

## Development of a Portable Embedded Patient Monitoring System

Jungmuk Kang<sup>1</sup>, Sungil Yoo<sup>2</sup> and Dongik Oh<sup>1</sup>

<sup>1</sup>Department of Medical IT Engineering,

<sup>2</sup>Department of Medical Science, SoonChunHyang University, Asan, Korea

wjdanr89@nate.com, sgyoo@sch.ac.kr, dohdoh@sch.ac.kr

### Abstract

*One of the first medical devices we encounter when we visit a hospital care unit is the patient monitoring (PM) system. This system informs doctors and nurses about the patient's vital signs. However, typical PMs do not have a remote monitoring capability, which necessitates constant onsite attendance by support personnel. Further, PMs are generally too large to carry. Thus, we have developed a Wireless Patient Monitoring System (WIPAMS), which is a portable PM that monitors the biomedical signs of patients in real time and sends them to remote stations for display, with alerts when necessary. WIPAMS differs from conventional PMs in two aspects: first, its wireless communication capability allows vital signs to be monitored remotely so it obviates the requirement for onsite support personnel; and second, it is portable so patients can move around while their biomedical signs are being monitored. WIPAMS is also notable because of its implementation: it uses Android as the underlying OS, while a pulse oximeter, thermometer, and electrocardiogram reader are integrated into the system. Monitoring and communication applications are implemented on the OS using threads, which facilitates the stable and timely manipulation of signals and the appropriate sharing of resources. Based on the graphic display of the biomedical data collected, we verified that the multitasking implementation used in the system was suitable for PMs and for other u-Healthcare applications.*

**Keywords:** patient monitoring, bio-medical signs, wireless portable system, embedded system, Android OS, u-Healthcare

### 1. Introduction

One of the first medical devices we encounter when we visit a hospital care unit is the patient monitoring (PM) system. Doctors and nurses are informed about the vital signs of patients by the PM so they can provide timely and appropriate treatment. The four most important vital signs are the pulse rate, oxygen level, body temperature, and electrocardiogram (ECG) activity [1, 2], which are provided by most PMs in the market. The measurement of vital signs should involve minimal intervention by the patient while providing accurate and stable data at the same time [3]. The PM also needs to detect emergencies and inform medical personnel when they occur.

Conventional PMs monitor the vital signs constantly but they are not provided to the medical personnel in real-time. This can be a serious problem during emergencies because the patient's life may be in danger if immediate attention is not provided. Another problem with

conventional PMs is that most are bulky standalone machines. Some models are connected to networks but they are usually hard-wired. If small PMs could be connected to a network wirelessly, patients would be able to move around freely while their vital signs are monitored. Thus, medical personnel could be informed about a patient's critical condition regardless of their whereabouts and they could be treated promptly if an emergency occurs. Furthermore, portable devices can be integrated into the u-Healthcare environment and used to develop novel applications [4, 5].

Thus, we developed a portable embedded device that can monitor the condition of patients in real time and provide various biomedical signals via wireless communication so that the vital signs may be monitored remotely. We call our device a Wireless Patient Monitoring System (WIPAMS). This type of PM allows the patient to move freely while their biomedical signals are measured continuously. The data are delivered to medical personnel in real time so appropriate actions can be taken if necessary.

The physical dimensions of WIPAMS are small and we ported an embedded operating system (Android OS) [6] into the module. A pulse oximeter, thermometer, and ECG reader are integrated into the system via serial communication ports. The monitoring and communication modules are implemented on the OS using threads, which facilitates stable and timely responses to external events and the appropriate sharing of resources. Using the custom-programmed graphical display of the monitoring data, we verified that the multitasking implementation of our system was suitable for the PM and various other u-Healthcare applications.

This paper is organized as follows. In Section 1, we provide the rationale behind the development of WIPAMS. In Section 2, we briefly explain the theories involved with measuring biomedical signals. We also describe the hardware and software platforms used to build the system. Section 3 describes the detailed implementation of WIPAMS by focusing on its software implementation. In Section 4, we describe the development of a WIPAMS prototype and discuss its application performance. Section 5 discusses future improvements to WIPAMS and concludes this paper.

## **2. Background Information Related to the Development of WIPAMS**

WIPAMS is a PM system that collects vital biomedical signs from patients using sensors and presents the information in a form that can be interpreted easily by medical personnel. In this section, we describe the biomedical sensors used for WIPAMS development and discuss the software platform used to process the data acquired by the sensors.

### **2.1. Biomedical sensors**

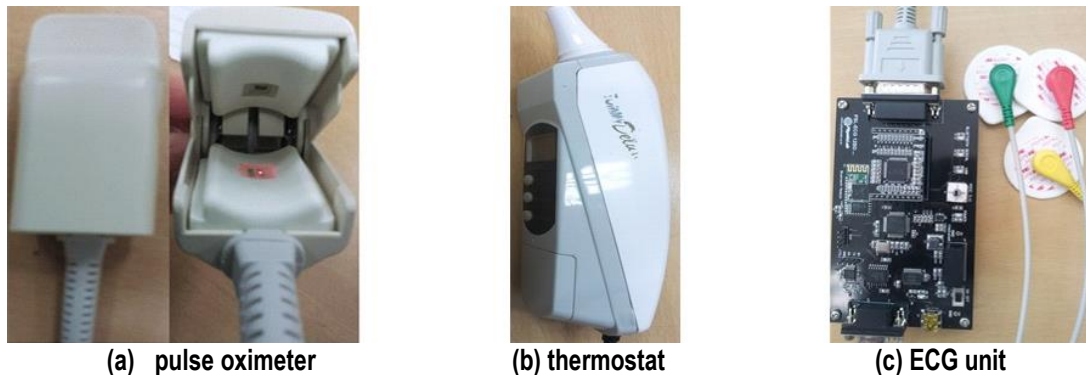
At present, WIPAMS measures four different biomedical signs: blood oxygenation, pulse rate, body temperature, and ECG activity.

Blood oxygenation is a very important factor when evaluating the circulatory and respiratory condition of a patient. Oxygen transported from the lungs to the tissues is carried mostly by the blood plasma. Measurements of arterial blood oxygen in patients are used by the physician to adjust mechanical ventilation. Oxygen can be measured noninvasively using a pulse oximeter, such as the one used by WIPAMS, as shown in Figure 1(a), which is produced by HyBus [7]. Two LEDs are turned on and off sequentially, and a photo detector on the other side reads the output from the lights after they pass through the finger. Based on the magnitude of the signal output detected by the unit (the magnitudes of the red and infrared light are measured by the photo detector), we can determine the oxygen level carried by the

blood plasma. The pulse rates can also be measured using the same unit by checking for periodic changes in the output measured by the detector.

Body temperature is one of the most tightly controlled physiological variables in patients. A high body temperature often indicates infection, whereas a low body temperature may indicate that the patient is in shock. Two types of devices are used to measure the body temperature: thermistors that measure the temperature in the armpit, mouth, or rectum; and non-contact thermometers that measure the temperature in the auditory canal. In our device, we measure the infrared radiation emitted by the device and the radiation reflected from the tympanic membrane. WIPAMS uses a non-contact thermometer produced by HuBDIC [8], as shown in Figure 1(b). To obtain the body temperature, WIPAMS users are required to pull the trigger on the device and the updated data are displayed on the remote screen instantly.

An ECG measures and amplifies the small electrical potential produced by the heart. ECG measurements can be used to detect abnormal heartbeats so possible heart malfunctions can be identified. The measurements are acquired using similar electrodes made of the same metal. Two or more similar pairs of biopotential electrodes can be placed near the heart to measure the electrical potential it generates. Figure 1(c) shows the electrode and processing board used by WIPAMS, which was a module developed by PhysioLab [9].



**Figure 1. Biosensors used by WIPAMS**

Each biomedical sensor is connected to a USB port on WIPAMS and RS232 serial communication is used for data exchange. Each device defines its own data format for the signals they measure. The data formats used by each unit are shown in Section 3.1.

## 2.2. Embedded systems based on operating systems

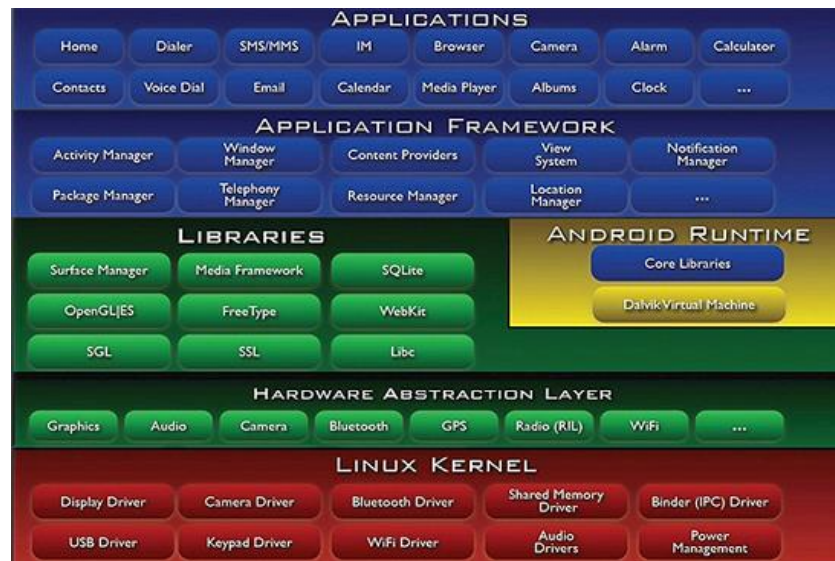
An embedded system is an electrical unit that provides a specific functionality, which gives control to the machine containing the system. This is the opposite concept to a general-purpose computer such as PC, which has many roles depending on the program logic it receives. Early stage embedded units performed their logic function using only hardware circuits, but the advent of cheap microcontrollers means that many embedded logics can be programmed and controlled by software [10].

However, the requirements of embedded systems may be highly complex so a single-process execution model on the microcontroller is not sufficient to meet the specifications. In particular, the real-time response to external events might not be integrated well using a single-process model.

Recently, microcontrollers have become more powerful and smaller in size, so OS porting to the embedded system is becoming a popular option. The main advantages of using an OS

on the embedded system are that the programmer can use various libraries and utilities, which are provided by the OS. OS support makes programming easier while the execution and processing of applications are more stable. The concurrency features of an OS make the real-time response to external events much easier to program.

The programs used by an OS are generally written in assembly language or C programming languages. However, recent advances in the mobile OS platform allow the writing of embedded programs in Java, which provides greater modularity and richer interfaces for the programmer. The OS used by WIPAMS is the Android OS, which was originally developed by Android Inc. and who later merged with Google. Android is an OS designed for smartphones and tablets, which are other examples of embedded systems. Android is a Linux-based OS that adds an extended application interface layer and a runtime unit to Linux, so Java applications can make use of Linux to run their executables. Figure 2 shows the Android OS architecture. Linux is the bottom layer and the Dalvic Virtual Machine is responsible for the multitasking features of Java.



**Figure 2. Android OS Architecture (copied from developer.android.com)**

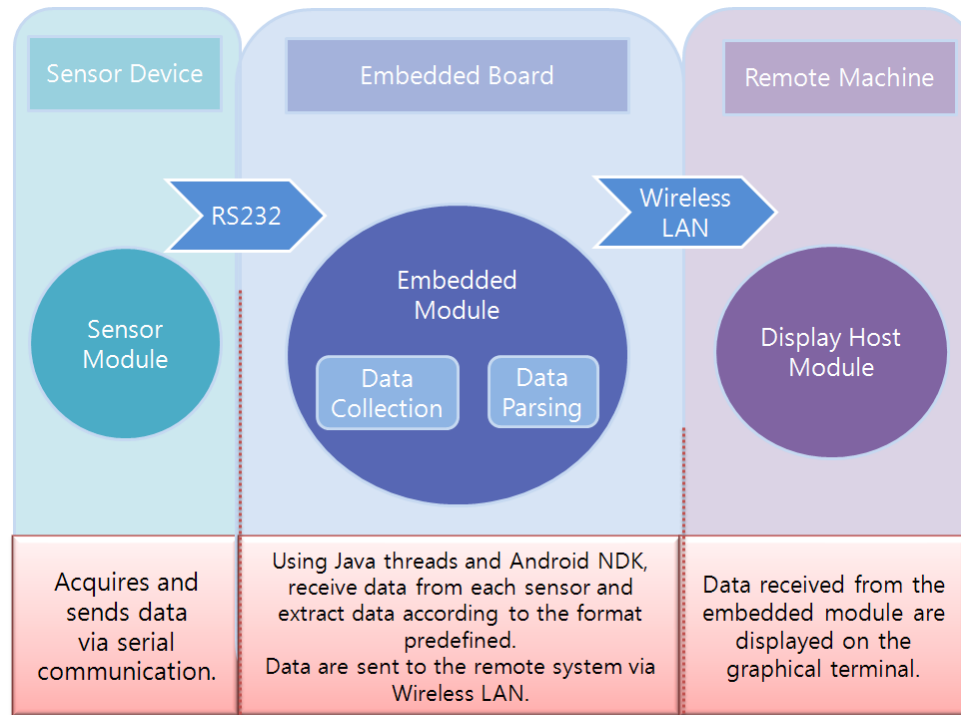
The hardware peripherals of embedded systems are accessed using libraries written in C/C++. The interface between this module and the Java code is achieved by Native Development Kit (NDK) integration. In Section 3, we describe the embedded integration of the WIPAMS software in detail.

### 3. Implementation of WIPAMS in the Android OS

WIPAMS comprises three major HW/SW modules, as shown in Figure 3. The first module comprises the biomedical sensors, which collect data from a patient and relay them to the main logic implemented on the embedded board, using standard serial communication (RS232). The second module is the control logic implemented in the Android OS. The logic monitors the sensor inputs attached to the board and responds to events immediately. Appropriate concurrency control is achieved by the underlying OS. We implemented the monitoring logic of each sensor using Android NDK and Java threads. The data are parsed and biomedical figures are extracted from the messages received. The data formats of each sensor are summarized in Section 3.1. A wireless LAN module is integrated into WIPAMS

and the parsed data are sent to the remote machine using this module. The received data are displayed on the graphical terminal of the remote station, which is the third major module in WIPAMS.

In this section, we describe the detailed implementation of the three major modules in WIPAMS.




**Figure 3. The three major components of WIPAMS and their roles**

### 3.1. Sensing module

As described in Section 2, many sensors can be used to acquire biomedical signals and various theories are involved with their measurements. However, the measurement theories were not the focus of this study. We were more concerned with developing an integrated system that can be used in various IT environments. We selected sensors that provide serial connectivity because this was the main communication channel on the hardware board we used (Odroid-X [11]). They are connected to the USB port for communication. Their data format was described in the data sheet provided by the manufacturers but a summary is shown in Table 1. In the WIPAMS implementation, we needed to filter out unstable data acquired by the sensors. This was possible because the data format clearly indicates the starting and ending pattern in the message. The elimination of unreliable data was not a problem because the data acquisition rate was much higher than that of the display unit.

**Table 1. Data formats of WIPAMS sensors**

Device	Data format										
Pulse-Oximeter	Start Bit	X	X	HR	HR	SPO2	SPO2	X	X	State	End Bit
	0xFA	-	-	m	n	X	Y	-	-	0x03	0xFB
	HR (m): Heartbeat (quantity in the third, second digit in Hex.) HR (n): Heartbeat (quantity in the first digit in Hex.) <Formula for getting heartbeat rate (HR)> HR = (m & 0x0F) * 100 + ((n & 0xF0) / 16) * 10 + (n & 0x0F)										
	SPO2 (X): Oxygenation (quantity in third, second digit in Hex.) SPO2 (Y): Oxygenation (quantity in the first digit in Hex.) <Formula for getting blood oxygenation (SPO2)> SPO2 = (X & 0x0F) * 100 + ((Y & 0xF0) / 16) * 10 + (Y & 0x0F)										
Thermometer	Data Type	Start Bit	X	X	Temperature	Temperature					
	Data	-	-	-	m	n					
Temperature (m): temperature in Hex. Temperature (n): an integer between 1 and 6 <Formula for getting the temperature> Temperature = n * 12.8 + m											
ECG	Data Type	Start Bit	Data	FrameData	Checksum	End Bit					
	Data	0x7E	0x01	27byte	1byte	0x03					
											

### 3.2. Control logic

The control logic of WIPAMS is responsible for acquiring sensor data from multiple input ports without causing conflicts or delays. The logic utilizes the tasking features of the underlying OS. Each input device is handled by a designated thread, so that an immediate response to the incoming data is possible.

The serial ports are accessed via a C language routine. The Android NDK allows Java programs to utilize embedded OS interfaces. Thus, we wrote a software module in the C programming language to access the port. This module initiates port access and reads the data via standard Linux I/O system calls. NDK allows this module to be built as a runtime library module, which is available during the application execution. By calling this runtime stub, we can access the underlying hardware devices from Android application programs. The simplified code used to acquire data from the SPO<sub>2</sub> device is shown in Figure 6.

```
public class Monitor extends Activity{
    static {
        System.loadLibrary("spo2"); // A library created by NDK. The library code is written in C
                                   // and it uses SO system calls to acquire data from the device
    }

    public native String getSpO2(); // A Java method to call "spo2" library stub
    public Spo2Thread spo2Thread; // A thread to acquire data from SpO2 device

    public void onCreate(Bundle savedInstanceState){
        spo2Thread = new Spo2Thread(); // Create the thread and execute
        spo2Thread.start();

        class Spo2Thread extends Thread {
            public void run(){
                while(true){
                    String message = getSpO2(); // Call to acquire pulse-rate and oxygenation
                    pulse=message.substring(0,4); // Pulse data extraction
                    spo2=message.substring(5); // Oxygenation data extraction
                    sendData(new DataMessage(
                        DataMessage.MsgType.SPO2, message)); // Send data to a remote station
                }
            }
        }
    }
};
```

**Figure 4. Skeleton code used to handle the SPO<sub>2</sub> sensor and data**

After a byte sequence is received by reading the serial port, biomedical signals are extracted from the sequence. The data is packed into an object and sent to the remote station via wireless communication. Currently, we use the wireless LAN (IEEE802.11) protocol. The object is prepared so the type and value of the data are integrated into an object. The output channel to the remote station is established as a socket, which is provided by the standard Java library.

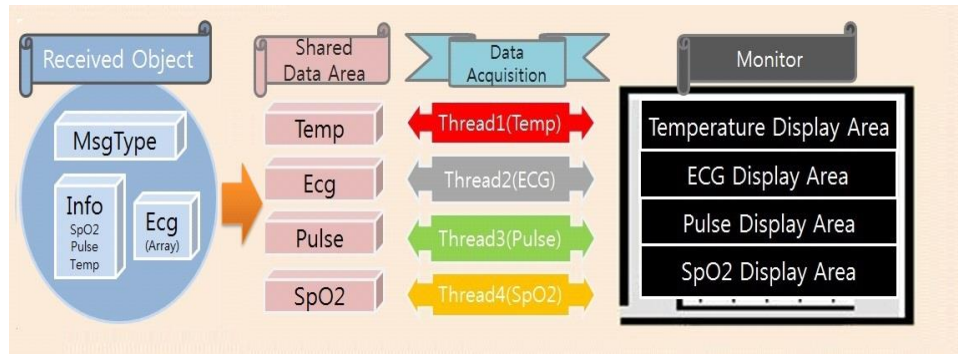
The socket to the remote station is shared by multiple threads that send out data concurrently so we need to protect the channel from possible race conditions. The socket's write function is thread-safe, not like the C implementation, so we may call the write function freely from several threads without worrying about possible race conditions.

### 3.3. Display module

The data sent from the embedded module are received by the remote site in the form of an object, and the server module extracts the type and value of the biomedical data. The structure of the biomedical data and their transitions are shown in Figure 5.

To display several types of biomedical data on the monitor screen, we introduced a worker thread for each display because the display rate of each signal was different. Therefore, we used a thread that extracts data from received objects and stores it in a shared data area. Next, each worker thread picks up appropriate data and displays it in its own area at its own pace. Another important feature of this module is that a warning signal appears on the screen and an alarm goes off if a value is outside the normal (predefined) range.





**Figure 5. Message structure and its transitions**

#### 4. Developing a prototype WIPAMS

We developed a prototype of WIPAMS. The specification of the embedded board is shown in Table 2. We ported Android OS onto the board using an image provided by the hardware manufacturer. The module obtained its power supply from an integrated battery we prepared. The casing was crafted using a rapid prototype machine. Figure 6 shows the inner and outer appearance of the WIPAMS prototype. The provision of battery power made it a truly portable device. Figure 7 shows the display of the remote machine. The same display was also available on the portable device. We could connect a LCD display to the device and view the same display as that seen by the remote machine.

**Table 2. Specification of the WIPAMS board (Odroid-X)**

Processor	Samsung Exynos4412 Cortex-A9 Quad Core 1.4GHz with 1MBL2 cache
Memory	1GB LP-DDR2 880Mega data rate
Video Out	Micro HDMI connector / RGB-24bit LCD interface port
LAN	10/100Mbps Ethernet with RJ-45 Jack (Auto-MDIX support)
USB2.0 Host	High speed standard A type connector 6 ports
USB2.0 Device	ADB/Mass storage(Micro USB)
Display	HDMI monitor / LCD panel with RGB or LVDS interface
Storage	Full size SDHC Card Slot
Power	5V 2A Power
System Software	u-boot 2010. 12, Kernel 3.0.15, Android4.0.x(ICS)



**Figure 6. WIPAMS prototype**



**Figure 7. Display of the remote station**



## 5. Conclusions

WIPAMS is a patient monitoring system. It collects the vital signs of patients using biomedical sensors and processes them so they can be interpreted easily by medical personnel. Unlike conventional PMs, WIPAMS is small, portable, battery-operated, and based on wireless communication. Thus, our device could be useful in a u-Healthcare environment where the remote monitoring of patients is essential.

We produced a prototype of WIPAMS and its performance was satisfactory compared with conventional systems. Real-time vital signs were displayed smoothly without errors. The immediate transfer of data occurred when the patient pushed a button.

In future developments, we plan to integrate a positioning capability in WIPAMS. Various value-added services may be possible if we can locate a patient who is carrying a WIPAMS. For example, we may identify the whereabouts of a patient who is moving around inside the hospital building. Thus, medical assistance could be provided immediately if there is an emergency.

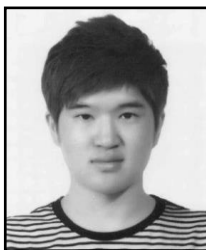
## Acknowledgements

This work was supported by Soonchunhyang University.

## References

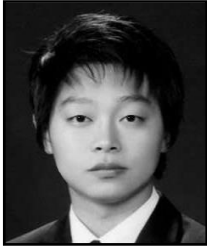
- [1] J. Enderle, S. Blanchard and J. Bronzino, in Introduction to Biomedical Engineering, Edited J. Bronzino, Elsevier Academic Press, London (2005), Second Edition, pp. 505-548.
- [2] R. Shahriyar, M. F. Bari, G. Kundu, S. I. Ahamed and M. M. Akbar, "Intelligent Mobile Health Monitoring System (IMHMS)", Journal of Control and Automation 2, 3 (2009).
- [3] S. W. Franklin and S. E. Rajan, "Personal Area Network for Biomedical Monitoring Systems Using Human Body as a Transmission Medium", Journal of Bio-Science and Bio-Technology, vol. 2, no. 2, (2010).
- [4] B. -W. Min, "Improvement of the Personalized Mobile U-Health Service System", International Journal of Multimedia and Ubiquitous Engineering, vol. 7, no. 1, (2012).
- [5] R. S. Tolentino and S. Park, "A Study on U-Healthcare System for Patient Information Management over Ubiquitous Medical Sensor Networks", International Journal of Advanced Science and Technology, vol. 18, no. 1, (2010).
- [6] Android OS, <http://developer.android.com/about/index.html>.
- [7] Pulse-Oximeter, HyBus, <http://edu.hybus.net/goods/>.
- [8] Thermometer, Hubdic, <http://www.hubdic.co.kr/e-web/html/babycare.html>.
- [9] ECG kit, PhySioLab, [http://www.physiolab.co.kr/Product\\_module\\_12bd.aspx](http://www.physiolab.co.kr/Product_module_12bd.aspx).
- [10] D. E. Simon, "An Embedded Software Primer", Addison Wesley, Boston, (1999).
- [11] Odroid-x, Hardkernel, [http://www.hardkernel.com/renewal\\_2011/products/](http://www.hardkernel.com/renewal_2011/products/).

## Authors



**Jungmuk Kang**

He is currently a senior in the Department of Medical IT Engineering at SoonChunHyang University, South Korea. His research interests include embedded systems, wireless sensor networks, and u-Healthcare systems.



### **Sungil Yoo**

He received his B.S. degree in Computer Science and M.S. degree in Medical-IT Engineering from SoonChunHyang University, South Korea, in 2009 and 2011, respectively. Currently, he is pursuing his Ph.D. degree in Medical IT Engineering. He is a research and teaching assistant at the university. His research interests include ubiquitous computing, wireless sensor networks, healthcare systems, and mobile applications.



### **Dongik Oh**

He is currently a professor of Medical IT Engineering at SoonChunHyang University, South Korea. He received his B.S. degree in Computer Science from The City University of New York (College of Staten Island), USA in 1985; M.S. degree in Computer Science from Florida State University, USA in 1989; and Ph.D. degree in Computer Science from Florida State University, USA in 1997. His current area of research includes embedded systems, ubiquitous computing, healthcare systems, and mobile programming.