센서투명성을 지원하는 센서노드 운영체제 구조

[1]은성배○, [2]소선섭, [3]김병호
[1]한남대학교 정보통신공학과, [2]공주대학교 컴퓨터공학부, [3]경성대학교 컴퓨터공학과
[1]sbeun@hnu.kr, [2]triples@longju.ac.kr, [3]bkim@ksu.ac.kr

A Sensor Node Operating System Architecture Providing Sensor Transparency

[1]Seongbae Eun, [2]Sun Sup So, [3]Byeongho Kim
[1]Dept. of Information Communication Eng. Hannam Univ., [2]School of Computer Eng. Kongju Univ.,
[3]Dept. of Computer Eng. Kyungsung Univ.

1. 서 론

센서 노드 개발에는 센서 HW 구현 및 FW 프로그래밍, MCU 프로그래밍 및 ADC 기능 구현, 무선 통신 프로그래밍, 저전력 기능 등이 구현되어야 한다. 현재, USN 응용 개발자는 센서노드에 요구되는 기능들을 직접, 통합, 구현하는데 이 점이 USN 개발을 어렵게 한다. PC나 Linux[1]등에서는 응용 개발자가 PC 플랫폼 HW, 잘 개발된 운영체제, 디바이스 개발자가 제공하는 드라이버를 활용하여 목표 응용만을 개발하면 된다. 운영체제는 표준화된 API를 제공하며 응용 개발자는 이 API에 익숙한 상태이다. 디바이스가 변경되더라도 이 API는 변하지 않는다. 새로운 디바이스를 채택할 때에도 디바이스 드라이버를 디바이스 공급자가 제공하므로 응용이 변경될 필요가 없다. USN 개발에서도 응용 개발자가 잘 개발된 센서노드 운영체제, 그리고 디바이스 드라이버 위에서 응용을 개발한다면 개발 효율을 극대화할 수 있을 것이다.

하지만 기존의 Tiny-OS[2], Nano-Q+[3] 등의 센서노드 운영체제들은 디바이스와 운영체제를 동적으로 연결하는 기능을 제공하지 못한다. 첫째로, 기존 운영체제의 HW 플랫폼들이 센서 연결을 위한 표준화된 인터페이스를 제공하지 못한다. 센서마다 사용전압이 다른데 이를 지원하지도 못한다. 둘째로, 응용 개발자에게 확정된 API를 제공c하지 못한다. USN 응용은 센서의 활용이 매우 다양한데 기존 운영체제들은 센서를 추상화하지 못함으로써 센서마다 별도의 API가 필요하다[4]. 셋째로, 디바이스 개발자에게 디바이스 드라이버를 별도로 개발하고 접속할 수 있는 기능을 제공하지 못한다. 디바이스 개발자가 센서를 공급할 때 이를 위한 디바이스 드라이버를 미리 만들어서 제공하려 해도 운영체제의 센서관련 HAL 라이브러리가 제공되지 못하므로 개발할 수 없다. Sensos[4]와 Nano-Q+의 스마트센서관리시스템[5]에서는 센서 추상화를 제시하고 Linux[1]와 유사한 구조의 센서접근 API를 제공하였다. 하지만 센서 인터페이스를 위한 HW 추상화가 지원되지 못하고 HAL라이브러리가 제공되지 않는다는 문제를 갖는다.

본 논문에서는 표준화된 센서노드 운영체제 기반의 USN 응용개발 방법을 제시한다. 첫째로, 센서장착을 용이하게 하는 표준화된 센서 HW 인터페이스를 기술한다. 둘째로, 센서를 추상화한 센서접근 API를 제공한다. 셋째로, 디바이스 드라이버를 개발할 때 요구되는 HAL 라이브러리를 정의한다. 예제 응용 프로그램을 작성하여 본 논문에서 제안한 센서노드 운영체제 구조가 성공적으로 센서투명성을 지원하는 것을 보인다.

2. 센서노드 운영체제 구조

2.1. 전체적인 구조

센서노드 운영체제에서의 USN 개발을 그림 1에서 도식화하였다. 응용프로그래머는 센서접근 API를 기반으로 응용을 개발한다. 그림에서 API는 open(), close(), read(), write(), ioctl() 등이며 Linux와 비슷하다. 이때 센서 데이터 처리는 센서 제작자가 개발한 디바이스 드라이버가 담당한다. 센서 제작자는 표준화된 센서 HW 인터페이스와 HAL 라이브러리를 기반으로 센서 디바이스 드라이버를 작성한다. 그림에서 드라이버는 HAL 라이브러리 hal_adcv_getdata(fd) 함수를 활용하여 플랫폼의 HW를 접근한다. HW를 직접 접근하지 않고 HAL 라이브러리를 통하여 접근하므로 센서 투명성을 얻을 수 있다.

2.2. 센서 인터페이스 추상화

본 논문에서는 표1 에서처럼 센서를 크게 9개의 인터페이스로 추상화하였다. ADCV는 센서 출력이 전압이며 ADCA는 출력이 전류라는 것을 말한다. ADC도 10비트, 14비트, 24비트 등, 다양하나 이는 인터페이스의 종류를 구분하는 것은 아니고 단지 인터페이스의 성능이 다르다는 것을 의미한다. 따라서 인터페이스 종류에 포함시키지는 않았다. 이와 별도로 전원을 추상화하였는데 V0(전원 없음), V1(3V), V2(5V), V3(9V), V4(12V), V5(24V) 등이다.

추상화명칭	내용	선수	구현
ADCV	ADC일반전압	1선	ADC포트 연결
ADCWV	ADC 미약전압	1선	증폭 후 ADC 포트 연결
ADCA	ADC 일반전류	1선	저항 연결 후 ADC포트 연결
ADCWA	ADC 미약전류	1선	증폭 후 저항 연결 ADC 포트 연결
FREQ	주파수	1선	인터럽트 or 일반 포트에 연결
INTR	인터럽트 방식	1선	인터럽트 포트 연결
I2C	클럭과 데이터	2선	일반 포트에 연결 또는 하드웨어 I2C
SPI	클럭, DI, DO, EN	4선	일반 포트 연결 또는 하드웨어 SPI에 연결
SERI	RS232, RS485 등	3선	하드웨어 직렬 있어야 함.

표 1. 9개의 센서인터페이스 추상화

2.3. 표준화된 API

Linux에서 디바이스를 3종류로 분류한 것과 같이 본 논문에서는 센서를 SENSOR형, EVENT형, ACTUATOR형의 3종류로 분류한다. SENSOR형과 EVENT형 센서는 입력 디바이스이며 ACTUATOR형 센서는 출력 디바이스이다. SENSOR형은 read() 함수 호출시 센서 값을 반환한다는 점에서 EVENT형 센서와 다르다. EVENT형 센서의 경우 사용자가 정의한 handler가 이벤트 발생 시 호출된다. 온도 센서나 조도 센서 등은 SENSOR형 디바이스에 해당되며 초음파 리시버는 동작이 시작된 후 특정 시간이 지나야 초음파를 수신하므로 EVENT형 센서에 속한다. 초음파 발신기는 정해진 시간만큼 초음파를 발신하므로 ACTUATOR형에 속한다.

응용 프로그램이 사용하는 센서 디바이스 API는 모두 5가지이다.

1) int fd = open("device name")
2) void close(fd);
3) int num = read(fd, struct sensor_type *, sizeof(struct sensor_type *));
4) int num = write(fd, struct actuator_type *, sizeof(struct actuator_type *));
5) int err = ioctl(fd, int operation);

2.4. 트랜스듀서 디바이스 메니저

그림 1은 트랜스듀서 디바이스 드라이버 인터페이스 구조를 보여준다. 그림의 자료구조는 운영체제 내에서 응용프로그램의 API호출과 센서 디바이스 드라이버를 연결한다. device_connect() 함수는 응용프로그램에서초기화 때 드라이버를 등록하는 함수이다. 그림에서 3개의 트랜스듀서가 등록되는 것을 볼 수 있다. Transducer Device Table(TDT)은 각 디바이스의 이름과 디바이스 타입, 드라이버 함수 표의 포인터를 저장한다. 응용프로그램이 디바이스를 open()하면 TDT의 인덱스를 리턴하고 함수 호출 시에 이 인덱스를 활용한다.

HW mapping table은 실제 트랜스듀서 디바이스가 하드웨어의 어떤 포트에 연결됐는지를 지정한다. 이 tbale은 운영체제가 초기화 때 지정하며 device_connect() 함수를 호출할 때 논리적 센서와 연결된다.

3. 결론 및 향후 연구 방향

본 논문에서는 센서투명성을 지원하기 위한 센서노드 운영체제 구조를 제안하였다. 이를 위하여 첫째로, 센서장착을 용이하게 하는 표준화된 센서 HW 인터페이스를 정의하였다. 둘째로, 센서를 추상화한 센서접근 API를 제공하였다. 셋째로, 센서 디바이스 드라이버를 작성할 때 활용될 HAL 라이브러리를 정의하였다.

2개의 센서와 1개의 구동기를 갖는 스마트 선풍기의 구현 예를 제시하였다. 그것은 센서 추상화 API를 활용한 응용프로그램, HAL 라이브러리를 활용한 디바이스 드라이버, 그리고 센서 HW 인터페이스를 추상화한 HAL 라이브러리로 구성 된다.각 단계의 프로그램들을 분석하여 본 논문에서 제안한 센서노드 운영체제가 센서 투명성을 성공적으로 지원하는 것을 보였다. 이러한 센서 투명성 기반 OS 구조는 응용프로그래머의 개발 부담을 크게 줄여서 USN 응용 개발을 활성화시킬 것이다.

현재, 센서 투명성 OS 구조를 간단한 응용 정도에 적용했을 뿐이며 다양한 응용에 적용하여 세부 사항을 보완해 나가야 한다. 또한, Nano-Q+ 기반으로 제안된 OS 구조를 구현할 것이다.

참고문헌

[1] A. Rubini, Linux Device Drivers, O'Reilly & Associates, Inc., 1998.

[2] P. Levis, S. Madden, D. Gay, J. Polastre, R. Szewczyk, A. Woo, E. Brewer, and D. Culler, "The emergence of networking abstractions and techniques in tinyos," Proc. of the First USENIX/ACM Symposium on Networked Systems Design and Implementation(NSDI 2004), 2004.

[3] S. Park, J. Kim, K. Lee, K. Shin, and D. Kim, "Embedded Sensor Networked Operating System," Proc. of 9th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing, 2006.

[4] Manseok Yang, Sun Sup So, Steve Eun, Brian Kim, Jinchun Kim, "Sensos: A Sensor Node Operating System with a Device Management Scheme for Sensor Nodes," International Conference on Information Technology (ITNG'07), 2007, pp. 134-139.

[5] Bumsuk Kim, Supsup So, Byeongho Kim, and Sengbae Eun, "A Smart Sensor Device Management System in Nano-Q+," Journal of KIISE: Computing Practices and Letters, Vol. 14, No. 1, Feb. 2008.

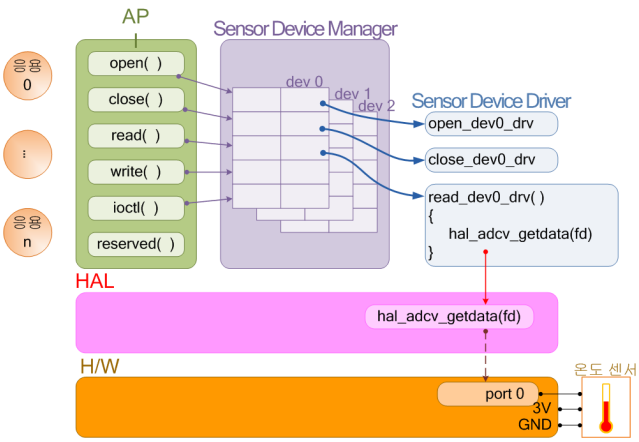

그림 1. 전체 구조

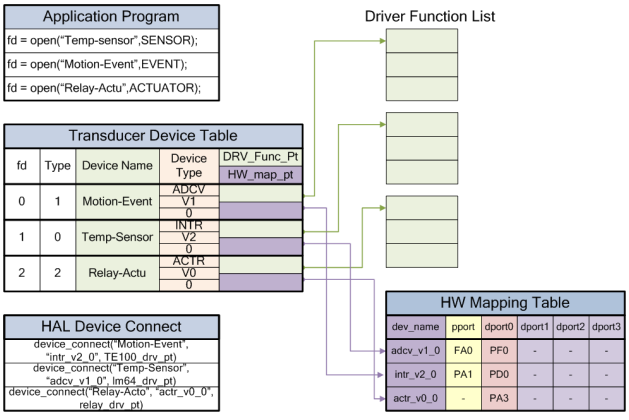

그림 2. 트랜스듀서 관리 매니저의 자료구조