Adequacy of Data Sensing Frequency for Mitigation Decision in Drone Firmware

Hoijin Yoon
Department of Computer Science and Engineering, Hyupsung University
14 Bongdam, Hwaseung, Kyunggi, South Korea
hjyoon@uhs.ac.kr

Abstract—To ensure a failsafe flight of a drone, adequate mitigation is conducted for accurate detection of the battery discharge level and problematic situations that may cause the drone to fall below a certain level. On one hand, if the mitigation starting time is too late, the drone would fall down before finishing failsafe landing, which is against the safety requirements. On the other hand, if the mitigation starting time is too early, the drone will still have some battery power left when it finishes landing, which is in contradiction to the efficient use of the battery resources. Therefore, in this study, we analyze how the mitigation starting time changes according to the frequency and aim to experimentally discover an adequate level of data sensing frequency. The experiment targets a 3DRobotics product, Pixhawk firmware, which controls a quadcopter, and utilizes a battery model developed mathematically. Experimental results revealed that the mitigation starting time was almost identically maintained for the frequency of certain domains and delayed rapidly for the low-frequency domain.

Keyword- Safety, Efficiency, Drone, Failsafe Mitigation, Sensing Frequency

I. INTRODUCTION

On October 13, 2014, the Linux Foundation announced the Drone Code Project for the development of unmanned aerial vehicles (UAVs) [1]. Major companies in the fields of drones, communication, and computing, such as 3DRobotics, Qualcomm, and Intel, participated in the Drone Code Project, and as a result, drones have passed the research level and have reached a level wherein they can be used in various applications such as commercial functions, architecture, nature conservation, and rescue activities. On March 1, 2015, 3DRobotics, a US-based drone company, announced an investment plan of approximately $50 million to secure a technology that can control drones by using mobile phones with built-in Snapdragon developed by Qualcomm, a global wireless communication company and mobile-phone CPU manufacturer [2]. As such, drones are increasingly becoming a part of our lives, but drone safety problems still remain. According to [3], drones were involved in many crash accidents and most of these accidents were attributed to fuel problems, i.e., battery problems. Because the remaining battery power plays a decisive role in drone operation, when the remaining battery power falls below a certain level, pre-designated mitigation has to be performed. For this, the firmware of the drone collects the battery level data continuously from a sensor, and based on the battery level, determines whether to enter into mitigation.

From the safety aspect, mitigation is a method for ensuring failsafe behaviors [4,5]. In the case of drones, to prevent failures due to battery-related problems, mitigation is performed when the read battery level is below a certain value. In this case, the number of times the current battery level read to determine entry into mitigation is defined as the “frequency” value. In general, it is expected that the accuracy of a decision based on the extracted data will be high when the frequency is high. In the case of drone battery too, if the extraction frequency during flight is low, the mitigation starting time will be too late, which risks failure. If the frequency affects the mitigation starting time at an identical rate, we can conclude that a larger frequency is better. Therefore, in this paper, the following research questions are considered and an experimental analysis is presented to find the answers to these questions:

<RQ 1> Does the mitigation starting time get delayed when the frequency decreases?

<RQ 2> To obtain an appropriate mitigation starting time for failsafe behavior, is it better to increase the data sensing frequency?

<RQ 3> Is there an adequate frequency for setting an appropriate mitigation starting time?

In this study, we analyze whether the data sensing frequency and the mitigation starting time are always the same by analyzing the changes in the effect of frequency improvement on the mitigation starting time. If the effect is not decreased at the same rate as decrease of frequency above a certain threshold value, setting a frequency higher than the threshold will produce an adverse effect of increasing the time and economic costs of sensing. In this study, we have simultaneously considered the safety standpoint of going after a high frequency and the efficiency standpoint of considering the relevant costs and have discussed the adequacy of frequency. For this analysis, we carried out a quasi-experiment using a battery discharge model composed with Matlab. In our experiment, the domain is Pixhawk’s firmware, which controls a quadcopter manufactured by 3DRobotics.

II. Related Works

The battery-related settings of a drone are set using two methods. One method involves hardcoding the settings inside the firmware, and the other uses Mission Planner software [6], which designs the flight path of the drone.

A. Battery Module in Pixhawk Firmware

In a drone, a controller such as Pixhawk[7] is installed along with a conventional RC transmitter for automatic flight. The firmware is installed in this control device to facilitate the autopilot. The firmware is software that provides various types of support during the entire flight and is the system software dependent on the drone machine. The 3DRobotics quadcopter and Pixhawk used in this study are shown in Fig. 1.

Fig. 1. 3DRobotics quadcopter and its Pixhawk used in the experiment

In Pixhawk, a program written in C++ and C# is built in as firmware. We analyzed the source codes [8] and observed how the battery settings-related part and the relevant settings were used. In the program package, sensor.cpp is used for settings related to the sensor, which is the target of this study. The coefficient values, which filter the battery sensing frequency and the measured battery level, are hardcoded in sensor.cpp. Fig. 2 shows a snippet of sensor.cpp for the pertinent part. Pixhawk's firmware sets the sensing frequency as 100 Hz and the filtering coefficient as 0.001.

```
……

#define    BATT_V_LOWPASS     0.001f
….
void
Sensors::adc_poll(struct sensor_combined_s &raw)
{
        /* only read if publishing */
        if (!_publishing) {
                return;
        }

        hrt_abstime t = hrt_absolute_time();

        /* rate limit to 100 Hz */
        if (t - _last_adc >= 10000) {
                /* make space for a maximum of twelve channels (to ensure reading all channels at once) */
                struct adc_msg_s buf_adc[12];
                /* read all channels available */
                int ret = read(_fd_adc, &buf_adc, sizeof(buf_adc));
…..
```

Fig. 2. Snippet of "Sensors.cpp" in Pixhawk firmware

B. Failsafe Settings in Mission Planner

Fig. 3 shows Mission Planner's failsafe setting screen. To ensure the failsafe flight of a drone, the designated mitigation is set to be performed if the battery level below a certain threshold is read. It is denoted by the yellow arrow in the figure. This setting implies that "if the level is less than 10.5, the copter is supposed to Land." The default value of 10.5 in Mission Planner denotes a boundary value at which a mitigation action called "Land" has to start. "Land" is also a basic mitigation action set up in Mission Planner. By using the menu of Mission Planner, a user can set a different value arbitrarily and designate a different mitigation action. The mitigation actions that can be set in Mission Planner are "Land," "RTL," and "Disable." "Land" denotes an immediate landing attempt, "RTL" implies returning to the home position, and "Disable" means to ignore the current status without performing any special mitigation activity.

Fig. 3. Failsafe settings in Mission Planner

Interest in risk mitigation has increased with the emergence of safety as an important issue. In the case of drones, potential risks are managed as shown in Fig. 3. With respect to the various risks of drone flights, the menus shown in Fig. 3 are used to manage responses to problems that may occur when these risks are actually realized. Through this process, the effect of reducing the risk probability or the risk impact is obtained. For these mitigation efforts, many patterns exist, such as *rollback*, *rollforward*, *immediate fixing*, *deferred fixing*, *retry*, *compensate*, and *go-to-failsafe-state* [4,5,9,10,11]. Among them, the mitigation setup shown in Fig. 3 corresponds to the go-to-failsafe-state. This mitigation pattern implies that a system is transferred into a mitigation state to avoid dangerous effects and stops.

III. QUASI-EXPERIMENT USING BATTERY MODEL

To solve the research questions considered in our study, an experiment was conducted with a quadcopter drone. The Pixhawk device, in which the firmware corresponding to the brain of the drone is installed, reads the battery discharge level continuously during flight, and if a problem occurs because of which this value falls below a certain threshold, a pertinent pre-designed mitigation action is started. This process was designed on the basis of an experiment. Further, we experimentally observed the changes in the mitigation starting time according to the battery level sensing frequency through a quasi-experiment[12], which used a battery discharge model implemented mathematically in Matlab.

A. Settings

This experiment considers an independent variable and a dependent variable. The independent variable is the data sensing frequency. The filtering coefficient and frequency are factors that affect the mitigation decision for a failsafe operation. Since the battery level read in from the sensor fluctuates, the sensed filtering coefficient value is not accepted 100%; values only up to the specified filtering coefficient are accepted. Thus, by defining the filtering coefficient-applied value as a filtered level, we determine whether the filtered value is a level that will go over to a mitigation action. As explained earlier, since the filtering coefficient is a value set by battery experts and a value limited to the battery, our experiment does not use this as a variable. Instead, frequency is the independent variable considered in our experiment, and its value shows how often the battery level is read in from the sensor. Hereinafter, in this paper, frequency is denoted as Fr, as shown in Table 1.

The dependent variable related to Fr is the mitigation starting time. If the mitigation starting time is too late, the drone will fall down before finishing failsafe landing. Further, if the time is too early, the drone will still have some battery power left when it finishes landing. Similarly, the mitigation starting time has an important significance in terms of safety. Hereinafter, the mitigation starting time is denoted as T_{ms}, as described in Table 1. Further, our experiment reveals the relation between Fr and T_{ms}.

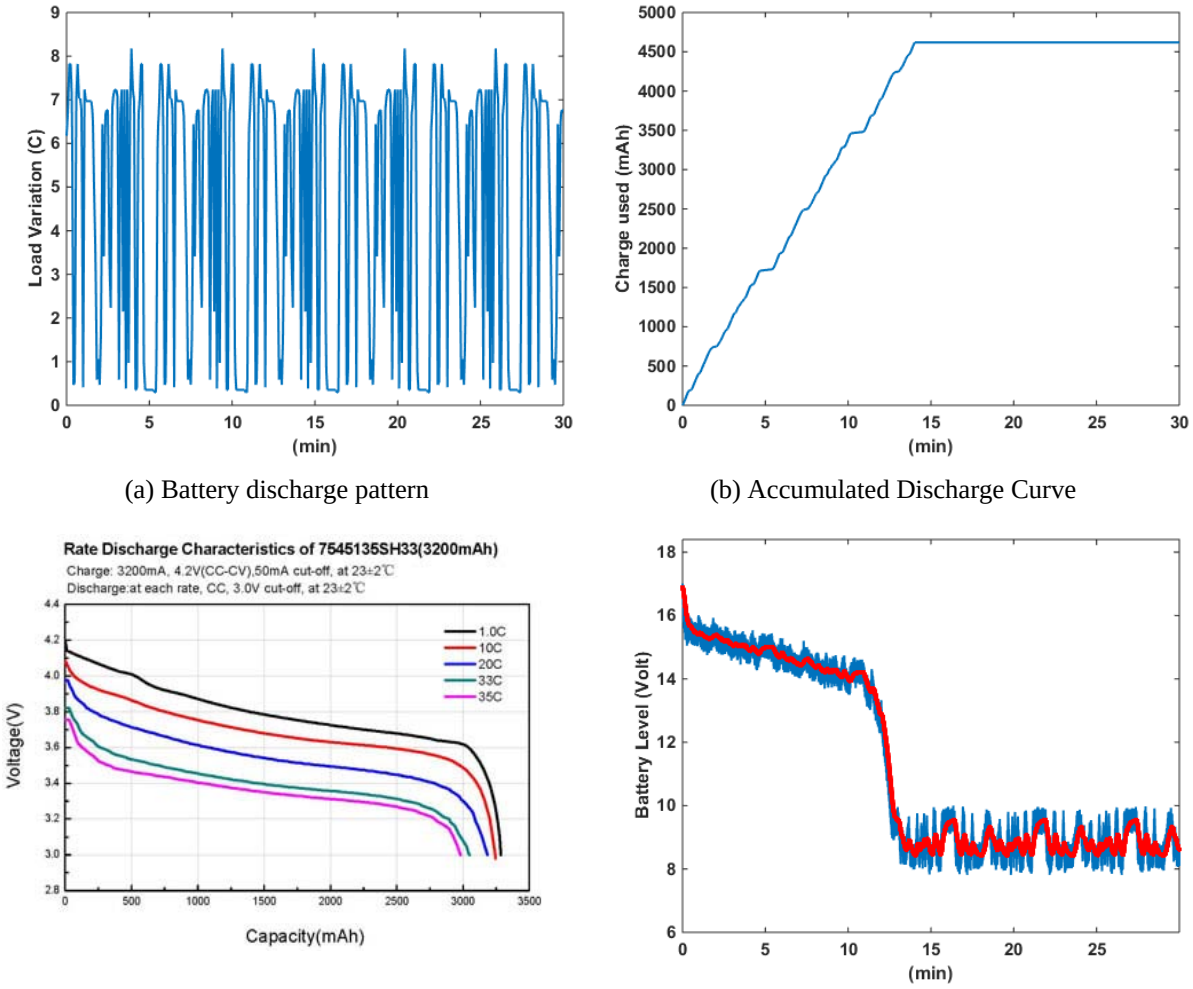
TABLE I. SETTINGS FOR THE EXPERIMENT

Settings	Description
Independent Variable	Sensing Frequency: *Fr*
Dependent Variable	Mitigation Starting Time: T_{ms}
Conditions	Filtering Coefficient: 0.001
	Critical Value for Mitigation: 10.2 V

This is a quasi-experiment controlled by some conditions; these conditions are specified in Table 1. Further, a filtered level is computed using the filtering coefficient of 0.001. This value of 0.001 is obtained from the source code of the drone firmware. In this experiment, we determine the starting time on the basis of the critical value of 10.2 V. This value of 10.2 V is a default value set in Mission Planner.

B. Battery Discharge Model

The experiment utilizes a battery discharge pattern of a drone [13] and a battery model of a typical Lithium Polymer battery of 4600mAh [14], which are implemented in mathematical functions. In Table 1, various *Fr* values are applied to the model and the related T_{ms} values are calculated. Running the experiment generates four charts showing how the battery works. Fig. 4 shows a set of these charts.



(a) Battery discharge pattern

(b) Accumulated Discharge Curve

(c) Lithium Polymer Battery Model [14]

(d) Output Voltage

Fig. 4. Mathematically modeled patterns.

In Fig. 4(a), the discharge pattern of the drone is expressed by the load variation over the flying time; it is expressed as "C rating," which is the ratio of the energy consumption to the rate that fully discharges the battery in an hour. Fig. 4(b) shows the accumulated energy consumption over the given load variation.

In Fig. 4(c), a typical 4200mAh lithium polymer battery is modeled [14], while Fig. 4(d) shows the resulting battery voltage as a composite result of the load variation and the battery model. As shown in this figure, a significant uncertainty is observed in the battery voltage, which is attributed to the fluctuation in the load variation as well as the modeling of the non-uniform chemical reaction of the battery cell.

Finally, since the red dots shown in Fig. 4(d) correspond to the filtered levels, our experiment uses these variables as the experiment conditions. These filtered levels, represented by the red dots, are computed using raw battery data from a sensor and the filtering coefficient assigned by a battery expert. We assigned a value of 0.001 to the coefficient variable, based on the definition in the firmware source code of the drone.

C. Results

Pixhawk's firmware defines the sensing frequency, i.e., *Fr*, as 100 Hz. *Fr* is an independent variable in our experiment and ranges *Fr* from 1 to 1300 Hz. As the value of *Fr* changes, the value of T_{ms}, which is its dependent variable, also changes. By operating the battery discharge model described earlier, we obtained the T_{ms} values for Fr of 1–1,300 Hz; these values are charted in Fig. 5. As shown in the chart, when a frequency above a certain threshold is set, the T_{ms} value of a similar frequency is obtained. In most cases, the mitigation starts at around 12 min 20 s. However, when Fr drops to below 100 Hz, the mitigation starting time is delayed rapidly (late). The decision to start the mitigation is made at 12.47 min for 100 Hz and at 24.42 min for 2 Hz. It should be noted that in the case of 1 Hz, the mitigation is not started throughout the flight time of 30 min.

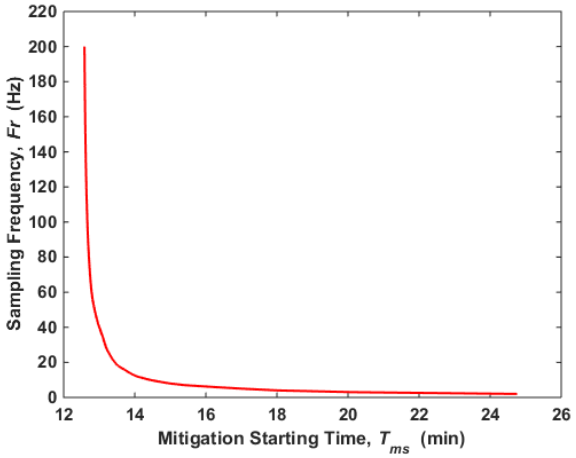


Fig. 5. Sensing frequency, Fr, and mitigation starting time, T_{ms}

D. Lessons Learned from the Experiment

The experimental results provide the values for the two considered variables, namely the frequency and the mitigation starting time from the data collected based on the frequency. For instance, (12.47 min, 100 Hz) denotes that the mitigation starts at 12.47 min after take-off, when the battery level values are collected 100 times per second. We show a chart with these pairs in Fig. 5. As mentioned in Section IIIC, T_{ms} is affected dramatically in the low-frequency range. On the other hand, in the high-frequency range, the T_{ms} values are almost identical. The mitigation starting time ranges from 12.3223 min to 12.4698 min, while the frequency ranges from 1300 to 100 Hz. This implies that a frequency increase does not affect the mitigation starting time considerably once the frequency is greater than the specified threshold; for example, 100 Hz in this case. This implies that when the data are more frequently sensed, the increase in process time and cost should be considered, and an appropriate frequency should be set so that the required efficiency can be achieved along with the desired level of safety.

E. Threats of Validity

1) Internal Validity: In the experiment, we examined the changes in the mitigation starting time (T_{ms}), i.e., which is dependent on the independent variable, frequency (Fr). However, in reality, in addition to Fr, the filtering coefficient is one of the factors that change T_{ms}. When the value of the filtering coefficient is changed, the filtered battery level changes. In other words, the distribution of red dots in Fig. 4(d) changes. However, the filtering coefficient is a value that reflects a battery characteristic, and because it is set as a constant by battery experts who are domain experts and is already hardcoded in the firmware, we did not have to consider a change in its value in this experiment. Furthermore, the value of the filtering coefficient used in our experiment was not an arbitrary value but "0.001," which is a coded value in the actual Pixhawk firmware source code. Consequently, once the filtering coefficient is assigned with a constant value, T_{ms} is affected by only by Fr according to the firmware's code logic.

2) External Validity: Our experiment is based on the filtered battery level data instead of raw level data collected from sensors. The battery level data used in our experiment are not randomly generated values. The values generated by implementing and executing a battery discharge model with Matlab were used. Although the battery discharge model shows the most common shape, the level of the flow battery consumed can change while the drone is flying because of external threats to the drone, such as weather changes and effects of a nearby structure. These external threats can eventually affect T_{ms}. However, since our experiment analyzes the extent of T_{ms} change with respect to the change in Fr rather than analyzing what the value of T_{ms} is, it is predicted that the extent of change in the T_{ms} values will be consistent in an identical external environment. Therefore, even if the battery discharge level values change because of a change in the external environment,

there should not be any change in the assertion with respect to the experimental results even if the shape of Battery discharge level under a certain frequency is changed.

IV. CONCLUSION

Mission critical systems including drones require the assurance of failsafe behavior. A drone failure leads to a crash, and as an analysis result of its causes reveals battery shortage as the cause of failure in most cases [3]. To ensure the failsafe behavior without drone failure, the battery discharge level is sensed and collected, and an adequate mitigation action is executed for a problem situation in which the battery level falls below a certain value. In the case of Pixhawk, a certain level that can cause a problematic situation is set and an adequate mitigation action for this is assigned through Mission Planner. The assigned setting value is ported to the firmware of the drone, and upon receiving the battery level from the sensor, the firmware continuously checks whether to execute the mitigation action or not. In the experiment, the time decided to go into mitigation was expressed as T_{ms}, and it was set as a dependent variable of the experiment. Furthermore, the data sensing frequency, which shows how often the data are received, was the independent variable of the quasi-experiment in this study. As a result of the experiment, we found that when Fr decreases below a certain level, T_{ms} gets delayed rapidly. On the other hand, for an Fr value above a certain level, the T_{ms} difference was very small compared with the corresponding Fr difference. The conclusions obtained on the basis of our research questions are as follows:

A. <RQ 1> Does the Mitigation Starting Time Get Delayed When the Frequency Decreases?

Depending on the frequency domain, two different aspects are shown. In the high-frequency domain, compared with the rate of Fr decrease, the change in T_{ms} was very small. When the frequency was reduced by 13 times, T_{ms} was delayed only by 1.4 min., i.e., the mitigation started about 4% late compared with the total time. In contrast, in the low-frequency domain, as Fr decreased, T_{ms} was rapidly delayed. When the frequency was reduced from 10 to 5, T_{ms} was delayed by about 2.3 min. Therefore, in the frequency domain below a certain level, since T_{ms} was delayed rapidly, there was a high possibility of an extreme failure: a crash might occur because the mitigation of the drone was not completed.

B. <RQ 2> To Obtain an Appropriate Mitigation Starting Time for Failsafe Behavior, Is It Better to Increase the Data Sensing Frequency?

As revealed by the experimental results, the T_{ms} value did not change at the same rate as the frequency. As mentioned above in <RQ1>, the change rate of T_{ms} was different between the high- and low-frequency domains. Of course, if the frequency is higher, T_{ms} can be calculated more accurately. Therefore, the result for <RQ2> is "YES." However, there is a part we cannot overlook. If the frequency increases, the amount of data to be processed increases and so do the corresponding processing time and cost. Would the accuracy of T_{ms} increase as much as the cost? Although accurate measurements were not taken, looking at the rate of change only, we found that the T_{ms} difference was just 4% when the frequency was increased by 13 times, as mentioned earlier. When this result is interpreted as a 4% benefit for a 13-fold cost increase, we are left with the question of whether this is good from the viewpoint of efficiency.

C. <RQ 3> Is There an Adequate Frequency for Setting an Appropriate Mitigation Starting Time?

As stated above in <RQ1>, when the frequency drops below a certain level, T_{ms} is rapidly delayed. Nevertheless, a high frequency is not preferred. As mentioned in the conclusion for <RQ2>, we cannot overlook the fact that when the frequency increases, the cost increases too. Therefore, taking efficiency into consideration, it is meaningful to find and set an adequate frequency for obtaining T_{ms} when there are no problems. From the chart of our experimental result, we can infer that a frequency right before the T_{ms} gets delayed rapidly can be set as the adequate frequency. A detailed measurement is planned as part of a follow-up research project.

ACKNOWLEDGMENT

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT & Future Planning(NRF-2014R1A1A3051827).

REFERENCES

[1] Linux Foundation Collaborative Project, http://www.dronecode.org, 2015
[2] Frank Bi, "Drone Maker 3D Robotics Raises $50 Million In Latest Round," Forbes, 2015
[3] Pål Moen Møst, Visual Navigation of an autonomous drone, Master Degree Thesis in Norwegian University of Science and Technology, Feb. 2014
[4] B. S. Lerner, S. Christov, L. J. Osterweil, R. Bendraou, U. Kannengiesser, and A. Wise, "Exception handling patterns for process modeling," IEEE Transactions on Software Engineering, vol. 36, no. 2, pp. 162–183, 2010.
[5] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," IEEE Transactions on Dependable and Secure Computing,, vol. 1, no. 1, pp. 11–33, 2004.
[6] Mission Planner Home, http://planner.ardupilot.com/
[7] Lorenz Meier, Petri Tanskanen, Lionel Heng, Gim Hee Lee, Friedrich Fraundorfer, Marc Pollefeys, "PIXHAWK: A micro aerial vehicle design for autonomous flight using onboard computer vision," Autonomous Robots, Vol. 33, Issue 1-2, pp 21-39, 2012.
[8] ArduPilot/APM development site, http://dev.ardupilot.com/

[9] F. Ye and T. Kelly, "Component failure mitigation according to failure type," in Proceedings of the 28th Annual International Computer Software and Applications Conference(COMPSAC 2004), pp. 258–264, 2004.
[10] Donmez, L. Boyle, and J. D. Lee, "Taxonomy of mitigation strategies for driver distraction," in Proceedings of the Human Factors and Ergonomics Society Annual Meeting, vol. 47, no. 16. pp. 1865–1869, 2003.
[11] S. Subramanian, L. Elliott, R. Vishnuvajjala, W. Tsai, and R. Mojdehbakhsh, "Fault mitigation in safety-critical software systems," in Proceedings Ninth IEEE Symposium on Computer-Based Medical Systems, pp. 12–17, 1996.
[12] Clae Wohlin et al, Experimentation in Software Engineering, Springer, 2012
[13] http://www.chargery.com/batteryPHS.asp
[14] http://www.rcgroups.com/forums/showthread.php?t=1653753&page=393

AUTHOR PROFILE

Hoijin Yoon received the Ph.D degree in Computer Science from Ewha Womans University in 2004. She has been working as a faculty in Department of Computer Engineering, Hyupsung University, since 2007. Her research interests are Software Testing and Safety Critical System.