International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:7, No:2, 2013

Ground System Software for Unmanned Aerial Vehicles on Android Device

Thach D. Do, Juhum Kwon, and Chang-Joo Moon

***Abstract*—**A Ground Control System (GCS), which controls Unmanned Aerial Vehicles (UAVs) and monitors their mission-related data, is one of the major components of UAVs. In fact, some traditional GCSs were built on an expensive, complicated hardware infrastructure with workstations and PCs. In contrast, a GCS on a portable device – such as an Android phone or tablet – takes advantage of its light-weight hardware and the rich User Interface supported by the Android Operating System. We implemented that kind of GCS and called it Ground System Software (GSS) in this paper. In operation, our GSS communicates with UAVs or other GSS via TCP/IP connection to get mission-related data, visualizes it on the device's screen, and saves the data in its own database. Our study showed that this kind of system will become a potential instrument in UAV-related systems and this kind of topic will appear in many research studies in the near future.

***Keywords*—**Android Operating System, Ground Control System, Mobile Device, Unmanned Aerial Vehicle.

I. Introduction

ACCORDING to Peter van Blyenburgh [1], UAVs are defined as uninhabited and reusable motorized aerial vehicles, which are remotely controlled, semi-autonomous, autonomous, or have a combination of these capabilities, and that can carry various types of payloads, making them capable of performing specific tasks within the earth's atmosphere, or beyond, for a duration. In reality, UAVs are used widely by the military for missions that are long, tiring, and dangerous for aircraft pilots. Consequently, the military has increased funding for the development of UAVs, which has caused the appearance of generations of UAVs with different capabilities related to their missions, such as tactical UAVs, MALE/HALE UAVs, or the modern Micro UAVs [2]. Moreover, the need for UAVs for non-military purposes increased considerably in recent years. Nowadays, we can see the role of UAVs in the entertainment and nuclear industries as well as in the fight against illegal activities, surveying wild animals, etc [3]. Interest in using UAVs is also growing within the scientific world. A number of studies have been done on the control method, system architecture, and software implementation for UAVs [4]–[14].

However, in order for UAVs to successfully complete more complicated missions, the GCS is indispensable. In particular, a GCS on an Android device is mobile and portable, and consequently can be deployed and operated anywhere with relative ease. On the other hand, implementing a miniature GCS [15] is not easy because of the hardware related limitations of small, portable devices. The first obstacle is the data link. Due to the restricted system hardware, a mini-type GCS can provide only a single, non-backup data link for data communication. The second problem is the display device. There is an obvious shortage of display area in a small device compared to powerful workstations or PCs. This makes the GUI design more difficult. While implementing our GSS, we tried to minimize those limitations by using reliable TCP/IP communication and making a careful plan of visualizing and displaying flight information.

In this paper, we present our approach to implementing a light-weight GSS that supports real-time data acquisition, real-time flight state display, flight data management, and off-line flight trajectory. In detail, through Ethernet communication, GSS acquires real-time GPS positioning information, the UAV's velocity and orientation. Such information is sent to the map and compass components so that operating personnel can visually see the UAV flight trajectory. The data management function is responsible for storing data for off-line flight trajectory or other analyses. In order to ensure the reliability of intercommunication, the GSS exchange a large volume of multi-type information via a TCP/IP connection that serves as a high-speed, reliable data link with the capability of detecting errors and faults in the data link. Moreover, due to the limitations of display devices, we carefully considered the process of visualizing and displaying information according to the different levels of importance and the real-time requirements for enhancing the efficiency of information interaction.

The rest of the paper is organized as follows: after giving a brief survey of some previous related research in Section II, the details of our implementation are described in Section III, including the system's workflow, classes, components, and database diagram. In Section IV, we show the real User Interface of our GSS and discuss its performance. Finally, in the last section we give some conclusions and suggestions for future work.

II. Literature Review

There have been many research studies involving GCSs for UAVs. Almost all of them are for large-scale GCSs with many functions: command transmission, flight control and navigation, 3D trajectory viewing, etc. Due to the large extent of their function, those systems require a powerful hardware

Thach D. Do is currently a student of the Department of Aerospace Information Engineering at the Konkuk University, Seoul, Republic of Korea (phone: +82-2-456-9169; e-mail: thachdo@konkuk.ac.kr).

JuHum Kwon received a Ph.D. degree from Korea University, Seoul, Republic of Korea. He was a Project Manager at central data processing center in Korea Air Force (email: jkweon@gmail.com).

Chang-Joo Moon is currently an Associate Professor of the Department of Aerospace Information Engineering at the Konkuk University, Seoul, Republic of Korea (e-mail: cjmoon@konkuk.ac.kr).

infrastructure with workstations or PCs. This raises the cost and makes the system architectures complicated. As an example, the purpose of the Ground Station Software System of the National University of Singapore [16] is to provide a friendly and realistic interface for users to monitor the process of flight tests. Thus, they focus on implementation of a 3D view of the helicopter, and it is actually convenient for users to navigate the UAV. Nevertheless, because of the lack of map component, users may have some difficulty keeping track of the UAV in a large area. Another example is the idea of AMFIS [17]. In this case, the developing team approached the GCS in a different way. They designed their GCS as a data integration hub that integrates and controls different kinds of sensors equipped in UAVs, UGVs, or in a sensor network. On the other hand, the heart of the system is the combination of workstations and servers, which are expensive and make the system more bulky.

In addition, some research has concentrated on the small-scale GCS. In those cases, the light-weight hardware is taken as an advantage and a smart phone is a good choice. Although the fact is that the limitations of small-scale hardware absolutely affect researchers' creativity in this case, a number of great works have been published in recent years. The Single-User Control of a MAV-Swarm [18] is an instance. In this project, they programmed an application on an HTC Android phone to control many micro aerial vehicles with different strategies. The AR.Drone [19] is also an excellent example of a small-scale commercial GCS. The AR.Drone appears as a game on Android or iOS that allows users to control the UAV using their own phone. In general, in the few attempts to install a GCS on small devices like smart phones, most of systems had friendly User Interfaces were easy to operate anywhere at any time. However, most of them have had the same approach, which was to use a portable and instable phone to control UAVs directly. This is an approach that may cause unexpected accidents for UAVs during operation. Consequently, this approach is quite risky and dangerous for both UAVs and the surrounding environment.

To summarize, to take advantages of both the functions and stability of a workstation-based GCS and the light-weight hardware of a small-scale GCS, we endeavored to implement as many of the functions of a workstation-based GCS as possible, such as flight tracking, data management, offline-trajectory replay, instead of allowing direct control of UAVs on an Android device.

III. Implementation

A. System Framework

The main framework, as shown in Fig. 1, invokes the Initial stage at first. In this stage, all of the graphic components, such as Compass, Map Views and some TextViews and other important variables including IP address, Port number, Connection Flag are created or declared. Then, the system starts to update some parameters that are set by the user in the Update Preferences stage. According to the parameters, the system knows exactly if it should make a connection and get data or just maintain the current Graphic User Interface (GUI). If the connection flag was set, the system, with updated network parameters, creates and calls a background thread for the next stage.

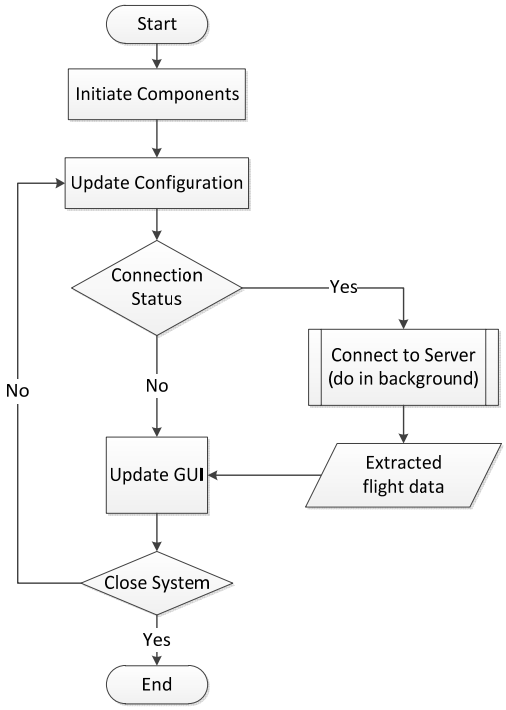


Fig. 1 The main workflow of Ground System Software

We call the next stage the Background stage because it runs totally in the background and is managed by the Android OS. Fig. 2 shows the detailed order of tasks in this stage. The main mission of this stage is to receive flight data, extract it to create meaningful information and pass it to appropriate components. In order, the socket and I/O Stream created first with the up-to-date network parameters. Later, the thread looks for the raw data in bytes that was sent from the remote data terminal. At the next step, the nine floating values of Position, IMU and Velocity are extracted from the raw data. Consequently, those meaningful values are stored in a database and then used to update the status of some GUI components. This routine of tasks is executed continuously until the connection mode is turned off.

B. System Components

With the properly designed framework and solution, we implemented our system's components using XML and Java programming language. Fig. 3 shows the main Java objects and their attributes, which are related to the designed components. In general, all of the components were implemented by inheriting and modifying some existing classes in the Android library, such as Activity for GUI, Content Provider for the database and AsyncTask for the background thread.

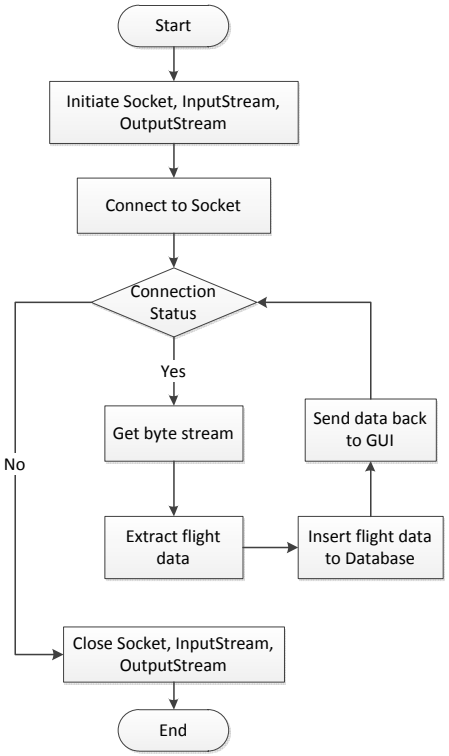


Fig. 2 The workflow of Background stage

In the set of GUI components, we implemented Pitching-Rolling and Yawing Compass corresponding to the information they have to represent and a Map component for a 2-D terrain display. Those Compasses are launched by the Java objects that are inherited from the same source – the native View class in Android library. In another case, the Map – an important component that appears on both the Main and Flight History Replay window – is implemented by the Open Street Map (OSM) library except for the native Google Map library. The most important reason the OSM was chosen is the offline map feature. To take advantage of this feature, we downloaded about 1 Gigabyte of map titles and stored them in the SD card on the Android device, and the Mobile Atlas Creator is the suitable tool in this case.

The second component – the Flight Data Provider – is involved in the database. As a child of the native Content Provider in the Android library, the Flight Data Provider provides an interface for publishing and consuming data based on a simple URI addressing model. As a result, it decouples the application layer from the data layer.

Last but not least, the AsyncTask takes responsibility for asynchronous tasks in the background. In detail, AsyncTask handles all of the thread creation, thread management and synchronization. It offers the event handlers to synchronize the GUI thread that updates View components and other graphic elements in order to report progress or publish results when the task is completed.

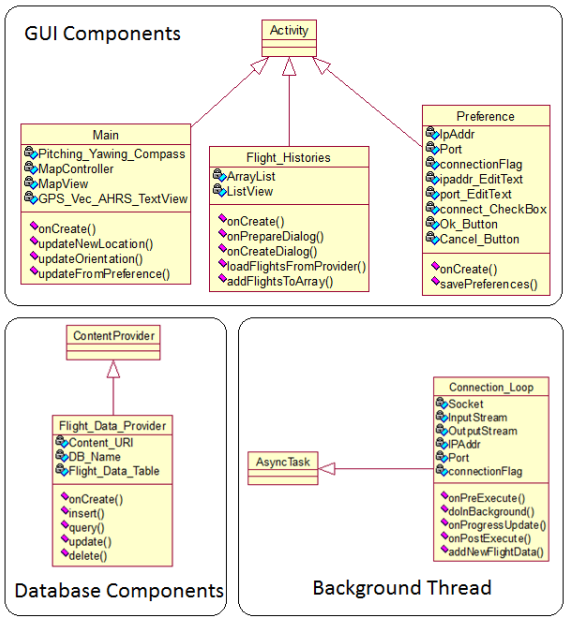


Fig. 3 The main Java objects of GCS in Android

C. Database Design

In addition to the real-time functions, we also added some functions that are useful to track and analyze the flight in past experimentations. With those functions, we can store the data and display it in the most visual way on a smart Android device. Thus, the selection of a relational database that can work with structured data such as SQLite in Android is necessary in our system.

Because the Ground System Software has to work with nine floating values of three kinds of data including GPS, IMU and Velocity in operation, it needs a database to store all of data in a way in which each value can be retrieved without any confusion. Our solution is modeled as an E-R diagram in Fig. 4 to satisfy the above requirements. We designed three tables to store three kinds of data respectively and another – the Event table – to join all the data into an event with the exact time when the event occurred.

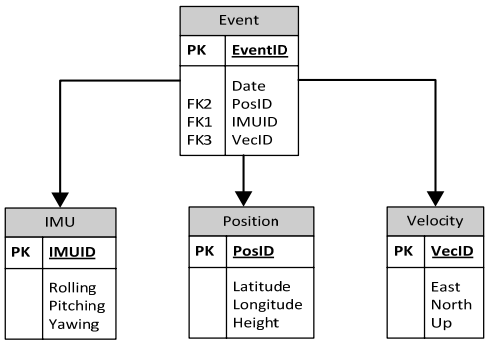


Fig. 4 The E-R diagram of the database in Ground System Software

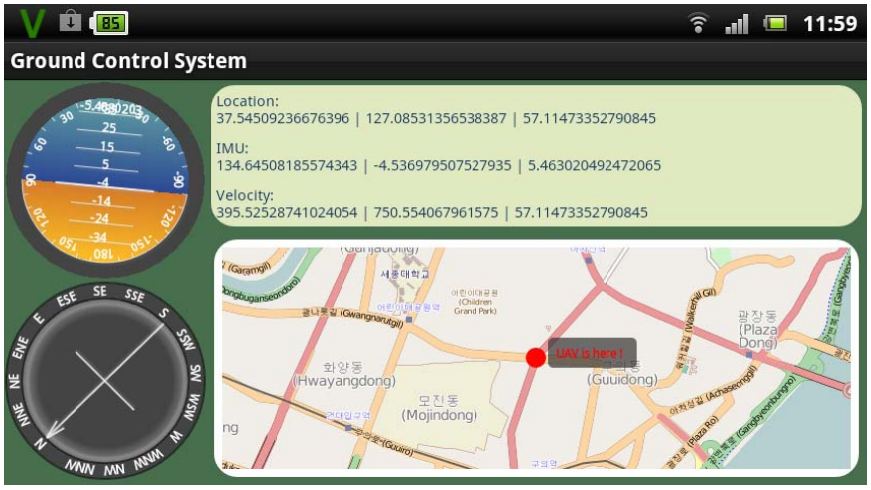


Fig. 5 Main window of Ground Station Software on Android device

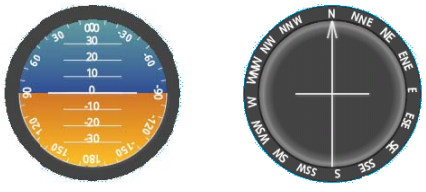
IV. Results and Discussion

A. User Interface

The ground station plays the role of terminal for end users to monitor the UAV through a wireless communication channel. Therefore, one of the important tasks of the ground station is to provide a friendly and realistic interface to display the data obtained and transferred by the UAV and for the user to interact with the system.

In our GSS, we divided and displayed the flight information as many windows that can be switched from the menu instead of putting everything in one because of the limited space on the screen of an Android device. The most important window, which is displayed first as the Main is shown in Fig. 5. It consists of three graphic components that represent three kinds of information. At the top of the Main window, we located a TextView component to display the raw data including nine 64-bit float numbers of Location, IMU and Velocity. In addition, as mentioned previously, we implemented two Compasses and a Map view that visualize the information of the IMU and GPS in the Main window.

The Compasses were made to present the information of IMU data that is the real state of the UAV in the air. As demonstrated in Fig. 6, the Rolling-pitching Compass obtains the pitching and rolling values received from the UAV and translates them into the visual image of ground and sky with some marker lines and numeric values, whereas the yawing state is represented by the Yawing Compass.



(a) Rolling-pitching Compass (b) Yawing compass

Fig. 6 (a) The Rolling-pitching and (b) Yawing Compass

As implemented from the Open Street Map library, the Map component, which is used to show the position of the UAV, supports a mechanism to store and load the offline map in any level of scale on an Android device. Using the offline map requires less time and Internet bandwidth than Google Map API, which requires the map to be downloaded continuously from the Google Server. Therefore, The Open Street Map is really a full replacement for the native Google Map in an Android system although the Open Street Map contains fewer details than the native Google Map, as shown in Fig. 7.

(a) Open Street Map

(b) Google Map

Fig. 7 The difference between Open Street Map (a) and Google Map (b) in the same area of Konkuk University, Seoul, South Korea

B. Flight Data Management

As mentioned in Section II, our system stores all flight data in its own database. Consequently, it needs a function that retrieves the data from the database and displays it on the screen. To satisfy that requirement, we offered users two options. The first option retrieves all flight data as the numbers and puts them on a ListView. Using this option, users can view all Location, IMU and Velocity information in the system's history, as shown on Fig. 8. In the second option, we implemented the off-line trajectory function. In detail, if this mode is turn on, the system will load 50 latest position of the UAV from GPS data and display them on the map. In Fig. 9, the UAV's positions are represented by the red dots.

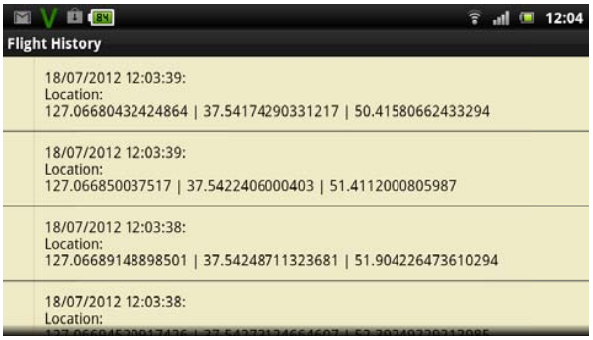


Fig. 8 The history flight data on the ListView

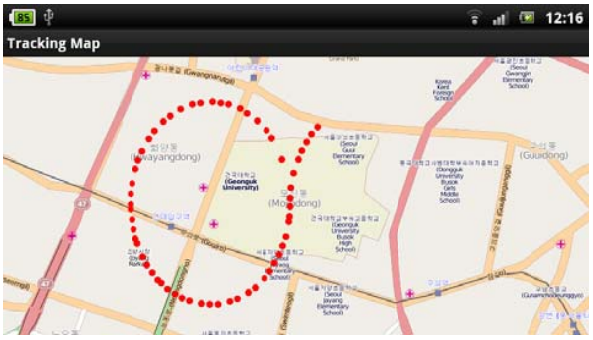


Fig. 9 The off-line trajectory replay mode

C. Communication

In our experiment, we chose TCP/IP as the main technology for the GSS and UAVs to communicate. Unlike UDP, the TCP guarantees that the receiver receives exactly what the sender intended to send, which means that data can be transferred in correct order without any error.

In addition to the connection protocol, the format of the data that is used to communicate between the Ground Control Systems and UAVs is also considerable. Our chosen data format is shown in Fig. 10. The data packet contains a total of 74 bytes including a 2-byte Header at first; the following 74 bytes cover the nine values of GPS, IMU and Velocity.

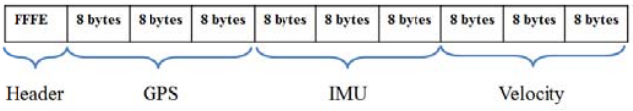


Fig. 10 The format of transferred data

V. Conclusion

The Ground Software System for Android devices presented in this paper includes most of the required basic functions of a Ground Control System: display function, flight data acquisition, management, trajectory replay. The display function was implemented with Android View components and the Open Street Map. The Open Street Map, which supports offline map, is a special point in the system because it reduces the data transferred via Ethernet and the amount of time to download the map tiles. The data acquisition function used TCP/IP to establish connections between different types of UAVs and GCSs. It ensures the reliability of data links with the capabilities of detecting errors and faults that occur in transferred data. In the case of data management, the SQLite database was used to reduce external dependencies, minimize latency, and simplify synchronization.

At this time, our GSS can only support real-time data acquisition and dynamic flight data display. Our plan in the future is to add more features such as Real-time Command Transmission, Video transmission and replay to make the proposed GSS become not only a data acquisition center but also a flight planning center for UAVs.

Acknowledgment

This work was supported by National Research Foundation of Korea Grant funded by the Korean Government (K2060100001).

References

[1] P. v. Blyenburgh, "UAVs: an Overview " Air & Space Europe, vol. 1, pp. 43-47, 1999.
[2] "Current and Future UAV Military Users and Applications " Air & Space Europe, vol. 1, pp. 51-58, 1999.
[3] "Civilian Applications: the Challenges Facing the UAV Industry " Air & Space Europe, vol. 1, pp. 63-66, 1999.
[4] G. Cai, B. M. Chen, K. Peng, M. Dong, and T. H. Lee, "Modeling and Control System Design for a UAV Helicopter " presented at the MED '06, Ancona, 2006.
[5] G. Cai, F. Lin, B. M. Chen, and T. H. Lee, "Development of Fully Functional Miniature Unmanned Rotorcraft Systems," in CCC 2010, Beijing, China, 2010, pp. 32-40.
[6] M. Dong and Z. Sun, "A Behavior-based Architecture for Unmanned Aerial Vehicles," in Intelligent Control, Taipei, Taiwan, 2004, pp. 149-155.
[7] A. H. Fagg, M. A. Lewis, J. F. Montgomery, and G. A. Bekey, "The USC autonomous flying vehicle: An experiment in real-time behavior-based control," in IROS '93, Yokohama, Japan, 1993, pp. 1173-1180.
[8] K. Harbick, J. F. Montgomery, and G. S. Sukhatme, "Planar spline trajectory following for an autonomous helicopter," presented at the Computational Intelligence in Robotics and Automation, 2001, 2001.
[9] M. J. Matari´c, G. S. Sukhatme, and E. H. ØStergaard, "Multi-Robot Task Allocation in Uncertain Environments," Autonomous Robots vol. 14, pp. 255-263, March 01 2003.
[10] A. Ollero, S. Lacroix, and L. Merino. (2005, June) Architecture and Perception Issues in the COMETS Unmanned Air Vehicles Project. IEEE Robotics & Automation Magazine.

[11] S. d. L. Parra and J. Angel, "Low cost navigation system for UAV's," Aerosp. Sci. Technol., vol. 9, pp. 504–516, 2005.
[12] E. Theunissen, A. A. H. E. Goossens, O. F. Bleeker, and G. J. M. Koeners, "UAV Mission Management Functions to Support Integration in a Strategic and Tactical ATC and C2 Environment," presented at the AIAA Modeling and Simulation Technologies Conference and Exhibit, San Francisco, California, 2005.
[13] G. Wang, J. Zhu, and H. Xia, "Model Identification and Control of a Small-Scale Unmanned Helicopter," presented at the ICCSE 2011, SuperStar Virgo, Singapore, 2011.
[14] F. Xu, Z. Zhaoying, X. Wei, and G. Qi, "MEMS-Based Low-Cost Flight Control System for Small UAVs," Tsinghua Sci. Technol., vol. 13, 2008.
[15] Y. Kang and M. Yuan, "Software design for mini-type ground control station of UAV," presented at the ICEMI '09, Beijing, China 2009.
[16] M. Dong, B. M. Chen, G. Cai, and K. Peng, "Development of a Real-time Onboard and Ground Station Software System for a UAV Helicopter," Journal of Aerospace Computing, Information, and Communication, vol. 4, pp. 933-955, 2007.
[17] F. Segor, A. Bürkle, T. Partmann, and R. Schönbein, "Mobile Ground Control Station for Local Surveillance," presented at the ICONS 2010, French Alps, France, 2010.
[18] M. Lichtenstern, M. Angermann, and M. Frassl, "IMU- and GNSS-Assisted Single-User Control of a MAV-Swarm for Multiple Perspective Observation of Outdoor Activities," in ITM 2011, San Diego, CA, 2011.
[19] AR.Drone 2.0. Parrot new wifi quadricopter. Available: ardrone2.parrot.com.

Thach D. Do (Vietnam, August 25th) received his B.S degrees in Computer Engineering from Ho Chi Minh city University of Technology, Vietnam in 2011. He is currently a student of the Department of Aerospace Information Engineering at the Konkuk University, Korea. His current research interests include embedded real-time systems, real-time operating systems, concurrent programming.

JuHum Kwon received a Ph.D. degree in the Department of Computer Science and Engineering from Korea University, Seoul, Republic of Korea in 2005. He received his master's degree in electrical and computer engineering from Wayne state university, U.S.A in 1999. His research interests include ontological foundations of knowlege representation with special emphasis on reasoning on taxonomic structure and ontology integration, Enterprise Architecture, and Semantic Web. He was a Project Manager at central data processing center in Korea Air Force.

Chang-Joo Moon received a Ph.D. degree in the Department of Computer Science from Korea University, Seoul, Republic of Korea, in 2004. He is an associate professor in the Department of Aerospace Information Engineering, Konkuk University, Seoul, Republic of Korea. His research areas include real-time embedded systems, computer security, and system integration.