Journal of Korea Multimedia Society Vol. 11, No. 6, JUNE 2008(pp. 835-844)

Compact Software Design and Implementation of IEEE802.15.4 and ZigBee

PHam Ngoc Thai[†], Victoria Que[††], Won-Joo Hwang[†††]

ABSTRACT

ZigBee devices are limited in resources especially on power and computational capacity but also require real-time operation at MAC layer. Therefore, it is important to take those requirement into consideration of system software design. In this paper, we proposed a compact system software design to support simultaneously ZigBee and IEEE802.15.4. The design strictly respects the resource and real-time constraints while being optimized for specific functions of both Zigbee and IEEE802.15.4. Various evaluations are done to show significant metrics of our design.

Key words: Sensor network, ZigBee, 802.15.4, design and implementation

1. INTRODUCTION

Known as the third wave in the computing paradigms, ubiquitous computing [1] uses small devices which can be embedded in objects or placed anywhere in the environment to collect, process, and deliver information wirelessly. The realization of ubiquitous computing environment pushes forward attention to wireless sensor network (WSN) from research and industry fields [2]. WSN is composed of small sensor nodes that have sensing, data processing and wireless communication components. Being low cost and low maintenance, WSN could be very appropriate in various application areas such as surveillances, home control, and battlefields. WSN testbeds and real-life applications have been started several years ago based on various platforms [3,4]. However, it was not a standard until June 2005 when ZigBee [5] is released as the first industrial standard for sensor networks based on IEEE 802.15.4 [6]. ZigBee enhances the functionality and interoperability of IEEE802.15.4 by providing flexible and extendable network topology. Furthermore, it is cost effective and easy to install. ZigBee network can easily adapt to topology changes caused by adding, removing or failure of any network node.

According to the computational capacity and power saving requirement of ZigBee network nodes, design and implementation must be done carefully to efficiently support low-power and low- capacity network nodes [7]. Software of such kind of system must be compact, simple, power efficient, small memory footprint, responsible in real-time and modular for ease of extension. However some current open source ZigBee stack released for application development, for example Chipcon Stack and openMac, do not completely meet such requirements.

※ Corresponding Author : Won-Joo Hwang, Address : (621-749) Obangdong, Gimhae, Gyeongnam, Korea, TEL : +82-55-320-3847, FAX : +82-55-322-6275, E-mail : ichwang@inje.ac.kr

Receipt date : Mar. 12, 2008, Approval date : June 30, 2008

[†] Dept. of Electronics and Telecommunication Engineering, Inje Engineering Institute, Inje University, South Korea.
(E-mail : thaipnt@yahoo.com)
[††] Dept. of Electronics and Telecommunication Engineering, Inje Engineering Institute, Inje University, South Korea.
(E-mail : tori_q81@yahoo.com)
[†††] Dept. of Information and Communications Engineering, Inje Engineering Institute, Inje University, South Korea.

1.1 Problem Description

For the reason stated above, we conclude the problems can should solved in this paper. Design and implementation of ZigBee must consider the constraints of real-time embedded network nodes.

Compact is an important required features of stack. Software system design, of course, must be optimized to fit low capacity device to save not only computational power but also power resources.

Timing is also a big issue for real-time operation. In the 802.15.4, timing precision is needed to accomplish back-off time of 320μs. Timing is usually done based on the integrated internal clock of micro controller. Moreover, we need to consider the memory constraints of the stack. To fit different type of devices, Reduced Function Device (RFD) and Full Function Device (FFD), memory management function must consider different memory allocation schemes.

Regarding the requirement in heterogeneous sensor network, a large number of different devices will be developed to conform to different environment. The devices, hence, will be adapted in terms of cost, power capacity and so on. Customization of stack design to fit those requirements is also a mandatory feature.

1.2 Paper Contributions

In this paper, we focus on the issues mentioned and present a system design of full ZigBee embedded software. The system is divided into several modules for ease of extension and maintenance. It consists of ZigBee protocol stack, operating system and application framework.

- Our design methodology aims to solve the constraints of ZigBee system and concentrate on a appropriate compact design for ZigBee. System design is based on event-driven architecture and non-preemptive multitasking single stack models to meet the real-time and low-capacity requirement. Hardware abstraction is done on Hardware Abstraction Layer (HAL) which allows adaptation of system over various type of hardware platform.

- Moreover, this paper provides discussion of the main problems in implementing IEEE 802.15.4/ZigBee and offers solutions to overcome those problems.

The remainder of the paper is organized as follows. On section 2, we describe related works on ZigBee and wireless personal area network and our motivation. On Section 3, we describe how we arrive to our implementation. We mention design factors, architecture, as well as the functional modules. Currently, we implemented the system over ATMega128L and Chipcon CC2420 platform [8]. On Section 4, we show test scenarios and results for the implemented protocol stack. Finally, on Section 5, we conclude the paper.

2. RELATED WORKS AND MOTIVATION

The diagram for ZigBee[5] protocol stack is shown in Fig. 1. It is composed of physical, MAC, network and application layers. The IEEE 802.15.4 Standard defines the lower layers: Physical (PHY)

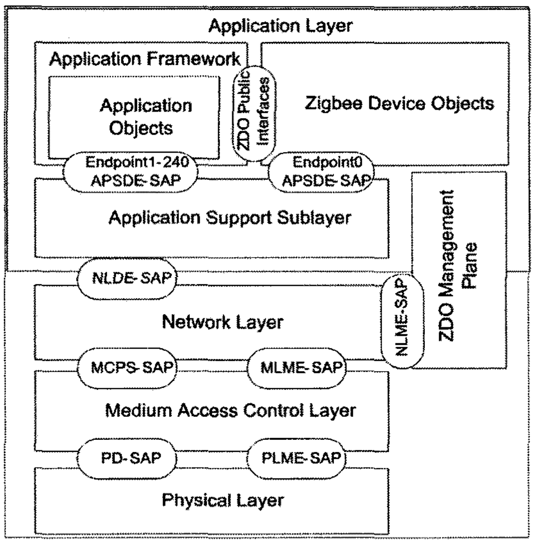


Fig. 1. MAC 802.15.4/ZigBee Protocol Stack

and Media Access Control (MAC) while the ZigBee alliance defines the upper layers. Communication between layer Application Objects and Zigbee Device Object (ZDO) are based on Application Sub-layer Data Entity SAP (APSDE-SAP) and Application Sun-layer Management Entity SAP (APSME-SAP). Functions of Network Layer are exposed to upper layer through Network Layer Data Entity SAP (NLDE-SAP) and Network Layer Management Entity SAP (NLME-SAP). Network layer accesses function of MAC layer over MAC Common Part Sub-layer SAP (MCPS-SAP) and MAC Sub-layer Management Entity SAP (MLME-SAP). In the last layer, Physical Layer exposes function over Physical Layer Data SAP (PD-SAP) and Physical Layer Management Entity SAP (PLME-SAP).

There are three different devices in ZigBee: the coordinator, router and endpoint. Each has tasks to perform in the network and particular capabilities. It is important to identify these differences so as to save available resources on them. For example, the memory on the coordinator may be more than the end point, such differences should be considered.

Presently, we can find several open releases of ZigBee architecture and implementation. Some of them appear to be not completely cooperating with the prerequisite of an embedded ZigBee system. For example, in the reference [9], a design and implementation of a MAC 802.15.4 is presented. In the reference [10], ZigBee software system Z-Stack which is an open source protocol stack still has some problems. Firstly, the separation between the libraries of the MAC layer and ZigBee stack creates redundant codes in the system. By merging common library between them we can reduce software size and complexity. Second, its memory allocation mechanism uses variable size which can cause memory partition and therefore degrade memory access speed. Some other ZigBee project suggests using TinyOS with NesC [11] on ZigBee system. However, TinyOS seems to be too complicated for a customizable sensor node. The macros in NesC [12] cannot be reduced completely by using optimization tools. From [13], authors conclude that motes and TinyOS are insufficient for IEEE 802.15.4 full functional device. They experienced a far too slow data transfer between layers in TinyOS.

3. COMPONENT DESIGN AND IMPLEMENTATION

On this section, we will discuss the design of the protocol stack. Problems in implementation and our solution to overcome them are also depicted in these sections.

The software system is divided into components based on functional abstraction methodology. Fig. 2 represents the layered design architecture of ZigBee stack. The system consists of following components: Embedded Network Operating system (ENOS), MAC IEEE802.15.4, ZigBee Stack. ENOS provides scheduling, timer service and memory management to ZigBee stack and all the application components as a basic operating system. It also provides a framework for all the ZigBee application in the ZigBee Application component. Application registers to the ZigBee stack over generic interface without interrupt ZigBee stacks. MAC IEEE802.15.4 is implementation component of IEEE802.15.4 standard. This

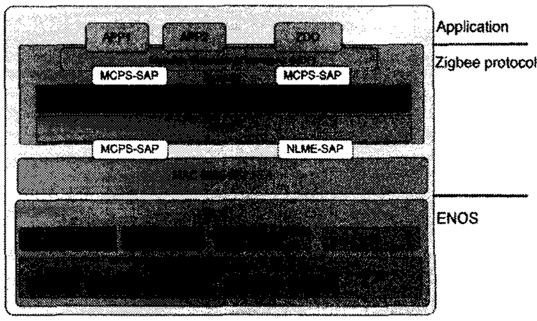


Fig. 2. ZigBee system architecture

component also uses functions of ENOS for all of its operations.

Here after, we will present component in the system and design and implementation issues related to specific those components.

3.1 Embedded Network Operating System-ENOS

In any embedded system, operating system is the most important component to take into account because it greatly affects the system performance. Our design philosophy of ENOS leans toward a compact, simple and appropriate for ZigBee functions. Components in the ENOS include Scheduler, Timer, Memory manager and Utils. A simple Message Dispatching Framework (MDF) is used support communication between tasks. We describe these components and their design considerations. The design of ENOS also considers constraints of ZigBee stack: Customizability, timing and memory requirement. General comparison between ENOS with TinyOS and uCOS is shown in the Table 1 to give an overview of ENOS.

3.1.1 ENOS Scheduler

Timing is critical design aspect of embedded sensor software. Delays can cause loss of data and degradation of system performance. Especially in ZigBee beacon-enabled network, all the nodes need to be synchronized with PAN (Personalized Access Network) coordinator to work properly. If a node loses the synchronization, there is a possibility of collision in the GTS (Guaranteed Time Slots), resulting from the overlap of CAP (Contention Access Period) with CFP (Contention Free Period). Timing requirement for tasks in the MAC 802.15.4 is for sending/receiving data and beacon synchronization. For the stated reason, we designed a two-level scheduling for ENOS scheduler. Periodic task (PTask) is scheduled every a fixed interval of time which is used for the critical tasks such as CSMA/CA procedure in MAC 802.15.4. Periodic tasks must terminate before the next activation. Aperiodic task (ATask) is a non-periodic task. There is no deadline for this kind of task. This task is activated by the scheduler and terminate whenever it finishes. For each kind of task, ENOS supports a number of priorities. That number of priorities should be considered adapted to specific requirement of system. Firing periodically by a hardware clock, periodic scheduling simply activates the task with higher priority each time. For aperiodic task, multi-level scheduling is also applied based on priority of the components. High priority aperiodic task should be assigned to protocol task, such as ZigBee. Applications should have low priorities. Interrupt can preempt both aperiodic and periodic task. Periodic task can also preempt aperiodic task but aperiodic task can not preempt any task. This design strategy can guarantee the real-time requirement of system and also reduce the overhead caused by context switching.

Communications between ATasks and from PTask to ATask are performed over MDF. ENOS scheduler maintains a queue of all the messages, and then message is dispatched to the appropriate task. This framework is simple and well fitted to ZigBee application framework

Table 1. Comparison of ENOS system with others

	ENOS	TinyOS	uCOS
Infrastructure	Event-Driven	Event-Driven	Threading
Memory	Fixed size, Dynamic	Static memory	Fixed size, Dynamic
Real-time	Yes	Yes	No

3.1.2. ENOS Timer

To efficiently utilize the priority of time critical task and computation task, ENOS support two types of timer, synchronous timer (Stimer) and asynchronous timer (Atimer). Synchronous timer will be activated immediately whenever it fire by using onboard clock interrupt. These interrupt callbacks a routine to activate those timers. Stimer can provide precise timer, however time consumed by this interrupt will affect the real-time feature of system that is reason why we need to keep as small as possible the number of Stimer. To do that, for non-critical task, for example on ZigBee stack or application layer, we should use Atimer. Asynchronous timer when started generates a message and sends to the destination task. This approach can reduce the time consumption of timer checking.

3.1.3 Memory management

Memory management component is necessary for ZigBee stack component and applications. Varied block size memory is usually used in some embedded systems, for example Z-Stack. However, with a small size of memory, varied size approach can cause the partition problem of the memory. We use fixed block size memory in ENOS. Memory management component supports several base sizes considering the memory utilization of system. For example, two base sizes are always needed. One is message size which is used for communication between tasks. The other one is maximum frame size (128 bytes) which is used for frames at MAC 802.15.4. To support customizability of stack, memory management needs to have a appropriate memory allocation scheme. There are two type of memory in the ENOS. The first type of memory is used at system level for function of ENOS and communication between components. This memory cannot be customized. The second type of memory is used for operation of application and ZigBee stack. This memory can be customized regarding to profile of applications.

3.1.4 Hardware Abstraction Layer - HAL

HAL provides abstraction interface to access hardware. Necessary abstraction can be added to extend interface of system to other peripherals. Currently, for ZigBee network nodes, we implemented the required components: RF, LED, Clock, UART and ADC. RF consists of functions and interrupts which allows MAC 802.15.4 to receive/send message and sense channel. HAL minimizes the change of system when porting to other hardware platform.

3.1.5 Message Dispatching Framework - MDF

Design of MDF is similar with design used for interprocess communication in the model operating system. Message in the system has a unified format including *{task_Id, message_Id}*. *task_Id* identifies the task will receive the message. *message_Id* identify the type of message. Message also contains a fixed size of data for customized uses. Messages are dispatched by sending directly to message queues of the task identified in the *task_Id*.

3.1.6 Interaction between components

In this section, we will detail the interactions between components described above following the interactions shown in the Fig. 3.

At the arrow (1), Application Support Layer component (APS) uses the Atimer in all its operation. At arrow (2), Network component (NWK) uses Atimer for all its operation. At the arrow (3), MAC 802.15.4 uses Stimer for its operation. Stimer can provide realtime guarantee for operation of MAC layer. At the arrow (4), applications access the ZigBee stack over a set of APSDE-SAP and APSME-SAP. Because APS

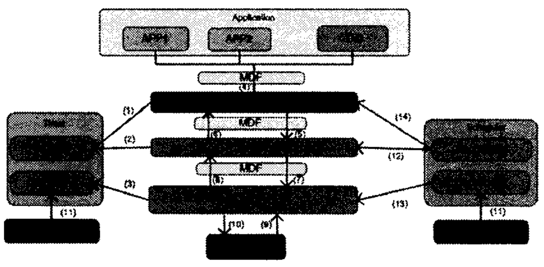


Fig. 3. Interaction between ZigBee stack and ENOS components.

and all other application work on independents Atask, then their communication are done over MDF. At the arrow (5), APS component access the service of NWK component. This interface is completely detailed in the ZigBee Standard [5]. At the arrow (6), Response/Data from NWK layer are passed to upper layer. This communication between NWK and APS is done over MDF. At the arrow (7), NWK component send request regarding the standard of 802.15.4 to the MAC. Those requests are queued at MAC and processed priodically. At the arrow (8), NWK component communicates with MAC layer over MDF. At the arrow (9), Data or Interrupt are generated by RF component. Generated interrupt will queue an Atask in the periodical task queue of MAC 802.15.4. That task will be queued until the previous generated tasks are processed. At the arrow (10), data or control parameters are sent to the tranceiver over the wrapped function of RF. Arrow (11) shows Internal hardware clock periodically generates interrupt to perform the Stimer and the Pscheduler. At arrows (12) and (14), Aperiodical Task Scheduler (AScheduler) will manage the task of NWK, APS and also all other application task. At arrow (13), tasks of MAC 802.15.4 are performed by Periodic Task Scheduler (PScheduler).

3.2 MAC IEEE802.15.4

Regarding the real-time requirement of MAC layer, MAC 802.15.4 is designed in periodical task paradigms. All the action in the MAC are Ptask. Hence there are two important components in the MAC layer. They are MAC_State which manages the internal information and current state of MAC. Task queue contains various queue of periodical task including different level of priority and Transactions.

Transactions contain the transactions generated by upper layer over two SAPs (MCPS-SAP and MLME-SAP). For the consistency of MAC, MAC_State must be check to start a new transaction.

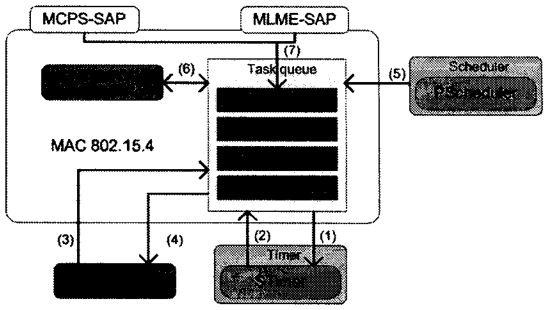


Fig. 4. Interaction between components of MAC 802.15.4

Details are well documented in the [5]. A new transaction will generate one or several Ptask. Those Ptask can also generate other tasks and queue it in the Task queue. RF and Stimer can also generate Ptask and queue it.

Internal interactions between components in MAC design are shown in Fig. 4. In the arrow (1), tasks in the task queue when activating can use Stimer for timing functionality. That interaction can be creation, or deletion a timer. In the arrow (2), when timer fired, it can call directly to task within the timer interrupt or create a new task and put it in to task queue. In the arrow (3), On Interrupt/Data, RF component generate task and put it into the Task Queue. In the arrow (4), PTask send data/control parameters to the RF. In the arrow (5), Scheduler schedules the Ptask in task queue of MAC component. PTask interact with MAC_State to manage/update the current MAC state in arrow (6). Request over access point of MAC will generate a Ptask on Transactions task queue. This queue has the lowest priority within all task priorities in arrow (7).

3.3 ZigBee Alliance

ZigBee Stack is implemented following the ZigBee standard defined by ZigBee Alliance. According to the defined functionalities, ZigBee stack is composed of four main components: NWK component which implements the network layer, APS component which implements the function of Application Support Layer and ZDO which implements function of ZigBee Device Object. In the case of ZDO, implementation of this component follows implementation scheme of a application and functionality defined in the standard. Hence we do not present further about it in this paper.

3.3.1 NWK Component

NWK component is implemented as a separated ATask. It communicates with upper layer, APS, and lower layer, MAC, over MDF. Details of this interaction have been presented in the Fig. 3. In this paper, we will present some keys point in implementation-dependent function of NWK layer which is not defined clearly in the standard. They are Beacon Scheduling algorithm (BSA). As we know, in the Beacon-Enable network, every router has to broadcast beacon to maintain synchronization of endpoint under its area. A new router joins network, it can only use unused segment within all segment of around routers. Beacon Scheduling also suffers from a hidden node problem. To avoid this problems, ZigBee network define a TxOffset value in the payload of the beacon, which indicates the timing difference between the beacon and the neighboring beacons. BSA is present in the Algorithm 1 to schedule Beacon and avoid the hidden node problem.

Algorithm 1: Beacon Scheduling Algorithm

```
Input: a: Current interval, b: beacon order, B:
An array of slot in the beacon interval.

- Scan all available beacon
- Mark used slot in B;a=b/2
While (a>1)
{
    Check segment number i*a in the array B
    ($0 \le i \le b/a-1$)
    If free slot found
        Return
    else
        a=a/2
}
```

3.3.2 APS Component

APS component implement the function of Application Support Layer. It's also implemented on a separate Atask as presented in the Fig. 3. In this component, we also integrate an important function ZigBee stack which is Application Framework. This framework provides entry point for ZigBee application enpoints. Each enpoints is registered as one aperiodical task at the compilation of software. Communication of application and ZigBee stack is over MDF as explained before. ZigBee application accesses to ZigBee stack over predefined SAP and access to the others system resource over NEOS.

3.3. Implementation challenges and solutions

The first challenge is to keep synchronization in the beacon-enabled network. When a node receives a beacon, it takes some delay until it can process that beacon. Two techniques are required to minimize that delay. Firstly, periodic task which process the beacon must have highest priority. Second, compensation to that delay must be added to the back-off time.

The second challenge is that we must control the time consumed of every periodic task. The periodic tasks are performed directly from a timer interrupt. Periodic tasks must be finished within one period (which is a back-off time: 320 micro-second in our system). This can be done by monitoring the time consumption of every created periodic task. The implemented protocol should avoid unnecessary operations. Several techniques can apply such as using appropriate variable, avoiding multiple copies of the same operation and data redundancy and optimizing the operation of frequently called functions.

4. EVALUATION METHODOLOGY

Table 2 shows the memory footprint of our stack

Table 2. Memory footprint of Protocol Stacks

	Coordinator	Router	End point
Our ZigBee	127kb	126kb	107kb
Z-Stack	236kb	255kb	177kb

and Z-Stack (with debugging functionality off) in Home Lighting Profile application. Our designed stack has a footprint significantly smaller than the Z-Stack footprint. In the next parts of this section, we will present the concern related to evaluation the correctness the protocol.

Evaluation of the ZigBee is performed over two kinds of test scenarios. Firstly, message format, message sequence, and addressing test evaluate the correctness of the ZigBee Stack in term of message behaviors which includes the correctness of message format, correctness of message sequences and addressing of the ZigBee. Second, ZigBee profiling testing is perform to evaluate the correctness of behaviors of ZigBee network in an appropriate application, which includes network formation, network maintenance and profiling of an appropriate application on ZigBee network.

4.1 Message Format, message sequence and Addressing test

The hardware for this test includes two nodes: one endpoint and one coordinator as in the Fig. 5a. We turn off/on repeatedly one of the hardware so that it would connect and disconnect. This simple process is done 100 times. While doing this, Air Sniffer [14] software is running on a computer and records the information on the network. The number of requests and responses are counted. Each requests should be able to get a corresponding join/disjoin as expected. We count a pair of request /response as a success otherwise an error. Table 3 shows the result of the test for MAC, NWK and all protocols. For both MAC and all protocols, there was a recorded 4% error, while on NWK there were none. The error is counted when an appropriate response is not received after a request.

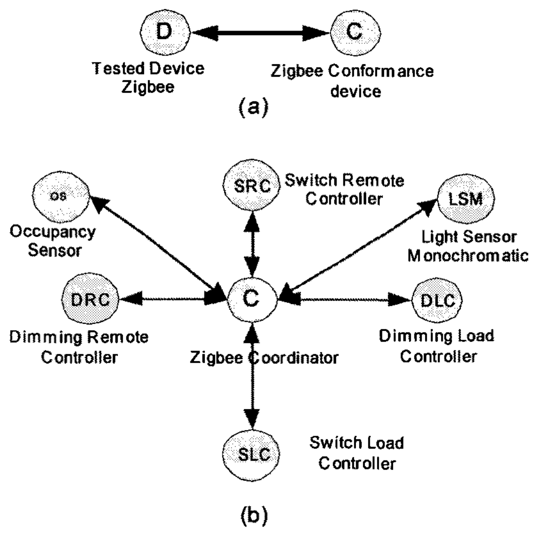


Fig. 5. Testing scenarios. (a) Test scenario for message format and addressing conformance. (b) Test scenario for Stack Profiling

Table 3. Summary of Error Percentage of Test Scenarios

Test Type	MAC	NWK	DATA	ALL
Message Format	4%	0%	N/A	4%
Message Sequence	0.1%	0.1%	0.04%	0.24%
Addressing	4%	0%	N/A	4%

4.2 ZigBee Profiling: Home Control Lighting (HCL) Profile Test

ZigBee Profiling test is done by implement Home Control Profile of ZigBee network. We implement six types of device in the HCL Profile including Switch Remote Controller, Switch Load Controller, Dimming Remote Controller, Dimming Load Controller, Occupancy Sensor and Light. Besides a full function ZigBee network coordinator also is integrated to be as Coordinator for this test scenario (Fig. 5b). After ZigBee network of 7 those nodes are established, we perform the function of Home Control Lighting Profile and monitor its using Air Sniffer. Message format, sequence and lighting event are monitor to assert the correctness of the profiling. The test results on different HCL test scenarios are shown in the Table 4. Failed tests are caused by the failures on the network layer.

Table 4. Success rate of the HCL Test Scenarios

Network formation success rate(%)	Binding success rate(%)	On/off success rate(%)	Dimming success rate(%)
98	98	99.44	99.74

5. CONCLUSION

Regarding the strict requirement in performance of ZigBee device, we notice the vital position of system software design. In this paper, we proposed a compact software design for ZigBee device supporting not only ZigBee standard but also realtime requirement for MAC IEEE802.15.4. In the further work, we will perform the exhaustive evaluation of system performance in comparison with other design.

REFERENCES

[1] T. Kindberg and A. Fox, "System Software for Ubiquitous Computing," *IEEE Pervasive Computing*, Vol.1, No.1, pp. 70-81. Mar. 2002.

[2] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, Vol.40, No.8, pp. 102-114, Aug. 2002.

[3] J.Y. Jung and J.W. Lee, "ZigBee Device Design and Implementation for Context-Aware U-Healthcare System" *In Proc of Proceedings of the Second International Conference on Systems and Networks Communications*, pp. 22, 2007.

[4] Z. Y. Wei, Z. X. Xing, and S. J. Feng, "The Design of Wireless Sensor Network System Based on ZigBee Technology for Greenhouse," *Journal of Physics*, Vol.48, No.1, pp. 1195-1199, 2006.

[5] ZigBee Specification Version 1.0, ZigBee Alliance.

[6] IEEE Standards 802.15.4: Wirereless Medium Access Control and Physical Layer Specifications for Low-Rate Wireless Personal Area Networks.

[7] W. Andy, "Embedded ZigBee design considerations," *Wireless Design & Development Magazine*, Dec. 2005.

[8] Chipcon CC2420 IEEE 802.15.4 compliant chip Datasheet.

[9] L. Ko, Y. Liu, and H. Fang, "Design and Implementation of IEEE 802.15.4 Beacon-enabled Network Devices," *In the Proc of Pervasive Computing and Communications Workshops 2006*, pp. 1-5, Mar. 2006.

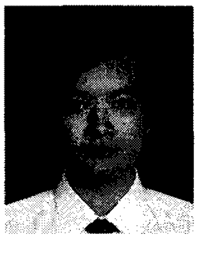
[10] Z-Stack ZigBee Software Stack, Texas Instruments Incorporated.

[11] TinyOS. http://www.tinyos.net.

[12] D. Gay, P. Levis, and R. Behren, "The nesC Language: A Holistic Approach to Networked Embedded Systems," *In the Proc of Conference on Programming Language Design and Implementation*, pp. 1-11, 2003.

[13] J. Thomsen and D. Husemann "Evaluating the use of motes and TinyOS for a mobile sensor platform" *Proc. of the 24th IASTED international conference on Parallel and distributed computing and network*, pp. 95-100, 2006.

[14] FTS4ZB IEEE 802.15.4 and ZigBee Wireless Protocol Analyzer, Packet Sniffer and Bus Analyzer.



Pham Ngoc Thai

Received his bachelor's degree in Information and Communication System from Hanoi University of Technology, Vietnam in 2004. He is now a Masters student of the Department of Electronics and Telecommunication Engineering, Inje University, Gimhae, Republic of Korea. His research interests are routing in wireless network and embedded sensor network.

Ma. Victoria Que

Received her M.S. degree from Inje University (Gimhae, South Korea) in 2007. Her research interest includes computer and sensor networks.

Won-Joo Hwang

Received his Ph.D Degree from Osaka University Japan in 2002. He received his bachelor's degree and M.S. degree in Computer Engineering from Pusan National University, Pusan, Republic of Korea, in 1998 and 2000. Since September 2002, he has been an assistant professor at Inje University, Gyeongnam, Republic of Korea. Currently, he is the director of the Computer Networks Laboratory of the same school. His research interest is in network optimization and ubiquitous sensor networks.