센서 투명성을 지원하는 센서 디바이스 매니저

방상호, 은성배
한남 대학교 정보통신공학과
e-mail : sam626@naver.com, sbeun@hnu.kr

A Sensor Device Manager Supporting Sensor Transparency

Sang-Ho, Bang, Seongbae Eun
Dept. of Information Communication Engineering, Hannam University

요 약

센서노드 운영체제는 응용 프로그래머의 개발 지원 및 체계적인 센서 관리를 위하여 센서 투명성을 지원해야 한다. 하지만 기존 센서 노드 운영체제들은 센서투명성을 지원하지 못한다. 센서 디바이스 드라이버를 응용이 직접 작성해야 하며 다양한 센서를 위한 공통 인터페이스를 제공하지 못한다.

본 논문에서는 센서 투명성을 지원하는 센서 디바이스 매니저를 제안한다. ETRI 에서 개발한 Nano-Q+에서 센서 디바이스 매니저 기능을 구현하기 위하여 센서노드 플랫폼, 응용 API, 디바이스 매니저, HAL 을 설계하고 구현하였다. 또한, 기존 Nano-Q+와 성능을 비교하고 평가하였다. 센서 디바이스 매니저를 구현하여도 처리 속도 및 용량에 대한 성능 저하가 없음을 확인하였다.

키워드: 센서 투명성, 디바이스 드라이버, 디바이스 매니저, 센서 노드 운영체제, Nano-Q+

1. 서론

USN 응용 개발자는 센서노드에 요구되는 기능들을 직접 통합, 구현, 관리한다. 이 점이 USN 개발을 어렵게 한다. Windows 나 Linux 등에서는 응용 개발자가 H/W, 운영체제, 디바이스 드라이버를 활용하여 응용만을 개발한다. USN 응용개발에 이러한 방법을 적용하면 개발 효율을 극대화할 수 있을 것이다.

기존 센서노드 운영체제[1,2,3,4,5,6]는 센서 디바이스와 운영체제간의 동적 연결 및 H/W 플랫폼과 센서 연결을 위한 표준화된 인터페이스를 지원하지 않는다

은[7,8,9,10]은 센서 투명성에 관하여 연구를 진행하고 있다. 2007 년 스마트 센서 디바이스 관리 시스템[10]과 2008 년, 센서투명성을 지원하는 센서노드 운영체제 구조[9]에서 센서에 대한 투명성을 지원하기 위하여 표준화된 센서노드 운영체제 기반의 USN 응용개발방법을 제시하였으며, 제안된 센서노드 운영체제 구조가 성공적으로 센서투명성을 지원하는 것을 보였다.

본 논문에서는 제시된 센서 투명성을 지원하는 센서노드 운영체제구조를 Nano-Q+[6]에 적용한다. 이를 Nano-Q++라 호칭한다. 또한, 기존 Nano-Q+와 제시된 센서노드 운영체제 구조를 갖는 Nano-Q++의 성능을 비교, 분석한다. 응용 프로그램의 처리속도 및 용량에 대한 성능평가를 통해 구동 및 성능에 문제가 없음을 보인다.

2. 배경

2.1 기존 센서노드 운영체제

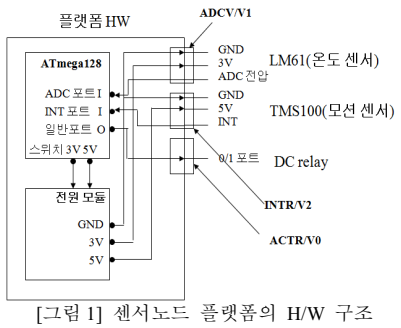
기존 센서노드 운영체제인 Tiny-OS[1], SOS[2], MANTIS[3], PEEROS[4], ANTS[5], Nano-Q+[6] 등은 센서 디바이스와 운영체제 간의 동적 연결 및 H/W 플랫폼과 센서 연결을 위한 표준화된 인터페이스를 지원하지 않는다. 표준화된 API 및 리프로그래밍을 제공하지 않는다. 그래서, 센서의 통합적인 관리기법을 지원하지 못한다.

2.2 Nano-Q+에서 스마트 센서 디바이스 관리 시스템

Nano-Q+에서 스마트 센서 디바이스 관리 시스템[10]에서는 센서 I/O 서브시스템을 제안하여 응용개발자에게 통일된 API 를 제공하며 디바이스 드라이버의 장탈착이 용이한 센서 디바이스 매니저를 구현하였다. 센서 추상화를 제시하고 Linux 와 유사한 구조의 센서접근 API 를 제공하였다. 하지만 센서 인터페이스를 위한 H/W 추상화가 지원되지 못하고 HAL 라이브러리가 제공되지 않는다는 문제를 갖는다.

2.3 센서투명성을 지원하는 센서노드 운영체제 구조

센서투명성을 지원하는 센서노드 운영체제 구조[9]에서와 같이 응용개발자는 센서접근 API 를 기반으로 개발한다. 이때 센서 데이터 처리는 센서 제작자가 개발한 디바이스 드라이버가 담당한다. 센서 제작자는 표준화된 센서 H/W 인터페이스와 HAL 라이브러리를 기반으로 센서 디바이스 드라이버를 작성한다. 드라이버는 HAL 라이브러리 함수를 통하여 플랫폼의 H/W 에 접근한다. HAL 라이브러리를 통하여 접근하므로 센서 투명성을 얻을 수 있다.



[그림 1] 센서노드 플랫폼의 H/W 구조

3. 설계 및 구현

3.1 플랫폼 설계

센서노드 플랫폼은 그림 1 와 같은 Hardware 구조를 가진다. 센서노드 플랫폼은 특정 센서의 인터페이스를 제공하지 않고, 표준화된 인터페이스를 지원한다. 표준화된 인터페이스를 통해 다양한 센서 사용이 가능하다.

3.2 응용 API 설계 및 구현

응용개발자가 디바이스에 상관없이 개발이 가능하도록 설계되었다. 리눅스 드라이버의 추상화 방법을 응용하였다. 하지만, 센서 네트워크에서는 센서노드 자체가 소규모 자원 제약적인 시스템이고 단순한 센싱 기능만을 요구하는 경우가 많아서 Open(), Close(), Read(), Write(), Ioctl() 5 가지로 정의 하였다.

3.3 디바이스 매니저 설계 및 구현

Transducer Device Manager 의 자료구조[9]를 바탕으로 설계 및 구현하였다.

3.3.1 Driver Function List

Driver Function List 에는 driver_Open(), driver_Close(), driver_Read(). driver_Write(), driver_Ioctl()의 함수들을 등록한다. 함수 등록은 함수 포인터로 저장한다.

[표 1] Drive Function List 의 함수 별 기능

Function	Function Explanation
driver_Open()	디바이스 초기화
driver_Close()	센서에 들어가는 공급 전원 또한 차단
driver_Read()	센서의 값을 읽음
driver_Write()	출력 상태 값을 전달
driver_Ioctl()	Open()과 관계없이 센서 전원을 차단

센서 디바이스 드라이버는 센서 공급자가 제공한다. 센서 공급자는 센서의 이름을 지정할 때, 다음과 같은 형식으로 지정한다.

```
DriverFuncList 센서이름_drv_func_list;
```

3.3.2 Transducer Device Table

Transducer Device Table 은 센서 이름으로 하드웨어 정보와 센서 디바이스 드라이버의 정보를 연결한다.

```
Struct transducerDeviceTable
{
    INT8 device_name[15];
    UINT8 mapping_table_index;
    DriverFuncList *driver_function_list;
    UINT8 isValid;
}trans_dev_table[DEV_MAX];
```

device_name 은 사용자가 device_connect() 호출 시 지정한 센서의 이름이다. mapping_table_index 는 연결된 디바이스 인터페이스의 H/W 정보를 가리킨다. driver_function_list 는 센서 디바이스 드라이버를 가리킨다. isValid 변수 값에 따라 위의 정보의 사용가능 여부를 나타낸다.

Transducer Device Table 은 device_connect()와 device_disconnect()를 사용하여 내용을 변경을 한다.

```
UINT8 device_connect(INT8 *name, UINT8 dev_name,
                     DriverFuncList *dev_func_list)
UINT8 device_disconnect(INT8 *name)
```

.

3.4 HAL 설계 및 구현

플랫폼 공급자는 다음과 같이 HAL 드라이버를 제공해야 한다. H/W 에서 제공하는 인터페이스들의 H/W Mapping Table 을 제공한다. 제안된 플랫폼의 인터페이스는 센서의 공급 전원과 인터페이스를 묶어 사용한다. Mapping Table 은 device_connect()로 Transducer Device Table 의 구성요소로 등록이 된다.

```
//mapping table 정보
typedef struct infoPowPort{
    UINT8 port;
    UINT8 enable;
}InfoPowPort;

typedef struct infoDataPort{
    UINT8 type;
    UINT8 port[9];
}InfoDataPort;

typedef struct mappingTable{
    InfoPowPort powPort;
    InfoDataPort dataPort;
}MappingTable;
```

4. 성능평가

센서 투명성을 지원하는 센서 디바이스 매니저를 적용 Nano Q++의 성능평가는 기존 Nano-Q+와 응용프로그램의 처리 속도와 메모리 사용량에 대해 비교하였다.

4.1 처리 속도 비교

처리 속도에 대한 비교는 Instruction 수를 이용하여 수행시간을 비교해 보았다.

[표 2] Nano-Q+와 Nano-Q++의 수행시간 비교

		Number of Instruction	Run Time
Led & Gas Sensor action	Nano-Q+	1723	4.3 ms
	Nano-Q++	6127	15.31 ms

표 2 와 같이 Nano-Q++가 Nano-Q+에 비해 Instruction 수가 증가하였다. 수행시간에 대한 Overhead 가 발생하였지만, 실제 수행시간 11.01ms 의 차이로 처리속도에 거의 영향을 주지 않았다.

4.2 메모리 사용량 비교

Atmega128 의 4 Kbyte RAM 에 대한 고려가 필요하다. Nano-Q++로 변경되면서 증가된 RAM 사용량은 표 4 와 같다.

[표 3] 추가된 데이터 형의 사용량

Data Type	Size
Mapping Table	12 byte
UNIFORM_IO_DRV	10 byte
WRITE_ARGV	3 byte
DriverFuncList	10 byte
trans_dev_table	20 byte

예를 들어, 5 개의 센서를 사용할 경우, 아래 계산식과 같이 총 275 byte 의 RAM 사용량이 증가한다.

$$12 \times 5 + (10 + 3 + 10 + 20) \times 5 = 60 + 215 = 275$$

4 Kbyte RAM 에서 275 byte 의 Overhead 는 문제가 되지 않는다.

5. 결론 및 향후 연구 방향

본 논문에서는 센서 투명성을 지원하는 디바이스 드라이버 매니저에 대해 기술하였다. 센서 투명성을 지원하기 위한 구조를 Nano-Q+에 적용하였다. 응용 프로그램 개발 시 Overhead 가 발생하였다. 하지만, 이러한 Overhead 는 실제 응용 프로그램 구동에 미치는 영향이 적다. 센서 투명성을 지원하는데 있어 성능 저하 없음을 확인하였다.

센서 투명성을 지원하기 위하여, 응용 API, 드라이브 매니저, HAL 을 간략화하여 설계하였다. 다양한 센서에 대한 통일된 접근방식으로 센서 디바이스에 접근할 수 있음을 확인하였다. 이러한 센서 투명성을 지원하는 OS 구조가 응용프로그래머의 개발 부담을 줄였다.

현재, 센서 투명성을 지원하는 H/W 가 존재 하지 않았다. 또한, 다양한 센서에 대하여 적용하지 않았다. 향후 연구방향은 플랫폼 H/W 제작 및 다양한 센서에 대한 응용에 대하여 연구하는 것이다.

참고문헌

[1] 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.

[2] Chic-Chieh Han, Ram Kumar, Roy Shea, Eddie Kohler and Mani Sricastava, "A dynamic operating system for sensor nodes," Proc. of MobiSys, 2005, pp.163-176.

[3] 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," Proc. of 2nd ACM International Workshop on Wireless Sensor Networks and Applications, 2003, pp.50~59.

[4] J. Mulder, S. Dulman, L. van Hoesel, and P. Havinga. PEEROS --- System Software for Wireless Sensor Networks. Preprint, August 2003

[5] KIM Daeyoung; TOMAS SANCHEZ LOPEZ; YOO Seongeun; SUNG Jongwoo; KIM Jaeeon, KIM Youngsoo; DOH Yoonmee, "ANTS : An evolvable network of tiny sensors", Lecture notes in computer science (Lect. notes comput. sci.) ISSN 0302-9743

[6] SooMah Pyeong / Nano OS for Defense Sensor Networks / The Korean Institute of Information Scientists and Engineers, September 2006, pp.70~74.

[7] 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.

[8] Manseok Yang, Sun Sup So, Brian Kim, Jinchun Kim, Seongbae Eun, “Sensos: A Sensor Node Operating System with a Device Management Scheme for Sensor Nodes”, IEEE Computer Society Proceedings of the Fourth International Conference on Information Technology: New Generations 2007

[9] Seongbae Eun, Sun Sup So, Byeongho Kim / A Sensor Node Operating System Architecture Providing Sensor Transparency / Korea Computer Congress, 2008 Vol35, No 1

[10] Bumsuk Kim, Sunsup So, Byeongho Kim, Seongbae Eun / A Smart Sensor Device Management System in Nano-Q+ / The Korean Institute of Information Scientists and Engineers 14-1