Monitoring Systems for Embedded Equipment in Ubiquitous Environments

Ji-Hye Bae*, Hee-Kuk Kang*, Yoon-Young Park, and Jung-Ho Park****

Abstract: Accurate and efficient monitoring of dynamically changing environments is one of the most important requirements for ubiquitous network environments. Ubiquitous computing provides intelligent environments which are aware of spatial conditions and can provide timely and useful information to users or devices. Also, the growth of embedded systems and wireless communication technology has made it possible for sensor network environments to develop on a large scale and at low-cost. In this paper, we present the design and implementation of a monitoring system that collects, analyzes, and controls the status information of each sensor, following sensor data extracted from each sensor node. The monitoring system adopts Web technology for the implementation of a simple but efficient user interface that allows an operator to visualize any of the processes, elements, or related information in a convenient graphic form.

Keywords: Ubiquitous Computing, Monitoring System, Sensor Network

1. Introduction

Embedded Systems are systems that contain embedded hardware and software designed to control special-purpose hardware that operates as a component of a large system [1]. They are utilized in a variety of specialized control systems from major power plants to those used for vehicle and house systems. The wide variety of control systems for complex industrial objects is an important application of embedded systems. Embedded systems are usually a complex combination of hardware and software modules with a built-in operational environment, a close architecture, specific interfaces, and a unique organization of external and internal data flow. For large industrial applications such as a power turbine, these control systems are often implemented as multiprocessor distributed systems. With the recent growth of embedded systems, the studies conducted on ubiquitous computing are gaining attention.

Ubiquitous computing is able to perceive its surroundings and provide an intellectual environment that offers timely and useful information to users and user devices. This is made possible by composing a co-network of ultra-small embedded computers with integrated communication devices positioned in the surrounding objects and environments. This means that although users are physically at a distance, they can be provided with specific and status-perceived information services by tracing or recognizing the situational variations of a wanted time and space. The growth of ubiquitous computing technology and wireless communication technology has enabled the development of large-scale and low-cost sensor network environments [5, 6].

With the development of ubiquitous computing technologies and the subsequent maturity of these technologies, wireless sensor networks have become available for connecting large amounts of data from various sensor devices that are distributed over a wide area. These sensor devices can now be connected to application servers through a wireless sensor network consisting of small devices called sensor nodes that can sense data through various sensors. It also has a processor to format the sensor data and a wireless transmission system to transfer the formatted data and any required control signals from the sensors to the main server or vice versa. Unlike previous networks that acted solely as a communication medium, the purpose of a sensor network is to collect information about the sensor environments. The ongoing progress and research activity in wireless communication technologies have resulted in low-priced, ultra-minimal, and low-powered devices, which have facilitated the development of several wireless sensor networks to suit the requirements of many communities. Wireless sensor networks are now used extensively in the development of applications in a wide range of fields such as science, medicine, the military, and commerce [7, 8].

This paper presents our proposed monitoring system architecture that collects, analyzes, and displays the information state from the wireless sensors. It also describes our communication mechanism to store and transfer large amounts of status and efficient monitoring systems, which uses a remote shell to collect the status data of embedded devices and to send control data to hub nodes and sensor nodes.

Embedded systems should present a sensor object's

Manuscript received February 10, 2006; accepted March 3, 2006.
This research was supported by the MIC (Ministry of Information and Communication), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute of Information Technology Assessment)(IITA-2005-C1090-0502-0031)
Corresponding Author: Ji-Hye Bae
* Department of Computer Science, Graduate School, Sun Moon University, Asan, Korea ({angdoo98, comhero}@sunmoon.ac.kr)
** Division of Computer and Information Sciences, Sun Moon University, Asan, Korea ({yypark, jhpark}@sunmoon.ac.kr)

information state to an operator and also react promptly to instructions from an operator. A user interface is necessary for an embedded system to display information and to interact with operational personnel. The development of these complex user interfaces can be both a time- and labor-consuming task. Hardware-dependent components and unique software modules are often used in the development of such interfaces, and problems can also occur in the timely distribution of software to client workplaces. Since the worldwide number of embedded systems is increasing rapidly, new approaches are needed for embedded systems operations, especially in a distributed environment. To solve this problem, our monitoring system has adopted Web technology for the implementation of a simple but efficient user interface.

Web technology is an efficient and cost-effective technology for embedded systems interface development and for remote object monitoring and management. It allows for various standardized templates for user interface organization including formatted textual display and examples of various types of tables, lists, the display of graphic images (for example, of a block of the controlled entity), and other required formats. Our monitoring system interface allows any process, element, or other form of information to be visualized by the operator in a convenient and easily interpreted form. Therefore, system operational personnel do not have to be computer experts to use the system. Furthermore, it presents a unified interface to the operator and affords that person the opportunity to control complex devices over an intranet or the Internet.

In this paper, we also developed a monitoring system using a kernel wrapper, which is a software module that enables us to monitor the state information of any embedded device without any modification to the kernel. By employing this wrapper, we can increase the portability of our system to other platforms and also reduce the time required to write software-monitoring modules.

The remainder of this paper is structured as follows. In the next section, we will introduce related works on embedded monitoring systems and sensor network middleware and data services in the sensor network. Then, in section 3, we describe the design to monitor the information state from sensor nodes, while in section 4 we demonstrate this design and implementation of the monitoring system architecture. In section 5, we will discuss how to solve problems in our design and implementation, and present future research regarding our approach to the sensor network environment.

2. Related Work

In this section, we introduce existing works on sensor network middleware and data services in the sensor network that relate to this paper. Also, we describe several cases which touch on the monitoring system and the existing monitoring tools used to evaluate embedded kernel performance in ubiquitous environments. Ongoing research into the provision of wide-area data services has advanced in the field of sensor networks. Sensor data management is considered important because sensor data has to be formatted into a proper form through server and hub nodes, and is displayed and monitored by users.

The most well-known research system regarding data-centric middleware, MiLAN, is being developed as a sensor network middleware for smart medical homes for managing health conditions at the Univ. of Rochester [2].

PADS (Power Aware Distributed System) is being developed as a sensor network middleware at UCLA/USC. This system implements suitable methods for managing organic energy management functions among CPU, RF modules and sensor modules for energy reduction. It manipulates dynamic CPU voltage and RF modulation scales at the proper time with RTOS schedule methods, and then accomplishes program execution, sensor management, and message transmission using minimal energy [3].

An example of a project using monitoring in the applicable field of sensor networks is Habitat Monitoring. In this project, an in-depth study of the application of wireless sensor networks to real-world habitat monitoring was undertaken. The project also develops a specific habitat monitoring application that is largely representative of the domain. It presents a collection of requirements, constraints, and guidelines that serve as the basis for a general sensor network architecture for many such applications. The College of the Atlantic (COA) is field testing for a habitat monitoring project, and is conducting ongoing field research programs on several remote islands. Great Duck Island (GDI) has well established on-site infrastructure and logistical support [4].

One of the monitoring systems in the ubiquitous environments, "Tornado", is an integrated development environment that has embedded operations, development environments, and execution environments, all of which are being used in real-time applications simultaneously; it has various development tools and provides an easy and consistent development environment to users by connecting each tool. Also, these tools are suitable for use in several development phases and they support several different target systems. Tornado uses VxWorks as its real-time operating system (RTOS). This RTOS has a very small kernel but it still provides multitasking environments, inter-process communications, and process synchronization. To communicate between a host machine and a target machine, Tornado provides various means of communication such as Ethernet, serial communication, ICE (in circuit emulator), or ROM emulator [9]. However, Tornado is a very expensive solution and it is difficult for a novice to learn the basic skills necessary to use the internal functions.

Qplus-P Esto is an IDE (Integrated Development Environment) for developing application software that executes on Qplus-P, an embedded operating system. Qplus-P is a process-based operating system founded on embedded Linux. Esto (Embedded Systems Tool) provides an IDE with a GUI (Graphic User Interface) for both Linux

and Windows system hosts, and enables embedded application program developers to perform several functions such as coding, compilation, execution, debugging, and monitoring on a single host platform [10]. Therefore, users of this IDE can develop new application programs conveniently. Esto includes a library, a target agent, and a target application program based on the Qplus Embedded Linux RTOS for the target. Furthermore, it also has a wide variety of application development tools such as a host agent based on the Linux/Windows operating system, a target builder, a cross compile tool-chain, a project manager, a remote shell, a remote debugger, a remote monitor, and an instrument for measuring power consumption on the host side. However, it would be desirable for Esto to have a user interface that is implemented by Eclipse. Therefore, we would need a great amount of expertise in Qplus-embedded Linux and in constructing Eclipse applications.

The Momentics IDE (Integrated Development Environment) for Neutrino RTOS from QNX supports several host platforms such as Windows, Linux, and Solaris. Compared to other OSs, Momentics is powerfully flexible and can produce RTOS images with the Neutrino Kernel, which has only one multiple-thread application process for very small-embedded systems. Also, its process manager can execute several application programs and it can execute on a distributed network of very large symmetric multiprocessing (SMP) clusters. QNX Momentics is a generalized package that integrates both enforced productivity methodology and analysis tools into one Eclipse-based IDE [11, 12]. However, QNX Momentics is too expensive and inflexible when compared to general purpose embedded Linux.

3. The Design of Our Monitoring System for Embedded Equipment

In this section, we describe the architecture of our monitoring system, which collects data obtained from wireless sensor nodes in ubiquitous network environments.

Sensor nodes consist of sensor, data processing, and communication components. When these components are connected by a network, they form a sensor network. A sensor network is composed of several sensors that are densely deployed either inside the nodes or very close to them [13]. Since we have used RF communications to communicate among the various and potentially vast number of sensor nodes, it is necessary to monitor them to determine the reliability and the fault tolerance of a sensor network and to conduct a network traffic analysis of a sensor network.

3.1 The Architecture of a Monitoring System in *Ubiquitous Computing Environments*

Ubiquitous network environments are divided into servers for monitoring and control, targets that are the objects to be monitored, and the clients who use the system Generally speaking, the targets are implemented as embedded equipment, although the servers can be implemented either as embedded equipment or as a separate server system. Fig. 1 shows the basic architecture of a monitoring system.

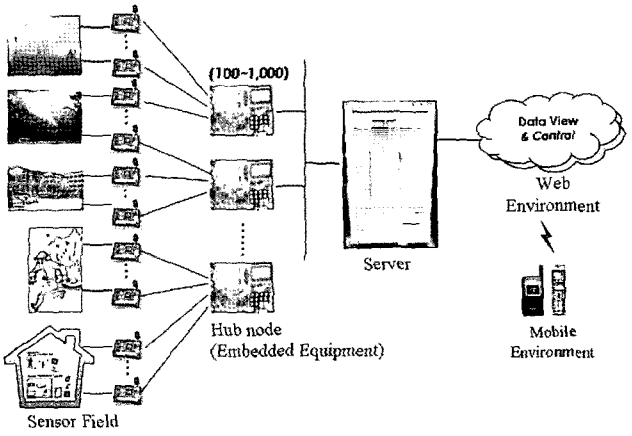


Fig. 1. The Basic Architecture of a Monitoring System

There are often several types of sensor nodes that are usually embedded into many kinds of devices. These sensor nodes collect the raw data. Hub nodes collect the raw data from the sensor nodes and deliver it to the servers. The servers receive the data from the hub nodes and send control data back to the hub nodes. The hub nodes, which are usually implemented as embedded equipment, communicate with the sensor nodes using RF communications. The raw data containing the status information of a sensor node is delivered to a hub node and transformed into the proper data format for onward transmission. A common Network File System (NFS) is used to connect the hubs and the server. To measure the performance of the hub node's internal kernel, we developed application programs to evaluate the status of each process. The collected information is provided to users through a GUI in the server system. Our monitoring system allows users to manage and control monitoring information in mobile environments.

The characteristics of the sensor nodes, the hub nodes, and the server are as follows:

- Sensor Node: It gathers various types of raw data from numerous sensors and delivers them to the hub nodes.
- Hub Node: It is an embedded equipment that gathers data delivered from sensor nodes, reformats it, and delivers it to the server. It also mediates between the server and the sensor nodes.
- Server: It collects, stores, and analyzes the status information from the hub nodes and houses the web server for monitoring and controlling this information.

Fig. 2 describes the connection structure of the monitoring components, the server, the hub nodes, and the sensor nodes in a ubiquitous network environment. The hub nodes communicate with the server in many ways using a target

shell. The server retrieves and controls information from the hub nodes by using a remote shell implemented with SNMP (Simple Network Management Protocol). These types of shells have an agent for communication, which performs all of the communication that is conducted between the server and the hub nodes and is connected using NFS and SNMP. Our system uses NFS and SNMP to deliver sensor information to a server. In our system, we implemented several tracing devices and SNMP as important middleware to enable us to perform a performance evaluation of the kernel in the hub nodes. With this middleware, the server can collect a variety of information from the hub nodes to measure the performance of an embedded hub kernel. Sensor nodes in the sensor fields use RF communications to communicate among themselves and the hub nodes.

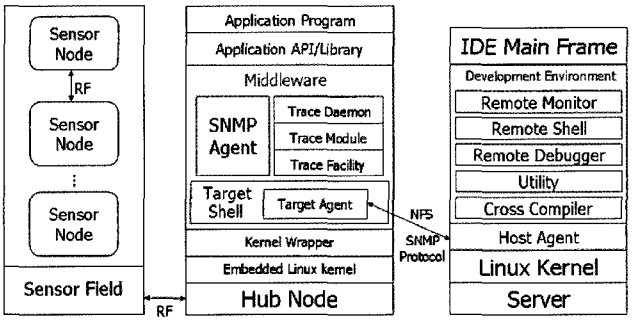


Fig. 2. The Network Architecture of the Monitoring Components

3.2 The Basic Design of our Monitoring System

In this section, we present the basic design of our monitoring system. Our monitoring system displays both the collected events data and the results of any event analysis that have been requested on any running embedded equipment or sensor nodes. Several events and the information that has been collected from the hub nodes and sensor nodes are stored in the form of files on NFS server systems.

By using our monitoring system's hypertext references, an operator can navigate simply between pages, data input forms, and the specifications of communication protocols to a server. Thus, an operator can easily control any entity in the system. Our monitoring system's user interface allows the operator to conveniently visualize any process, element, or other aspect of system information without having to be a computer expert. Furthermore, it provides the operator with a unified interface through which he or she can easily manipulate complex systems over an intranet or the Internet.

Many events and information collected from the hub nodes and the sensor nodes are stored in the form of files on NFS server systems.

Fig. 3 shows the design plan for the monitoring system. There are several instances of embedded equipment items (target nodes) that are all treated as separate objects in our monitoring system. The embedded equipment sends the monitoring information on various systems such as Process Memory, CPU, Traffic, Command, Kernel Trace, and Sensor to the server.

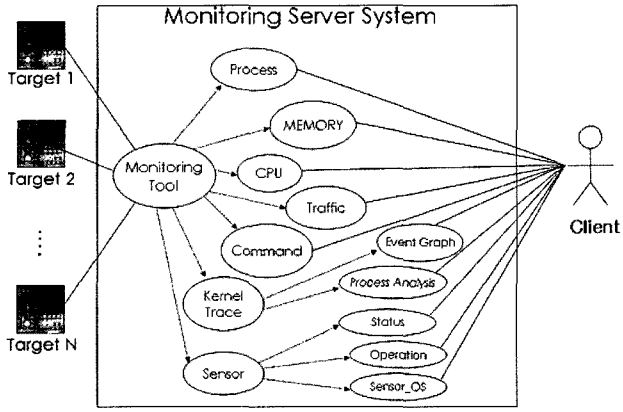


Fig. 3. The Design Plan of the Monitoring System

The "Kernel Trace" is composed of an "Event Graph" and a "Process Analysis." The "Sensor" is composed of Status, Operation, and the Sensor OS. All data is presented to the operator through the Web interface on the server.

3.3 The Kernel Wrapper of the Monitoring System

This section describes the kernel wrapper mechanism employed to build the applications that are necessary to obtain the kernel information in an easy and efficient manner.

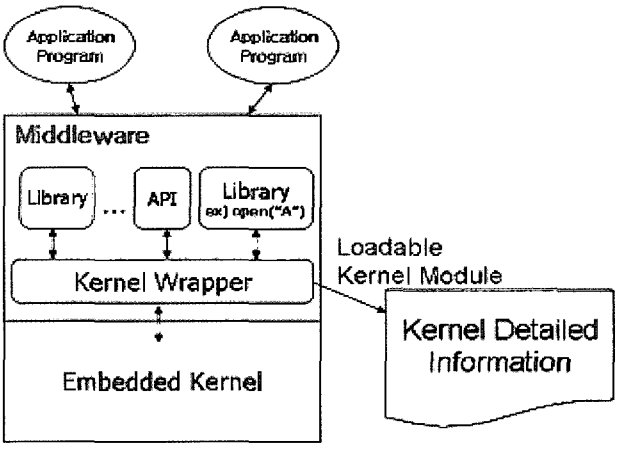


Fig. 4. The Architecture of the Kernel Wrapper

Fig. 4 describes the architecture of the kernel wrapper used for gathering and monitoring a variety of data about the embedded kernel. Loadable modules or libraries that need to communicate with the embedded kernel do so using the kernel wrapper. It gathers information about the kernel such as the frequency of primitive operations and the execution time of a particular process. We can extract any piece of kernel monitoring information using this kernel wrapper without modifying and recompiling the embedded kernel. By using the concept of a loadable

kernel module, we can construct the kernel wrapper in the form of a dynamic library that we can then easily add to the embedded kernel even though we do not have any detailed knowledge about the embedded kernel itself. This allows us to conveniently extract a variety of modules constructed by the kernel wrapper call *sys_our_open()* that is implemented by the kernel wrapper instead of the general system call *sys_open()*. Therefore, we can trace which files a user has opened.

4. Implementation

In this section, we describe the implementation details of our monitoring system and the results that we obtained using our system.

4.1 The Monitoring Information of Embedded Equipment

Since there are various types of monitoring data obtained by embedded equipment, we divided the monitoring data into three categories: system resource, network resource, and sensor resource [15].

4.1.1 System Resource

"System Resource" data is related to information on the performance evaluation of the embedded kernel. Its five types of data are:

- PROCESS: Information about the running processes;
- CPU: Information about CPU usage and CPU idle time;
- MEMORY: Memory information used by operators or memory resident daemon processes;
- COMMAND: Information obtained by a remote shell. A remote shell consists of many functions that are executable commands used to obtain information related to the system, such as directory information, running processes, memory usage, and disk contents. It has a simple menu structure through which we can obtain many pieces of information about embedded equipment. The information that can be obtained by a remote shell menu includes the following:

- Directory information shows information about the files in a directory. It includes information about hidden files and detailed information such as file format, access authority, and file size.
- Process Information shows information about a running process including environment variables that are related to the relevant process and the inheritance relationships among relevant variables.
- Memory Information shows a menu that can be used to display the memory status of system outputs.
- Disk Information; there is no disk in the embedded equipment but data is stored on the file system of the server by the NFS. Therefore, this item shows a menu that represents disk information, which describes detailed file information in a directory as well as basic directory information.

- KERNEL TRACE: This is the data for the kernel performance evaluation that is created during a kernel trace. We implemented several application programs to trace the internal kernel system and to evaluate the performance of the embedded kernel. Kernel patches were also used for tracing. In general, these application programs were applied based on the */proc* file system. The purpose of the kernel evaluation is to estimate both the side effects and the interference problem that might occur when the hub nodes and server receive a large amount of sensor data simultaneously. The "Kernel Trace" includes the following components:
  The Event Graph displays processes or system behaviors that are generated by the kernel. Fig. 5 shows the number of system calls with a time unit of μs (microseconds). The left side of the system call viewer shows all of the running processes and the right side displays the occurrences of a particular system call (e.g. read() and write()) that are issued by these processes. There is a system tracing utility for Linux, Linux Trace Toolkit (LTT) [1], which is used for analyzing a subset of process execution and for recording some important system events. The kernel-tracing tool in LTT is based on the GTK library. However, in our system we based the event viewer on GDI Plus from Microsoft. Since our event viewer is based on a Windows platform, we implemented the viewer with ActiveX technology. Fig. 6 consists of magnified figures of Fig. 5. It shows the running processes list, a graph that is used by these processes, and system call information respectively. This system has many useful functions including the characteristics of a simple interface, an easily recognizable graph, easy installation, and originality. In (Fig. 6), the running processes list includes the process id and time-ordered the process name, and the system calls used by these processes are presented in a graph. The menu of the "System Call Info" includes the information about current time, time length, and each system call name. The description of each system call is shown by the menu of the "System Call Desc" in (Fig. 5).

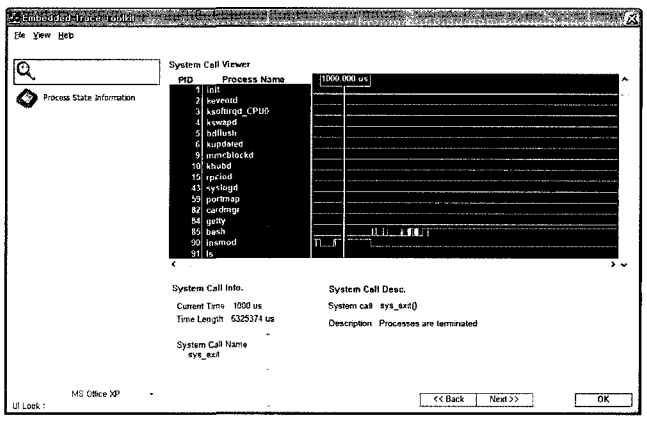


Fig. 5. Event Graph of an Embedded Kernel System

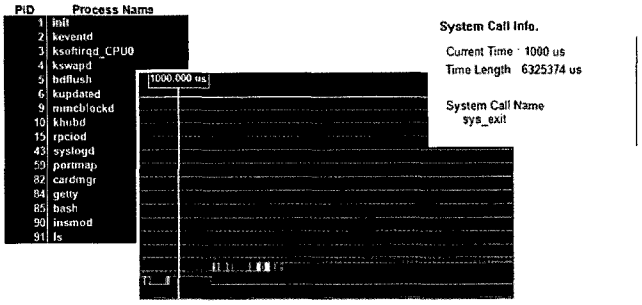


Fig. 6. A Magnified Figure of Fig.5

- The Process Analysis provides the per-process statistics and system statistics. It shows the CPU time, the execution time for a running application program, the I/O waiting time, and information about the system calls by each application program.

4.1.2 Network Resource

Network resource has a single type of data called TRAFFIC, which analyzes network traffic.

- TRAFFIC: This data consists of information about the network traffic that is delivered from the server to the embedded equipment. We can calculate the amount of packets that are composed of received packets, transmitted packets, error packets, dropped packets, collision packets, etc.

4.1.3 Sensor Resource

Sensor resource consists of the sensor data that are gathered from the sensors. In our system, we designed the hub nodes to collect the sensor information. Then, our monitoring user interface displays this information in the form of a graph. The information from the sensor nodes is as follows:

Table 1. Monitoring Information and Description of the Embedded Equipment

Resources	Components	Function	Descriptions
System Resources	PROCESS	PROC Info	Process information about USER, PID. PPID, ST, NAME, CPU, VMEM, TTY, TIME, CMD, PRI, ADDR
		PROC Time	Process analysis time
	CPU	CPU Usage Rate	CPU usage rate
		CPU Usage Time	CPU usage time (Maximum, Average, Present)
		CPU Idle Time	The amount of time spent in idle process (Maximum, Average, Present)
		CPU Info	CPU information
	MEMORY	Total Memory Size	Total memory size installed in the system
		Cached Memory	Cached memory size in run time
		Available Memory	Available memory (Maximum, Average, Present)
		MEM Info	Memory information about memory pages allocated and memory pages free.
	COMMAND	System	Information about the system obtained by a remote shell (e.g. Hostname, System, Kernel information, Date information)
		General Command	Information obtained by general commands (ls, ls -al, ps -ef, free, du -a, etc.) using a remote shell
	Kernel Trace	Event Graph	Traced kernel information (running processes, system behavior)
		Process Analysis	Traced kernel information related to a process (process statistics, system statistics)
Network Resources	TRAFFIC	Received Packet	Total number of packets received by a network device
		Transmitted Packet	Total number of packets transmitted by a network device
		Error Packet	Total number of errors detected by a device driver
		Dropped Packet	Total number of packets dropped by a device driver
		Collision Packet	Number of collisions detected on the interface
Sensor Resources	Sensing Data	Status	Information about the status of sensor nodes
		Operation	Information about the behavior of sensor nodes
		OS	Information about the sensor operating system of sensor nodes

- Sensor Status describes the current status of the various sensors connected to the hub nodes.
- Sensor Operation indicates the behavior status of various sensors. Operators can directly control the sensors through the GUI.
- Sensor OS (Kernel Status) shows any necessary data concerning the kernel that is built into the sensor operating system.

Table 1 summarizes these various types of monitoring information [14, 15].

4.2 The Display of Monitoring Information

Fig. 7 shows various types of monitoring information about embedded equipment generated by our monitoring system. We placed the menu frame on the left side to facilitate the use of the many functions related to process, CPU, memory, traffic, command, kernel tracing, and sensor data from the embedded equipment.

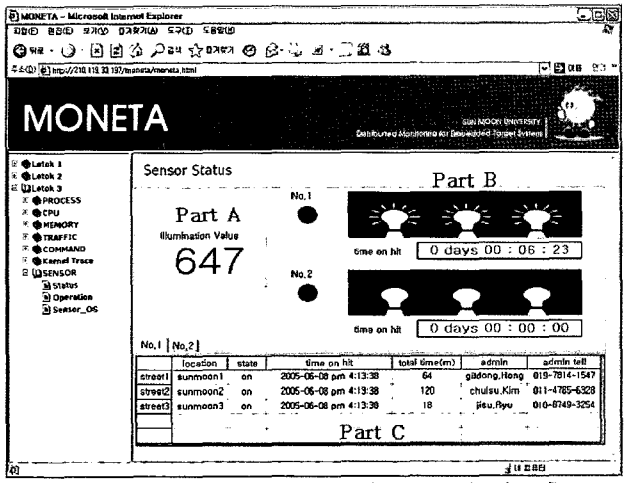


Fig. 7. The Sensor Status Display of the Monitoring System

In (Fig. 7), we can see the required information from the monitored embedded equipment. We can also obtain the status information of the sensors connected to the hub nodes. In this example screen, we can see the control information for street lamps with illumination sensors. If the value of illumination exceeds some pre-selected value as shown in Part A, the button in the No.1 area is turned on as shown in Part B. Then, the operation time starts to be recorded as shown in Part C. Detailed information on the street lamps, such as the position, the status, the operation time, and the manager of the street lamps is displayed in Part C.

5. Conclusion

Accurate and efficient monitoring of dynamically changing environments is one of the most important requirements for ubiquitous network environments. To exploit these ubiquitous environments, we designed and implemented a monitoring system that can obtain sensor data transmitted from sensor nodes to the hub nodes in embedded equipment, and also implemented some application programs on a server system to control many hub nodes. Our monitoring system adopts Web technology for the implementation of a simple but efficient user interface. Web technology is an efficient and cost-effective technology for both embedded systems interface development and remote object monitoring and management. Our system interface allows an operator to conveniently visualize any process, element, or other piece of information using a GUI. This gives a clear and simple presentation of the data to an operator who does not have to be a computer expert, and who can receive unified access to control devices and operate complex control systems via an intranet or the Internet. This enables the operator to monitor the control system anytime and anywhere using any available PC or any mobile computer equipment such as a PDA as long they are connected to a network. We developed our monitoring system using a kernel wrapper mechanism. A wrapper is a software module or library that enables us to monitor the information state of several embedded devices without modifying the relevant kernel. By employing this type of wrapping method, we were able to increase the portability of our monitoring system to other platforms while reducing the time required to write a monitoring software module. For our future work, we hope to develop debugging tools that will make it possible to find and fix some software bugs and system errors. We also plan to develop an IDE for our system.

References

[1] Ji-Hye Bae, Yoon-Young Park, Jeong-Bae Lee, Sung-Hee Choi, Chae-deok Lim, "A Study on the Design of the Monitoring Architecture for Embedded Kernels based on LTT," Proceedings of 4th Asia Pacific International Symposium on Information Technology, Gold Coast, Australia, pp.68~71, Jan., 2005.

[2] W.B.Heinzelman, A.L.Murphy, H.S.Carvalho, and M.A.Perillo, "Middleware to Support Sensor Network Applications," IEEE Network, Vol.18, No.1, pp. 68~71, Jan., 2005.

[3] Power Aware Distributed System, Homepage, http:// pads.east.isi.edu/

[4] A.Mainwaring, J.Plastre, R.Szewczyk, D.Culler and J.Anderson, "Wireless Sensor Networks for Habitat Monitoring," In ACM International Workshop on Wireless Sensor Networks and Application (WSNA '02), Atlanta, GA, USA, Sept., 2002.

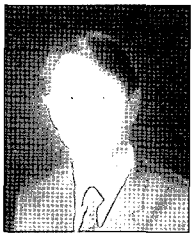
[5] G.Chen and D.Kotz, "A Survey of Context-Aware Mobile Computing Research," Dartmouth Computer Science Tech Report TR2000-381, 2000.

[6] M.Satyanarayanan, "Pervasive computing: vision and

challenges," IEEE Personal Communications, pp.10-17, Aug., 2001.
[7] D.Y.Kim, "Sensor Networks(v) Sensor Network Middleware," FA Journal, Jul., 2 004.
[8] K.Romer, O.Kasten, and F.Mattern, "Middleware Challenges for Wireless Sensor Networks," ACM SIGMOBILE Mobile Computing and Communications Review, Vol.6, No.4, Oct., 2002.
[9] Yoon-Young Park, A Study on the Monitoring Model of Distributed Objects, ETRI, Korea, 2000.
[10] Embedded Software Technology Center, Q+ Esto Manual, ETRI, Korea, 2003.
[11] QNX Homepage, http://www.qnx.co.kr
[12] Eclipse Homepage, http://www.eclipse.org
[13] Ian F.Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci, "A Survey on Sensor Networks," IEEE Communications Magazine, pp.102~114, 2002.
[14] Hyo-Sung Kang, Jong-Mu Choi, Jai-Hoon Kim, Young-Bae Go, "Agent-Based Embedded Monitoring System for Ubiquitous Networks Environments," Proc. of the 2004 International Conference on Parallel and Distributed Processing Techniques and Application (PDPTA 04), Las Vegas, USA, Jun., 2004.
[15] Ji-Hye Bae, Yoon-Young Park, Jung-Ho Park, "A Study on the Design of the Monitoring Architecture for Embedded Kernels based on LTT," International Journal of Information Processing Systems (IJIPS), Vol.1, No.1, pp.1~8, Dec., 2005.
[16] Ji-Hye Bae, Hee-Kuk Kang, John Y.Kim, Yoon-Young Park, "Monitoring Systems for Embedded Equipments in Ubiquitous Environments," Proceedings of 5th Asia Pacific International Symposium on Information Technology, Hangzhou, China, pp.123~126, Jan., 2006.

Ji-Hye Bae
She is a post-graduate Ph.D. student in the Department of Computer Science, graduate school, Sun Moon University, Korea. She received her MS degree in computer science at Sun Moon University in 2005. She majored in embedded systems and her current research interests are ubiquitous computing and sensor network environments.
Department of Computer Science, Graduate School, Sun Moon University, Asan, Chungnam, 336-708, Korea

Hee-Kuk Kang
He is a post-graduate Ph.D. student in the Department of Computer Science, graduate school, Sun Moon University, Korea. He received his MS degree in computer science at Sun Moon University in 1999. He was in charge of the department of developing web technologies at Julynet Co., Ltd. from 1999 to 2004. He majored in distributed systems and his current research interests are ubiquitous computing and sensor network environments.
Department of Computer Science, Graduate School, Sun Moon University, Asan, Chungnam, 336-708, Korea



Yoon-Young Park
He is a professor in the Faculty of Computer and Information Sciences, Sun Moon University, Korea. He received his MS and Ph.D. degrees in computer science from Seoul National University in 1985 and 1994, respectively. His main majors are distributed operating systems and real time systems. His recent research includes sensor networks and ubiquitous computing. He is a member of KIPS (Korea Information Processing Society).
Division of Computer and Information Sciences, Sun Moon University, Asan, Chungnam, 336-708, Korea

Jung-Ho Park
He is a professor in the Faculty of Computer and Information Sciences, Sun Moon University, Korea. He received his MS and Ph.D. degrees in computer science from Osaka University in 1987 and 1990 respectively. His current research interests are distributed algorithms and electronic commerce. Recently, he has developed an interest and takes an active part in Internet Ethics. He is a member and vice president of KIPS (Korea Information Processing Society).
Division of Computer and Information Sciences, Sun Moon University, Asan, Chungnam, 336-708, Korea