the information of current flow using the current sensor resistor denoted as Rsense. In addition, other information can be gathered from the target battery and transferred to the processor. In Fig. 1, one-wire interface is adopted to transfer the information to the processor. In addition, the information from the BKM can be transferred using I2C bus. By subtracting the consumed charge from the residual charge, the amount of the residual charge can be updated. In this case, SOC can be formulated by:

$$SoC = SoC_0 - \frac{1}{C_N}\int I_L\, dt \ , \qquad (1)$$

where SoC_0, C_N, and i_L denote the initial value of SOC, full battery capacity, and estimated current flow, respectively. In this case, the current sensor is an essential device to estimate the accumulated value. The error of the current sensing, therefore, is critical to the accuracy of the estimation. In addition, due to the discontinuity of the current sensing, the Coulomb counting cannot be the realistic solution in the estimation of SOC.

The OCV method adopts the stable voltage output from a battery when the current flow is negligible for a long time. When there is no current flow, it is known that the voltage output can be fixed according to the residual charge. Even though the full capacity can be changed with the number of charging cycles and temperature, the voltage output for the SOC is not changed. However, because long idle time is required, when current flow is not negligible, the estimation cannot be correct somewhat.

Considering the weakness of the Coulomb counting and OCV method, a lot of techniques have been studied using the linear model, impedance analysis, artificial neural networks [4], fuzzy logic [5], Kalman filter [6], etc. Even though the techniques mentioned above require large amount of computations, which can be great burden in the mobile system. For example, a fuel gauge IC is in charge of estimating SOC in a mobile system. When the system is idle, the fuel gauge IC should operate to calculate SOC. Even though there is no need to show SOC frequently, large computations require inconvincible energy consumption. Therefore, the simple method for amortizing the weakness of the Coulomb counting and OCV methods is required, where the realistic implementation should not be complex.

III. PROPOSED METHOD AND ARCHITECUTURE

Considering the weakness of the weakness of the Coulomb counting and OCV methods, these two methods are combined. The OCV method provides the initial SOC in the proposed method. In addition, the compensation of current error reduces the accumulated errors to provide the reliable current estimation.

A. Overview of proposed method and architecture

In the Coulomb counting, the initial SOC is the starting point, where the error from the initial SOC can influence the estimation. The OCV method is applied to alleviate the influence of the error in the Coulomb counting when the current flow is negligible. The main reason why the estimation error in the current flow exists is the architecture of the current sensor that adopts the current resistor and ADC (analog-to-digital converter). The main concept of the proposed method can be formulated by (2) and (3) as follows:

$$SoC_n = SoC_i - \frac{1}{C_N}\int {i'}_L\, dt \ , \qquad (2)$$

$$i'_L(t) = i_L(t) - error \ , \qquad (3)$$

where SoC_n SoC_i, C_N, $i_L^i(t)$, and $i_L(t)$ denote the current SOC, the initial value of SOC, full battery capacity, current after compensation, and current before compensation, respectively. In addition, $error$ denotes the compensation function of current error.

Figure 2 shows the proposed architecture for estimating SOC that is implemented using a semiconductor process. In order to estimate OCV and current flow, monolithic ADCs are adopted. The values of current and voltage are transferred into microcontroller. In the microcontroller, SPRAM (single port RAM) and DPRAM (dual port RAM) are used to store the program for calculating SOC and battery data from ADC and battery intrinsic characteristics, respectively. In the Flash

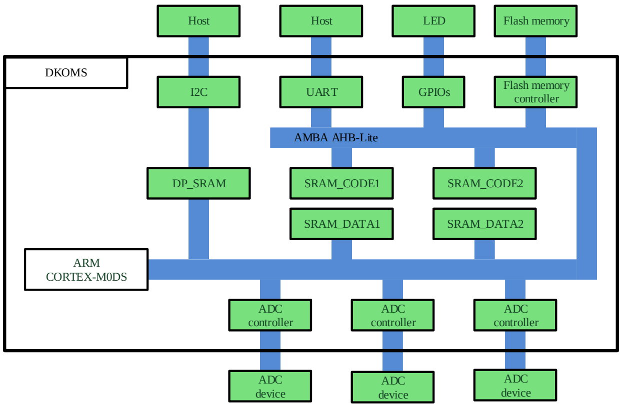

Fig. 3. Architecture of prototype IC.

TABLE I
Memory map of prototype IC.

Before Memory Remapping		After Memory Remapping	
REG	0x59FFFFFF	REG	0x59FFFFFF
	0x59000000		0x59000000
GPIO	0x58FFFFFF	GPIO	0x58FFFFFF
	0x58000000		0x58000000
Timer	0x57FFFFFF	Timer	0x57FFFFFF
	0x57000000		0x57000000
ADC3	0x54FFFFFF	ADC3	0x54FFFFFF
	0x54000000		0x54000000
ADC2	0x53FFFFFF	ADC2	0x53FFFFFF
	0x53000000		0x53000000
ADC1	0x52FFFFFF	ADC1	0x52FFFFFF
	0x52000000		0x52000000
UART	0x510000FF	UART	0x510000FF
	0x51000000		0x51000000
Special Register(I^2C)	0x310000FF	Special Register(I^2C)	0x310000FF
	0x31000000		0x31000000
SRAM(code)2	0x30007FFF		
	0x30004000		
SRAM(code)1	0x30003FFF	SRAM(data)2	0x20007FFF
	0x30000000		0x20004000
SRAM(data)2	0x20007FFF	SRAM(data)1	0x20003FFF
	0x20004000		0x20000000
SRAM(data)1	0x20003FFF	SRAM(code)2	0x00007FFF
	0x2000000		0x00004000
SPI Flash	0x00007FFF	SRAM(code)1	0x00003FFF
	0x00000000		0x00000000

memory, the program for calculating SOC is initially stored. In order to provide the calculated SOC into the host, UART (universal asynchronous receiver/transmitter) and I2C interconnections are used.

B. Regression analysis for compensating current error

The regression analysis can estimate the relationships among variables in statistical modeling. Using the observed data, the scatter diagram is obtained. The degree of scattering is adopted to apply the curve fitting for getting the regression curve. By applying the mean squared error, the regression curve can be obtained in general [7].

In this paper, unlike the previous method using the look up table, the equation from the regression curve is adopted in order to estimating the residual charge. Because there is no need to store the data into the flash memory, the memory requirement can be greatly reduced. In other words, the implementation area or cost can be minimized. In our approach, MATLAB is adopted, where the polyfit function [8] is used to obtain the 4-th order equation in the proposed method.

IV. IMPLEMENTATION OF PROTOTYPE

In order to prove the effectiveness of the proposed method, the silicon-based prototype is implemented using ARM-based microcontroller. Figure 3 shows the detail architecture of the implemented prototype. The blocks inside large black box are implemented in the IC prototype. ARM Cortex-M0 DesignStart [2] is adopted for the microcontroller. In addition, SPRAM, DPRAM, ADC controller, I2C controller, UART controller, GPIO, and flash memory controller are implemented. These hardware blocks are communicated with the AMBA (Advanced Microcontroller Bus Architecture) AHB (Advanced High Performance)-Lite bus protocol [9]. In addition, monolithic ADC ICs are mounted in the evaluation board. Therefore, the prototype has the full digital IC, where the analog hardware blocks are not included in the IC prototype.

TABLE I shows the memory map of the implemented prototype IC. After downloading program in the flash memory into RAM, the memory remapping is performed. After remapping the memory map, the program in SRAM is executed. Therefore, the instruction can be fetched each clock cycle from SRAM.

The implementation of the prototype IC adopts the Dongbu 0.11um semiconductor process and its library. Figure 4 shows the design process of the prototype IC. In the front-end, the test code is tested using Keil µVision 4 compiler and debugger. Then, the test code is compiled and translated into the hex file, which is used in the functional simulation. After performing logic synthesis, equivalence check, static timing analysis, and pre layout simulation, the gate-level netlist is obtained. Using the netlist, the back-end process is also performed. The test code can be translated into binary file to initialize the flash memory. The program is coded considering CMSIS (Cortex Microcontroller Software Interface Standard) [10], where the basic template of CMSIS is used.

TABLE II shows the implementation results of the prototype IC, which is packaged in a form of QFN46. In addition, the number of used gates is 460K. The die size is fixed and not minimized because the MPW program for the semiconductor process provided the fixed sized die. The core and IO voltages are 1.2V and 3.3V, respectively. Figure 5 shows the micrograph of the prototype IC. There are four SRAM blocks placed in the boundary. One DPSRAM is located near I2C controller block. There are 40 IOs, where 12 IOs are used for power sources. Using Design Compiler tool, total dynamic power consumptions are expected as 75mW. In addition, the leakage power consumption can be 0.013mW.

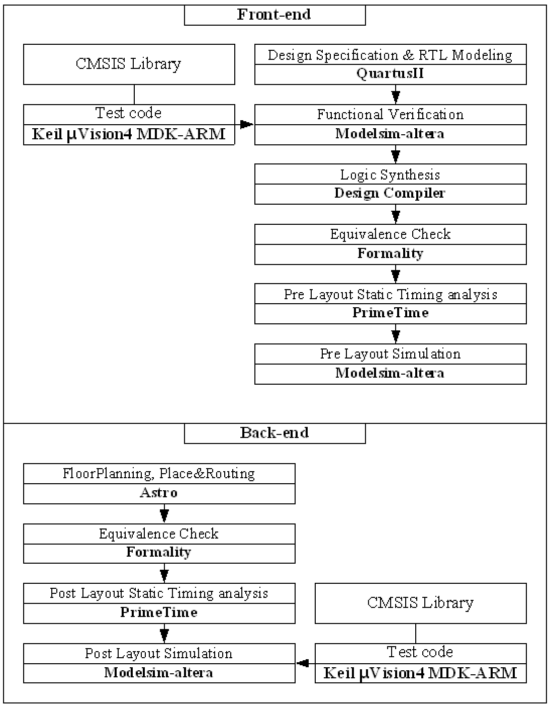

Fig. 4. Design process of prototype IC.

TABLE II.
Implementation result of prototype IC.

Specification	Result
Gate count	459,823
Target operating frequency	50M [Hz]
Core/ IO voltages	3.3/1.2 [V]
Single port SRAM	64KB X 4
Dual port SRAM	4KB
Die size	2350 [mm] x 2350 [mm]
Package	QFN 46

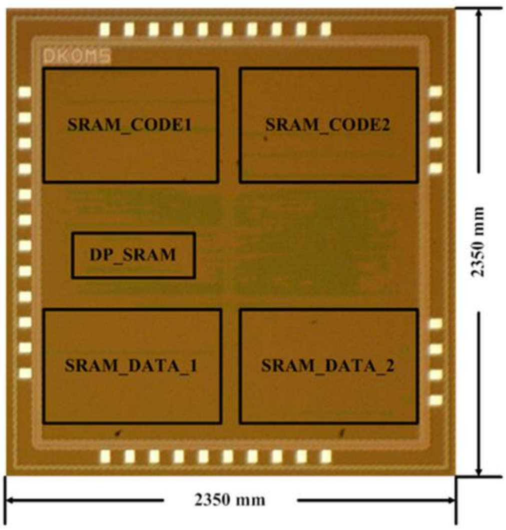

Fig. 5. Micrograph of prototype IC.

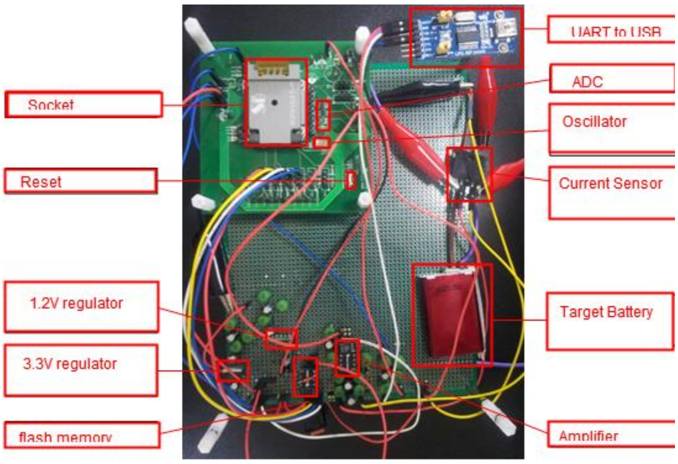

Fig. 6. Evaluation board of prototype IC.

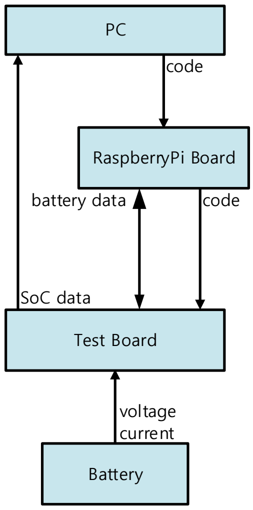

Fig. 7. Connection for evaluation board.

An evaluation board is made to test the prototype IC and its application. Figure 5 show the photo of the evaluation board of the prototype IC. In the test socket, the prototype IC is equipped. The power of 1.2V and 3.3V was provided from the monolithic regulator ICs. The external flash memory can be accessed using SPI (Serial Peripheral Interface) bus with the prototype IC. The precise current sensor resistor with 50mΩ is adopted. In order to amplify the voltage level in the current sensor resistor, a current sense amplifier is equipped. The output of the current sense amplifier is inputted into the monolithic ADC.

V. EXPERIMENTS OF SOC ESTIMATION

A. Experimental environments

In this experiment, a target battery of Sanyo UF103450 [11] was adopted. Even though the specification of the target battery provided the table between SOC and OCV, the data between SOC and OCV were measured. In the experiments, the charging and discharging were performed at 1 C rate. The battery was charged and discharged using a power supply and electrical load.

In the experiments, the evaluation board, RaspberryPi [12] board, host computer were connected as shown in Fig. 7.

Firstly, the test code was compiled in the host computer. The binary machine code was transferred into the RaspberryPi board. Then, the machine code was transferred from the RaspberryPi board to the flash memory in the evaluation board using SPI protocol. In addition, the battery data can be transferred between the RaspberryPi board and evaluation board using I2C protocol. In the evaluation board, the machine code in the flash memory was run. The prototype IC in the evaluation board calculated SOC and then transferred the calculated data into the host computer.

B. Source program

In this experiment, a test program was coded and downloaded into the prototype IC. The code below was the main part of the source to estimate SOC. Firstly, the program in the flash memory was copied to the SRAM. When the copy was done, memory remapping was performed. From line 20, several functions were called to estimate SOC. Function ADCdata got 20 values from ADC. In order to reduce the effectiveness of noise, the average value did not consider the two largest and smallest values. Functions CurrentErrorComp and VoltageComp compensated the data from ADCdata.

```
1   int main(void)
2   {
3     for (i = 0x0000; i < 0x1000 ; i++)
4       *(unsigned int*) (AHB_SRAM_CODE + 4*i)
5     = *(unsigned int*) (AHB_FLASH + 4*i);
6     *(unsigned int*) (AHB_REG) = 0x00000001U;
7     delay(1000);
8     returnCode = SysTick_Config(SystemCoreClock / 1000);
9     buffer = 1;
10    if(returnCode != 0)
11    {
12      *(unsigned int*) (AHB_UART) = 0x00000005 + 48;
13      *(unsigned int*) (AHB_UART) = 0x00000005 + 48;
14    }
15    while(1)
16    {
17      if(msTicks > 1000)
18      {
19        msTicks = 0;
20        ADCdata();
21        CurrentErrorComp();
22        VoltageComp();
23        EstSoC();
24        buffer = 0;
25        dectoascii();
26        UARTsend();
27      }
28      else WaitForTick ();
29    }
30  }
```

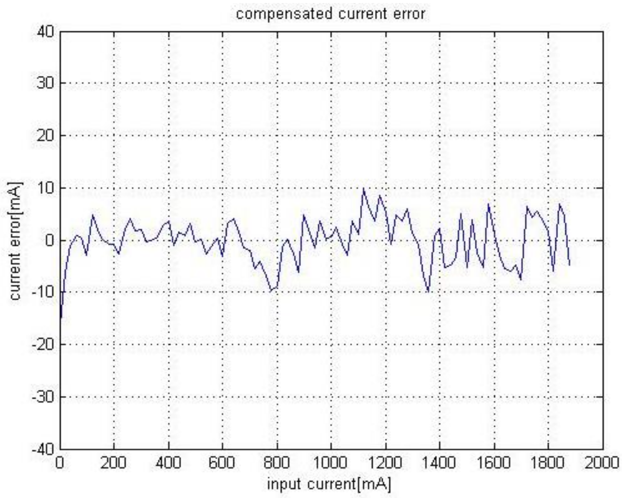

Fig. 8. Compensated current error.

Function EstSoC provided SOC using the compensated current and voltage data. In the current sensing, when the current was under 25 mA, open-circuit voltage was measured. Function dectoascii prepared the data to be transferred into the host by function UARTsend. Function WaitForTick provided the delay of 1 second.

C. Experimental results

For (3), the error was calculated using the polyfit function in MATLAB [8]. In this experiments, the error can be obtained as follows:

$$\text{error (mA)} = 2.271\text{x}10^{-12} i^4 - 2.393\ \text{x}10^{-3} i^3 + 5.170\ \text{x}10^{-5} i^2 - 4.968\ \text{x}10^{-2} i + 17.509 \quad (4)$$

Using (4), the current error can be compensated in Fig. 8. After compensation, the maximum error rate can be reduced into 0.81% using (4), compared to that of 2.82% before the compensation.

TABLE III and IV shows the estimated SOCs that did not compensate errors and did compensate errors using (4), respectively. After charging the target battery fully, the battery was discharged at 1C rate. In each iteration, 5 minutes and 30 minutes were applied for the discharging time and idle time. In Table 3, the rate of error in terms of SOC was 7.790% on average. On the other hand, when the compensation using (4) was applied, the rate of error in terms of SOC was reduced up to 2.529% on average. Therefore, the rate of error was decreased by 5.262%.

VI. CONCLUSION

This paper proposes a method for estimating SOC using the current error compensation technique in the battery-powered mobile system. The error compensation equation can be provided by adopting the regression analysis. The compensation technique is applied to the implemented prototype IC. Experimental results show that the error is under 2% on average. Compared to the traditional Coulomb with 7% error, the proposed error compensation technique can enhance the quality of the SOC estimation. Therefore, it is concluded that the proposed method and prototype IC are helpful to enhance the estimation of SOC.

TABLE III.
Estimation SOCs that did not compensate errors.

Time (Min)	Discharging Current [mA]	Real SCC [%]	Est. SOC [%]	Error [%]
0	0	100	100	0.000
5	156	91.667	83.636	8.761
10	313	83.334	78.182	6.183
15	469	75.001	70.909	5.456
20	626	66.668	61.818	7.275
25	783	58.335	52.727	9.613
30	939	50.002	47.273	5.458
35	1096	41.669	38.182	8.369
40	1252	33.336	30.909	7.280
45	1409	25.003	23.636	5.466
50	1565	16.670	18.182	9.069
55	1722	8.337	7.273	12.766
60	1880	0.000	0.000	0.000

TABLE IV.
Estimation SOCs that did not compensate errors.

Time (Min)	Discharging Current [mA]	Real SCC [%]	Est. SOC [%]	Error [%]
0	0	100	100	0.000
5	156	91.667	92.727	1.157
10	313	83.334	81.818	1.819
15	469	75.001	72.727	3.032
20	626	66.668	67.273	0.907
25	783	58.335	56.364	3.379
30	939	50.002	50.909	1.814
35	1096	41.669	40.000	4.005
40	1252	33.336	32.727	1.826
45	1409	25.003	24.164	3.356
50	1565	16.670	16.364	1.838
55	1722	8.337	8.727	4.681
60	1880	0.000	0.000	0.000

ACKNOWLEDGMENT

This work was supported by the IDEC.

REFERENCES

[1] Pop, V., Bergveld, “Battery Management Systems,” vol. 9. Springer, 2002.
[2] ARM Ltd., ARM Cortex-M0 DesignStart Processor and v6-M Architecture, 2010.
[3] H.J. Bergveld, W.S. Kruijt, and P.H.L. Notten, “Battery Management Systems, Design by Modelling,” Philips Research Book Series. Kluwer Academic Publishers, 2002.
[4] O. Gerard, J.N. Patillon, F. d’Alche-Buc, “Neural Network Adaptive Modelling of Battery Discharge Behavior,” Lect. Notes Comput. Sci., vol. 1327, pp.1095–1100, 1997.
[5] A.J. Salkind, et al., “Determination of State-of-Charge and State-of-Health of Batteries by Fuzzy Logic Methodology,” J. Power Sources, vol. 80, pp.293–300, 1999.
[6] He Hongwen, et al., “State of Charge Estimation of the Lithium-ion Battery Using an Adaptive Extended Kalman Filter based on an improved Thevenin model,” IEEE Transactions on Vehicular Technology, 2011.
[7] Oliver C., “Fundamentals of Applied Probability and Random Processes,” Elsevier, 2008.
[8] Polynomial curve fitting, MATLAB function, Available: https://kr.mathworks.com/help/matlab/ref/polyfit.html?requestedDomain=www.mathworks.com.
[9] ARM IHI 0033A AMBA 3 AHB-Lite Protocol V.1 Specification, ARM LTD., 2006
[10] Cortex Microcontroller Software Interface Standard, ARM co., Available: https://www.arm.com/products/processors/cortex-m/cortex-microcontroller-software-interface-standard.php.
[11] Sanyo UF103450, Sanyo co., Avaliable: http://www.meircell.co.il/files/Sanyo%20UF103450P.pdf.
[12] Raspberry Pi , Available: https://www.raspberrypi.org.

Min-Su Oh received the B.S. and M.S degrees in electronics and electrical engineering from Dankook University, Yongin-si, Republic of Korea, in 2012 and 2015, respectively. Now, he works as digital circuit designer in ICTK.

HyunJin Kim received B.S., M.S., and Ph. D degrees in Department of Electrical and Electronic Engineering from Yonsei University, Seoul, Republic of Korea, in 1997, 1999, and 2010, respectively. In 2002-2004 and 2010-2011, he worked in the R&D center of Samsung ElectroMechanics and the Memory Division of Samsung Electronics in the field of circuit design and implementation. From 2011, he is the assistant professor in School of Electronics and Electrical Engineering, Yongin-si, Republic of Korea. His interests include parallel & embedded systems, network virtualization, pattern matching engine, and IoT (Internet of Thing) devices.