센서노드의 센서 투명성을 지원하는 TinyOS의 확장

소선섭* · 은성배** · 김병호***

Design and Implementation of TinyOS Supporting Sensor Transparency of Sensor Nodes

Sun Sup So* · Seong Bae Eun** · Byungho Kim***

이 논문은 2008년도 정부(교육인적자원부)의 재원으로 한국대학교육협의회 대학교류 연구비 지원과 공주대학교 연구년 사업에 의하여 연구되었음

요 약

본 논문에서는 범용 운영체제에서 제공하는 디바이스 투명성을 센서노드 운영체제에 적용한 센서 투명성 아키텍처를 제안하였다. 센서 투명성을 지원하기 위한 표준 API와 센서 디바이스 추상화를 설계하고 TinyOS 운영체제에서 구현하였다. 본 논문에서 제안한 센서 투명성 지원 센서노드 운영체제를 사용하면 응용 개발자는 운영체제에서 제공되는 표준 API를 통해 센서 디바이스에 독립적으로 응용 프로그램을 개발할 수 있고, 센서 디바이스 공급자 또한 표준화된 하드웨어 인터페이스와 HAL 인터페이스를 통해 센서노드 하드웨어 플랫폼에 독립적으로 센서 디바이스 드라이버를 개발하고 공급할 수 있다.

ABSTRACT

In this paper, we proposed an architecture for supporting sensor transparency in sensor node operating systems, design the standard APIs (Application Programming Interfaces) and sensor device abstraction to provide the sensor transparency and implemented the sensor transparency in the TinyOS, the most well known sensor node operating system. With the proposed sensor node operating system which can support the sensor transparency, application developers can develop the target applications independent to each sensor device by using the standard APIs provided by the sensor node operating system and the sensor device manufacturers also can develop sensor device drivers by using the standard hardware interfaces and HAL (Hardware Adaptation Layer) interfaces independent to the specific hardware platform of sensor nodes.

키워드

센서네트워크, 센서노드 운영체제, 센서 투명성, TinyOS

Key word

Sensor Network, Sensor Node Operating System, Sensor Transparency, TinyOS

* 공주대학교 컴퓨터공학부
** 한남대학교 정보통신공학과 (교신저자, sbeun@hannam.ac.kr)
*** 경성대학교 컴퓨터학부

접수일자 : 2010. 04. 19
심사완료일자 : 2010. 08. 16

Ⅰ. 서 론

유비쿼터스 센서네트워크(USN)는 사물에 부착된 센서노드로부터 수집된 데이터를 센서노드 간 무선 네트워크를 통해 사용자에게 전달한다[1]. USN의 핵심요소인 센서노드는 MCU(Micro Controller Unit), RF(Radio Frequency) 모듈, 센서 등 하드웨어와 무선통신 프로그래밍, 저전력 기능, 센서 디바이스 제어 및 응용 프로그램 등의 소프트웨어로 이루어진다. 특히 센서노드 소프트웨어의 개발의 경우, 응용에 따라 센서 하드웨어의 종류와 특성이 다양하고 코드 재사용이 어렵기 때문에 유사한 개발과정이 각 프로젝트마다 반복되고 있다[2].

예를 들어 대표적인 센서노드 운영체제인 TinyOS [3][4]를 기반으로 대기 환경 모니터링을 위한 USN을 개발한다고 할 때 일반적으로 부딪칠 수 있는 문제점을 살펴보자. 먼저 하드웨어 측면으로는, 대기 모니터링을 위한 CO2 센서를 하드웨어 플랫폼에 부착할 때 센서가 요구하는 전원과 플랫폼이 제공하는 전원이 다를 수 있고, 같다 하더라도 전원 연결 인터페이스가 같지 않을 수 있다. 소프트웨어 측면에서는, 운영체제에서 그 센서에 대한 API(Application Programming Interface)를 지원하지 않으면 소프트웨어 개발자는 그 센서의 디바이스 드라이버부터 직접 개발해야 하고, TinyOS 운영체제와 응용프로그램이 확실하게 분리되어 있지 않기 때문에 개발자는 응용프로그램 개발을 위해 TinyOS 내부 구조를 파악하고 있어야 한다. 이러한 문제의 근본 원인은 센서노드 운영체제에 센서 디바이스와 운영체제를 동적으로 연결하는 기능이 없기 때문이다.

윈도우나 리눅스와 같은 범용 운용체제는 응용 개발자에게 표준 API를 제공함으로써 응용 개발자는 API를 사용하여 해당 디바이스에 독립적으로 응용 프로그램을 개발할 수 있다. 또한 그 표준 API를 준수하는 디바이스 드라이버를 사용함으로써 응용 프로그램의 변경없이 디바이스를 수정하거나 추가할 수 있다. 이를 디바이스 투명성이라고 한다.

본 논문에서는 범용 운영체제의 디바이스 투명성 개념을 센서노드 운영체제에 적용한 센서 투명성 아키텍처를 제안하고, 센서 투명성 지원을 위한 표준 API와 센서 디바이스 추상화를 설계하고 TinyOS 운영체제에서 구현한다.

본 논문에서 제안하는 센서 투명성 지원 센서노드 운영체제를 사용하면 응용 개발자는 운영체제가 제공하는 표준 API를 통해 센서 디바이스에 독립적으로 응용 프로그램을 개발할 수 있다. 이 때 센서 데이터 처리는 센서 디바이스 공급자가 개발한 센서 디바이스 드라이버에 의해 수행되고, 센서 디바이스 공급자 또한 표준화된 하드웨어 인터페이스와 HAL 인터페이스를 통해 센서노드 하드웨어 플랫폼에 독립적으로 센서 디바이스 드라이버를 개발하고 공급할 수 있다.

본 논문의 구성은 다음과 같다. 2장에서 센서노드 운영체체와 센서 투명성 지원 현황에 대하여 살펴보고, 3장에서 센서 투명성을 지원하는 TinyOS 설계에 대하여 기술하고, 4장에서 구현에 대하여 설명한다. 하드웨어 구현과 시스템 통합에 대하여 5장에서 기술하고 6장의 결론으로 맺는다.

Ⅱ. 관련 연구

2.1 센서 투명성

센서노드 운영체제에서 센서 투명성은 디바이스들을 운영체제에 동적으로 연결할 수 있는 기능이다[2]. TinyOS[3], SOS[5], MANTIS[6], Nano-Q+[7] 등 대부분의 기존 센서노드 운영체제들은 센서 투명성을 지원하지 않는데 그로 인한 문제를 세 가지로 요약하면 다음과 같다. 첫째, 운영체제의 하드웨어 플랫폼이 센서 디바이스 연동에 필요한 표준화된 인터페이스를 제공하지 못한다는 점이다. 둘째는 센서노드 운영체제가 응용 개발자에게 표준 API를 제공하지 않는다는 점이다. USN 시스템은 응용에 따라 다양한 종류의 센서들을 필요로 하는데 기존 운영체제들은 센서 디바이스를 추상화하지 않아 각 센서마다 개별 API가 있어야 한다. 셋째, 센서 디바이스 개발 회사가 디바이스 드라이버를 특정 센서노드 하드웨어 플랫폼에 독립적으로 개발하고 공급할 수 있는 방법을 제공하지 못한다는 점이다[2].

2.2 TinyOS

TinyOS[3]는 UC Berkeley 대학에서 개발한 센서노드 운영체제로써 이벤트 기반 상태천이 방식을 채택한 상

태머신 기반 프로그래밍 개념을 사용한 운영체제이다. 각각의 상태는 컴포넌트에 해당되고, 응용프로그램은 컴포넌트 구현에 독립적으로 컴포넌트에 연결된다. 컴포넌트의 명령은 이벤트 처리기에 의해 상태변화를 감지하고 수행한다.

응용프로그램은 컴포넌트 형식으로 구성되고 태스크 큐나 인터럽트 벡터에 저장되었다가 FIFO(First-in First-out) 방식으로 순차적으로 실행되고 태스크 큐가 비면 슬립상태로 전환되어 인터럽트를 기다린다[4].

TinyOS를 활용한 여러 제품과 기술이 개발되었는데, Moteiv사는 TinyOS-2.x를 기반으로 고유의 패키지를 만들었고[8], USN 관련 맥스포의 TIP700CM[9], 휴인스의 UBee430[10] 등 많은 모듈들이 TinyOS를 지원한다. 한국유지관리에서는 TinyOS를 활용하여 시설물 안전관리 USN 시스템과 무선 계측 시스템을 개발하였으며[11], 삼성SDS는 실내용 대규모 USN 네트워크 기술을 개발하였다[12].

2.3 기존 센서노드 운영체제의 한계

센서노드 운영체제로써 가장 먼저 개발된 TinyOS [4]의 경우 하드웨어 플랫폼이 센서 장착을 위한 표준 인터페이스나 센서 추상화나 센서 디바이스 드라이버 접속을 위한 인터페이스를 제공하지 않는다.

SOS[5]는 메시지 전달, 동적 메모리 할당, 모듈의 자율적인 적재와 제거를 지원하는 공동 커널과 동적 재구성이 특징이다. 이를 통해 새로운 모듈의 업데이트가 필요할 때 무선 네트워크를 통하여 동적으로 변경할 수 있는 구조를 제공한다. 동적 업데이트의 대상은 센서 디바이스 드라이버를 포함하여 응용 프로그램 수준까지 가능하지만 센서 디바이스와 운영체제를 연결할 하드웨어나 소프트웨어 인터페이스를 제공하지는 못한다.

MANTIS[6]는 계층 기반 운영체제로써 하드웨어를 추상화하는 디바이스 드라이버의 특징을 가지고 있다. 그러나 디바이스의 순서가 커널에 고정되어 있어 새로운 디바이스의 추가가 쉽지 않고, 센서와 운영체제를 연결하기 위한 인터페이스가 제공되지 않는다.

Nano-Q+[7] 운영체제는 한국전자통신연구원이 개발한 센서네트워크를 위한 초소형 운영제제로써 C 기반 프로그래밍 환경을 지원하고 다중 스레드 스케줄러 방식 등의 특징을 가지고 있다. 그러나 하드웨어 플랫폼에 센서 인터페이스가 고려되지 않았고 센서를 위한 표준 API도 제공되지 않는다.

Sensos[13]와 Nano-Q+[7]의 스마트 센서 관리시스템에서는 센서 추상화를 제시하고 Linux와 유사한 구조의 센서접근 API를 제공하였다. 하지만 센서 접근을 위한 하드웨어 추상화 지원이 없고 HAL(Hardware Abstraction Layer) 라이브러리가 제공되지 않는다는 문제가 있다.

2.4 IEEE 1451

IEEE 1451[14]은 다양한 트랜스듀서를 네트워크에 연결할 때 트랜스듀서의 투명성을 보장하기 위한 표준이다. 즉, 센서나 구동소자 업체는 연결될 네트워크의 종류나 연결 구조에 상관없이 오직 표준 인터페이스만 제공하면 되고, 네트워크 입장에서는 연결된 트랜스듀서의 종류에 상관없이 공통 인터페이스를 통해 정보를 취득하고 제어할 수 있게 하자는 것이다.

IEEE 1451의 주요 구성 요소는 TEDS(Transducer Electronic Data Sheet), TIM(Transducer Interface Module), NCAP(Network Capable Application Processor) 세 가지이다. TEDS는 트랜스듀서의 특성을 기술하는 데이터 형식으로써 전자적으로 읽기 가능한 메모리 형태로 저장된다. TIM은 TEDS, 트랜스듀서, 데이터 컨버터, 주소 로직으로 구성된다.

IEEE 1451을 센서네트워크 입장에서 보면 TIM은 센서노드 자체가 되고 NCAP는 싱크노드에 해당된다. 따라서 IEEE 1451을 센서노드의 센서 디바이스 추상화 모델로 고려할 수eg 있지만 IEEE 1451 자체가 광범위하여 그대로 센서노드에 적용하기에는 어렵다.

Ⅲ. 센서 투명성 지원 TinyOS 설계

센서 투명성을 지원하는 센서노드 운영체제 아키텍처는 그림 1과 같이 크게 3개 인터페이스로 구성된다. 즉, 운영체제가 응용 프로그램에 제공하는 응용프로그램 인터페이스, HAL을 통해 센서 디바이스와 디바이스 드라이버를 연결하는 HAL 인터페이스, 센서 드라이버의 인터럽트 처리를 위한 센서 인터페이스이다.

본 장에서는 TinyOS 기반으로 설계된 세 가지 인터페이스들에 대하여 기술한다.

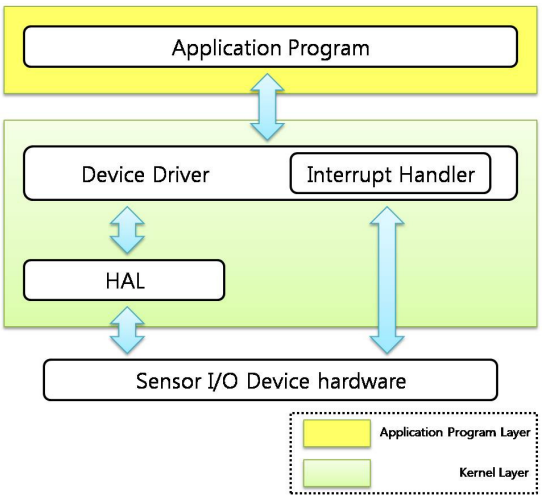


그림 1. 센서 투명성 지원 아키텍처
Figure 1. Architecture for Sensor Transparency

3.1 응용프로그램 인터페이스

응용프로그램 인터페이스는 아래와 같이 open(), read(), write(), close(), ioctl()의 5개 함수로 구성된다. readReady()는 TinyOS의 특성인 이벤트 처리 방식을 그대로 사용하기 위한 인터페이스로써 센서 값 생성이 완료되었을 때 호출되는 이벤트이다.

```
interface sensorAPP {
  command result_t open();
  command result_t close();
  command result_t read();
  command result_t write(uint8_t data);
  command result_t ioctl(uint8_t type, request, data);
  async event result_t readReady(void *data);
}
```

그림 2는 온도 센서의 예를 통해 응용프로그램 인터페이스의 작동 방식을 나타낸 것이다.

open() 함수는 센서 디바이스 사용을 위한 디바이스의 초기화와 read()나 write() 같은 작업에 대해 준비하고, 스위치 포트나 인터페이스 포트의 레지스터 설정과 입출력 설정 등을 결정한다.

close() 함수는 센서 디바이스의 사용을 중단할 때 사용하며 센서 디바이스의 전원을 차단함으로써 저전력 모드로 전환한다.

read() 함수는 준비된 센서 디바이스로부터 센싱된 데이터를 읽을 때 사용하고, write() 함수는 구동기에 데이터를 쓸 때 사용한다. 그림 2와 같이 온도센서의 임계값을 30도로 설정하고 온도가 30도 이상이면 에어컨을 작동시키려 할 때 read() 함수를 이용하여 온도정보를 읽어내고 write() 함수를 이용하여 구동기를 작동시켜 에어컨을 켠다.

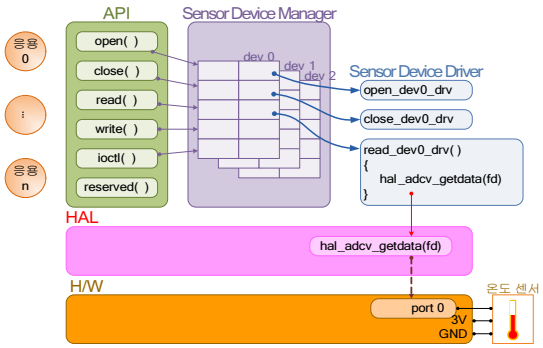


그림 2 HAL 인터페이스와 온도 센서의 예
Figure 2. HAL Interface and Example of Thermometer

ioctl() 함수는 센서 중에 특별한 조작이 필요할 때 쓰이는 함수이다. 한 번의 동작으로 센서값을 읽지 못할 때, 또는 특별한 조작을 취해야 하는 센서의 경우 이 함수를 이용한다.

3.2 HAL 인터페이스

HAL 인터페이스는 응용프로그램 API로부터 센서 디바이스 드라이버를 통해 센서 디바이스의 접근을 지원한다. 플랫폼 공급자는 HAL 인터페이스 드라이버를 제공해야 한다. 플랫폼 인터페이스들은 센서에 공급되는 전원과 센서 인터페이스를 함께 가지고 있어, 제공되는 HAL Interface 드라이버에는 인터페이스 종류와 스위치 핀 번호가 할당되어 있어야 한다.

3.2.1 전원 제어 포트

전원 제어 포트의 제어는 포트 초기화, 전원 ON, 전원 OFF의 세 가지로 동작이 정의된다.

```
interface HALPPORT {
  command result_t init(uint8_t port, uint8_t enable);
  command result_t on(uint8_t port, uint8_t enable);
  command result_t off(uint8_t port, uint8_t enable);
}
```

각각의 포트 정보는, 상위 4비트로 포트 종류를 나타내고 하위 4비트로 연결된 핀의 정보를 나타낸다.

3.2.2 하드웨어 플랫폼 센서 인터페이스

하드웨어 플랫폼 센서 인터페이스는 센서 종류에 따라 인터페이스를 구현하고 접근하게 한다. 인터페이스의 종류는 HALADC, HALINTERRUPT, HALI2C이다. 이 인터페이스는 플랫폼 공급자가 자신의 통합 인터페이스 동작에 관련된 제어를 하기 위함이고 센서 디바이스 드라이버 개발자는 사용하지 않는다.

```
interface HALADC {
  command result_t open();
  command result_t close();
  command result_t read(uint8_t port);
  command result_t write(uint8_t cmd);
  command result_t ioctl(uint8_t data);
  async event result_t dataReady(uint16_t data);
}
interface HALINTERRUPT
interface HALI2C
```

3.3 센서 인터페이스

센서 디바이스 드라이버가 사용하기 위한 인터페이스로써 전원 포트와 센서 인터페이스를 함께 제어한다. 명칭은 HALADCInterface, HALINTRInterface, HALI2CInterface와 같이 HAL$NAME$Interface 형태를 사용한다.

```
interface HALADCInterface
interface HALINTRInterface
interface HALI2CInterface
```

Ⅳ. 시스템 구현

4.1 응용프로그램 API 구현

앞의 3.1절의 sensorAPP 인터페이스 각 함수의 동작 방식은 아래와 같다.

- open(): 전원 포트와 센서 인터페이스를 초기화 한 후 센서에 전원을 공급한다..
- close(): 센서 디바이스의의 전원을 차단한다.
- read(): 연결된 SensorAPP 인터페이스로 센서 값을 요청한다. 완료되면 이벤트를 발생시킨
- write(): 연결된 SensorAPP 인터페이스의 센서 디바이스로 데이터를 보낸다.
- ioctl(): 연결된 SensorAPP 인터페이스의 센서 디바이를 제어한다. 각 전달인자의 의미는 다음과 같다.

type	센서의 어떤 부분을 제어 할지 지정
request	센서의 어떤 동작을 제어 할지 지정
data	지정된 동작의 동작값을 지정

- readReady(): 연결된 SensorAPP 인터페이스에 센싱 값이 있을 때 호출하는 이벤트 함수이다.

4.2 HAL 인터페이스 구현

HAL 인터페이스는 전원 제어 포트와 하드웨어 플랫폼 센서 인터페이스로 구성된다.

하드웨어 플랫폼 센서 인터페이스인 HALADC를 구현한 예를 보면, 인터럽트 서비스 루틴으로써 TOSH_SIGNAL()은 ADC 완료 인터럽트가 발생할 때 dataReady()를 호출하고, 이 때 dataReady()의 전달인자가 16bit 데이터인 이유는 ATmega128의 ADC변환 레지스터가 10비트이기 때문이다.

4.3 센서 인터페이스 구현

센서 인터페이스는 디바이스 드라이버를 위한 표준 인터페이스이다. 센서 디바이스 드라이버 공급자는 센서 인터페이스를 통해 센서 디바이스 드라이버를 작성한다. 따라서 센서를 사용하는 특정 애플리케이션에 관계없이 표준 인터페이스만을 제공함으로써 센서와 애플리케이션을 독립적으로 개발할 수 있다.

센서 디바이스는 ADC 방식과 인터럽트 방식으로 구분할 수 있으며 본 절에서는 각각에 대한 센서 인터페이

스의 구현을 설명한다.

4.3.1 ADC 방식 센서 인터페이스

그림 3은 구현된 Hardware Interface ADC의 와이어링 부분이다. HALPPORTM과 HALADCM 모듈을 HALADC와 HALPPORT 인터페이스를 통해 연결하여 HIADCVM을 만든다. HIADCVC는 HIADCVM의 와이어링을 담당하고 HALADCInterface로 연결한다. 모듈(xxM) 파일은 실제 구동을 정의하는 부분이고 와이어링(xxC) 파일은 컴포넌트들의 연결을 담당하는 부분이다.

그림 3. ADC 하드웨어 인터페이스
Figure 3. Hardware Interface ADC

4.3.2 인터럽트 방식 센서 인터페이스

그림 4는 구현된 인터럽트 하드웨어 인터페이스의 와이어링 부분이다. HALPPORTM과 HALINTR5M 모듈을 HALINTERRUPT와 HALPPORT 인터페이스를 통해 연결하여 HIINTRM을 만든다. HIINTRC는 HIINTRM을 담당하고 HALINTRInterface로 연결한다. 모듈(xxM) 파일은 실제 구동을 정의하는 부분이고 와이어링(xxC) 파일은 컴포넌트들의 연결을 담당하는 부분이다.

그림 4. 인터럽트 하드웨어 인터페이스
Figure 4. Hardware Interface Interrupt

4.4 하드웨어 탑재 및 시험

하드웨어는 센서네트워크 전문업체인 옥타컴[15]에서 제작하였다. 센서노드 플랫폼 인터페이스는 ADC, 인터럽트, I2C, SPI로 구성되고 완성된 센서노드 플랫폼 보드는 그림 5와 같다.

센서는 그림 6과 같이 ADC, 인터럽트, I2C, SPI 센서들을 인터페이스 타입으로 구성하였다.

그림 5. 센서노드 플랫폼
Figure 5. Sensor Node Platform

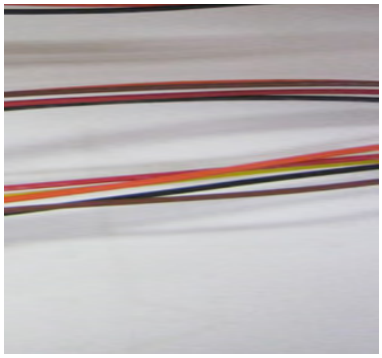


그림 6. 센서 모듈
Figure 6. Sensor Module

V. 결 론

본 논문에서는 범용 운영체제에서 제공하는 디바이스 투명성을 센서노드 운영체제에 적용한 센서 투명성 아키텍처를 제안하고, 센서 투명성을 지원하기 위한 표준 API와 센서 디바이스 추상화를 설계하고 TinyOS 운영체제에서 구현하였다.

센서노드 운영체제에서 센서 투명성이 지원되면 응용 개발자는 운영체제가 제공하는 표준 API를 통해 센서 디바이스에 독립적으로 응용 프로그램을 개발할 수 있고, 센서 데이터는 센서 디바이스 공급자가 개발한 센서 디바이스 드라이버에 의해 처리된다. 또한 센서 디바이스 공급자도 표준화된 하드웨어 인터페이스와 HAL 라이브러리를 기반으로 센서노드 하드웨어 플랫폼에 독립적으로 센서 디바이스 드라이버를 개발하고 공급

할 수 있다. 센서 투명성 지원의 목적은 응용프로그램의 개발을 센서노드 하드웨어 플랫폼이나 센서 디바이스로부터 독립적으로 수행가능하도록 함으로써 개발의 효율을 높이는 것이다. 향 후 연구방향으로써 운용중인 센서네트워크에서 센서노드를 동적으로 추가할 수 있는 Plug&Play 기능에 관한 연구를 수행할 계획이다.

참고문헌

[1] M. Molla, Ahamed, “A survey of middleware for sensor network and challenges,” International Conference on Parallel Processing Workshops, 2006, pp. 223-228.

[2] 은성배, 소선섭, 김병호, “센서투명성을 지원하는 센서노드 운영체제 구조,” 한국정보과학회 학술대회 제35권 제1호(A), 2008, pp. 311-312.

[3] TinyOS Project, http://www.tinyos.net

[4] D. Gay, M. Welsh, P. Levis, E. Brewer, R. Von Behren, D. Culler, “The nesC language: A holistic approach to networked embedded systems,” ACM SIGPLAN, 2006, pp. 1-11.

[5] C. C. Han, R. Kumar, R. Shea, E. Kohler, and M.B. Srivastava, ″A Dynamic Operating System for Sensor Nodes,″ Proc. of MobiSys, 2005, pp.163-176.

[6] H. Abrach, S. Bhatti, J. Carlson, H. Dai, J. Rose, A. Sheth, B. Shucker, J. Deng, and R. Han, ″MANTIS: System Support For MultimodAl NeTworks of In-situ Sensors,″ 2nd ACM Int. Workshop on Wireless Sensor Networks and Applications, 2003, pp.50-59.

[7] S. Park, J. Kim, K. Lee, K. Shin, and D. Kim, ″Embedded Sensor Networked Operating System,″ 9th IEEE Int. Symp. on Object and Component-Oriented Real-Time Distributed Computing, 2006.

[8] M. Becker, A. Beylot, R. Dhaou, A. Gupta, R. Kacimi, M. Marot, ″Experimental study: Link quality and deployment issues in wireless sensor networks,″ LNCS, 2009, pp. 14-25.

[9] 맥스포, http://www.maxfor.co.kr

[10] 휴인스, http://huins.co.kr

[11] 한국유지관리, http://www.kmclab.co.kr

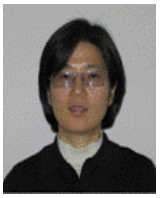
[12] 삼성SDS, http://www.sds.samsung.co.kr

[13] M. Yang, Sun Sup So, Seong Bae Eun, Byungho Kim, Jinchun Kim, “Sensos: A sensor node operating system with a device management scheme for sensor nodes,” Int. Conf. on Information Technology-New Generations, 2007, pp.134-139.

[14] Institute of Electrical and Electronics Engineers, Inc., ″IEEE Standard for Smart Transducer Interface for Sensors and Actuators: NCAP Information Model,″ Mixed- Mobile Comm. WG of the Tech. Committee on Sensor Technology TC-9 of the IEEE Instrum. and Measurement Society, 1999.

[15] 옥타컴, http://www.octacomm.net

저자소개



소선섭(Sun Sup So)

1986년: 이화여대 전산학과 학사
2001년: KAIST 전산학과 석박사
1988년~1995년: 국방과학연구소 연구원
1995년~현재: 공주대학교 컴퓨터공학부 부교수
※관심분야: 소프트웨어 테스팅, 임베디드 소프트웨어, 센서네트워크

은성배(Seong Bae Eun)

1985년: 서울대학교 전산학과 학사
1995년: KAIST 전산학과 석박사
1995년~현재: 한남대학교 정보통신공학과 교수
※관심분야: 실시간시스템, 유비쿼터스 센서네트워크

김병호(Byungho Kim)

1990년: 연세대학교 전산과학과 학사
1997년: KAIST 전산학과 석박사
2007년~현재: 경성대학교 컴퓨터공학과 조교수
※관심분야: 컴퓨터구조, 센서네트워크, 모바일 OS