Interrupt 기반의 반응 속도가 향상된 센서 데이터 전송 방법 구현

이민구, 강정훈, 임호정, 윤명현
전자부품연구원

The Implementation of Interrupt-based method to transmit in USN

Min goo Lee, Jeong Hoon Kang, Ho Jung Lim, Myung Hyun Yoon
Korea Electronics Technology Institute

Abstract - 본 논문은 무선 센서 네트워크 플랫폼을 이용한 어플리케이션 개발 과정에서 비규칙적인 센싱 데이터의 획득 후 빠른 시간 내에 이를 전송하기 위한 효율적인 방법에 대한 고찰에서 시작되었다.
개발과정에서 진동 센서를 채택하여 이원화된 메시지 전송 프로세스를 구현하였다. 즉, 진동센서의 불규칙 센싱 완료 인터럽트 발생에 따른 메시지 전송 프로세스와 타이머에 의한 주기적인 진동 센싱 및 메시지 전송 프로세스로 동작할 수 있는 방법을 구현하였다.
따라서 본 논문에서는 인터럽트 기반의 센싱 데이터 전송방법과 주기적인 센싱 데이터 전송방법에 대한 비교 분석을 통해 반응속도 측면에서 인터럽트 기반의 센싱 데이터 전송 방법이 더욱 효율적인 데이터 전송 프로세스임을 보여주고자 한다.

Key Words -인터럽트 기반, 진동 센서, 반응속도, 신뢰도

1. 서 론

무선 센서 네트워크 기술이 1998년 U.C 버클리 컴퓨터 사이언스의 컬러(Culler) 교수팀에 의해 등장한 이래, 센서 노드 플랫폼, 저전력 MCU, 저전력 RF 트랜시버, 라우팅 알고리즘, MAC 프로토콜, 저전력 프로토콜, 무선 소프트웨어 다운로드, 어플리케이션 등 무선 센서 네트워크 관련된 여러 분야에서의 획기적인 발전을 이룩하였다.

본 논문에서는 이 같은 무선 센서 네트워크 기술 분야의 발전 과정 가운데 하나인, 센서 플랫폼의 다양한 센서 인터페이스 기술의 개발에 대해 초점을 맞추어 다루고자 한다.

센서 네트워크 플랫폼에 신규 센서를 부착하기 위해서는 두 가지 측면에서 고려와 구현이 동시에 이루어져야 한다. 즉, 하드웨어 측면과 소프트웨어 측면에서의 고려와 구현이다.

본 논문 관련하여 개발하고자 하는 어플리케이션은 센서 플랫폼에서 센서 값을 획득한 후 매우 짧은 시간 내에 이를 전송해야 하는 요구사항을 가지고 있다. 즉, 진동(Vibration) 센서가 부착된 센서 플랫폼에서 진동 센서 값을 획득할 때마다 무선으로 데이터를 전송하여 원격에 위치한 서버에서 이를 인지하는 어플리케이션을 개발하고자 한다.

이때 중요하게 고려할 사항이 진동 센서 값은 주기적으로 발생하기 보다는 임의의 시간에 불규칙하게 발생한다는 점이고, 이 같은 특성을 보완하기 위해 주기적인 센싱값 획득보다는 센싱값의 획득 이벤트 발생 시점에 전송하는 프로세스를 선호하게 되었다.

이를 위해 본 논문에서는 먼저 '주기적인(Periodic) 센싱값 전송 어플리케이션'과 '인터럽트 기반(Interrupt-based)의 센싱값 전송 어플리케이션'을 각각 구현하여, 반응 속도와 신뢰도 두 가지 측면에서의 성능을 비교 분석하였다.

2. 본 론

2.1 OS와 프로그래밍 언어

본 논문에서 구현하고자 하는 진동 센서 어플리케이션은 U.C 버클리에서 배포한 무선 센서 네트워크를 위한 전용 OS인 TinyOS와 TinyOS를 위해 특별히 개발되어진 프로그래밍 언어인 NesC로 구현하였다.

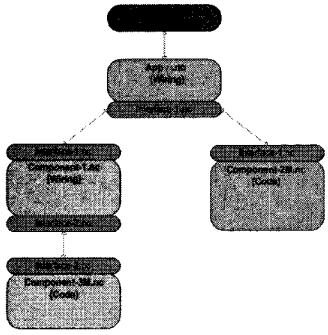
TinyOS는 센서 네트워크와 같은 네트워크 임베디드 시스템(Network Embedded System)들을 위해 특별히 만들어진 OS 이다. 이는 이벤트 기반의 어플리케이션, 소형의 코어 OS(400 바이트 정도의 코드), 작은 데이터 메모리를 갖는 초소형 용량의 OS를 만들기 위해 고안되었으며, 이벤트 기반의 단순 스케줄러 기능만을 제공하며, 현재 TinyOS-2.x 버전을 제공하고 있다.

TinyOS는 FIFO 구조의 테스크 큐(Queue)에 처리해야 할 테스크(Task)의 유무를 확인하고 처리해 준다. 이 과정이 끝나면, Sleep 상태에 빠지게 되고, 이벤트가 발생하게 되면 이벤트 핸들러가 인터럽트 서비스 루틴(ISR)에 따라 인터럽트 벡터 내에 있는 상응하는 동작을 취하게 된다. 이 과정을 완료하면 다시 Sleep 상태로 돌아가게 된다.

본 논문과 관련한 어플리케이션의 구현을 위해 사용한 프로그래밍 랭귀지는 NesC이다. NesC는 TinyOS의 특성들을 지원하기 위해 U.C 버클리에서 만들어 배포한 무상의(free) 프로그래밍 언어이다.

NesC로 구현된 어플리케이션들은 컴포넌트와 양방향의 인터페이스들로 구성되고, 테스크와 이벤트들에 기반을 둔 동시성 모델을 제공한다는 점이 특징이다.

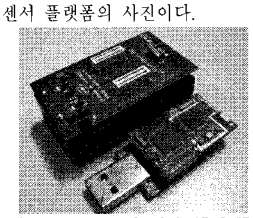
<그림 1>처럼 NesC는 초소형, 초저전력, 컴포넌트 기반의 구조라는 특성을 만족시키기 위해 컴포넌트라는 개념에 기반을 두어 TinyOS의 이벤트 기반의 동시성을 지원하고, 공유된 데이터의 동시 액세스가 가능하도록 지원하고 있다.



<그림 1> NesC 구조

2.2 RF 플랫폼과 센서 플랫폼

본 논문에서 구현하고자 하는 진동 센서 관련 어플리케이션의 동작 검증을 위해 RF 플랫폼은 Kmote를 사용하였고, 센서 플랫폼으로는 (주)유니온사에서 개발한 진동 센서 플랫폼을 사용하였으며, <그림 2>는 Kmote와 (주)유니온사의 진동 센서 플랫폼의 사진이다.

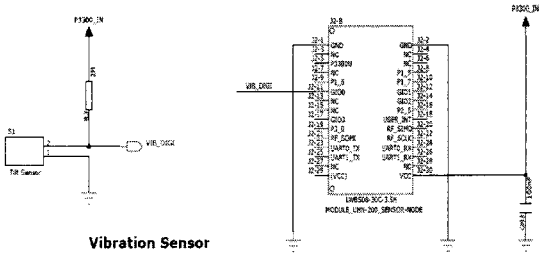


<그림 2> Kmote와 진동 센서 플랫폼

센서 네트워크 플랫폼으로 가장 대표적인 것은 U.C 버클리의 MOTE 시리즈이다. 이 같은 MOTE는 1998년 처음 발표된 이후 다양한 형태(DOT, MICA, MICA2, TelosA, TelosB, MicaZ 등)로 발전되어 오고 있다.

본 논문에서 사용한 RF 플랫폼은 <그림 2>처럼 전자부품연구원에서 개발한 Kmote를 사용하였는데, Kmote는 2.4GHz 무선 RF 대역을 사용하고, 무선 통신 칩셋으로는 TI사의 CC2420 모듈을 적용하고 있다. 무선 통신을 위한 안테나로는 Inverted F-Type 세라믹 안테나와 다이폴 안테나를 선택하여 사용할 수 있는데, 본 논문에서는 세라믹 안테나를 채택하였다.

Kmote의 MCU는 TI사의 MSP430 F1611을 사용하였으며, 프로그램을 다운로딩 할 수 있는 48KB의 플래시 메모리와 10KB의 RAM을 사용한다. 전원은 AAA 사이즈의 1.5V 배터리 2개를 직렬로 연결하여 3V 전원을 공급해주고 있다. 특히, 프로그램 인터페이스로 USB를 이용한다는 점이 매우 편리하다.



<그림 3> 진동 센서와 MCU의 연결

본 논문에서 사용한 센서 플랫폼은 <그림 3>처럼 진동 센서(Tilt_Sensor)를 부착하였으며, <그림 4>와 같이 MCU의 GIO0 포트(2번 Port 1번 Pin)를 통해 센싱값을 인식하도록 설계되었다.

```
60 // GIO pins
61 TOSH_ASSIGN_PIN(GIO0, 2, 0);
62 TOSH_ASSIGN_PIN(GIO1, 2, 1);
63 TOSH_ASSIGN_PIN(GIO2, 2, 3);
64 TOSH_ASSIGN_PIN(GIO3, 2, 6);
65
```

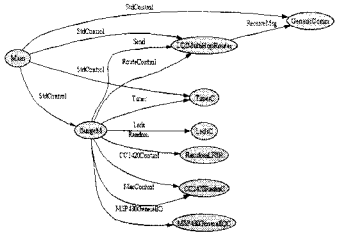
<그림 4>

2.3 어플리케이션 구현

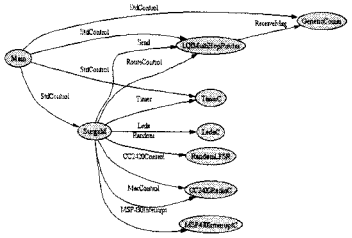
<그림 5-1>과 <그림 5-2>는 '주기적인(Periodic) 센싱값 전송 어플리케이션'과 '인터럽트 기반(Interrupt-based)의 센싱값 전송 어플리케이션'에 대한 각각의 소프트웨어 연결도이다. 전체적인 구현은 비슷하게 이루어졌으며, 가장 큰 차이점은 센싱값의 획득 및 변환을 위해 사용된 컴포넌트(Component)와 인터페이스(Interface)가 다르게 쓰였다는 점이다.

먼저, 주기적인 센싱값 전송 어플리케이션을 구현하기 위해 사용된 컴포넌트는 <그림 5-1>처럼 MSP430GeneralIOC 컴포넌트와 MSP430GeneralIO 인터페이스가 사용되었다.

이에 비해 인터럽트 기반의 센싱값 전송 어플리케이션의 구현은 <그림 5-2>처럼 MSP430InterruptC 컴포넌트와 MSP430Interrupt 인터페이스를 사용하여 구현하였다.



<그림 5-1> 주기적 센싱값 전송 어플리케이션



<그림 5-2> 인터럽트 기반의 센싱값 전송 어플리케이션

2.4 실험 환경 및 결과

본 논문에서 '주기적인(Periodic) 센싱값 전송 어플리케이션'과 '인터럽트 기반(Interrupt-based)의 센싱값 전송 어플리케이션'의 성능을 비교 분석하기 위해 반응 속도와 신뢰도 두 가지 측면에서의 성능을 비교하였다.

2.4.1 실험 환경

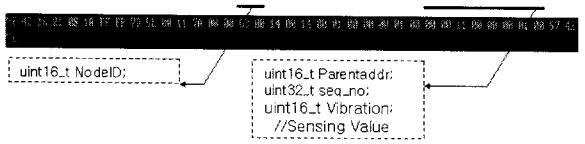
이를 테스트하기 위한 환경으로는 센서 플랫폼이 결합되어 있는 센서 노드(A)와 무선으로 센싱 데이터를 수신하기 위한 센서 노드(B), 그리고 센서 노드(B)와 USB로 연결되어 센싱 데이터를 분석하기 위한 노트북(VAIO VGN-T27LP/L)으로 구성하였다.

센서 노드(A)에 프로그래밍 된 어플리케이션을 '주기적인(Periodic) 센싱값 전송 어플리케이션'과 '인터럽트 기반(Interrupt-based)의 센싱값 전송 어플리케이션' 각각을 순차적으로 구분하여 퓨징하였으며, 센서 노드(B)에는 무선으로 메시지를 수신할 수 있는 TOSBase 어플리케이션을 퓨징하여 테스트하였다.

센서 노드(A)와 센서 노드(B)간의 무선 통신 환경으로는, GroupID(0x7a), RF Power(31), Channel(21)을 사용하였다.

센서 노드(B)와 결합되어 사용된 노트북에 설치된 메시지 분석 Tool은 TinyOS에서 배포한 'ListenRaw' Java Tool을 시그윈(Cygwin) 상에서 사용하였다.

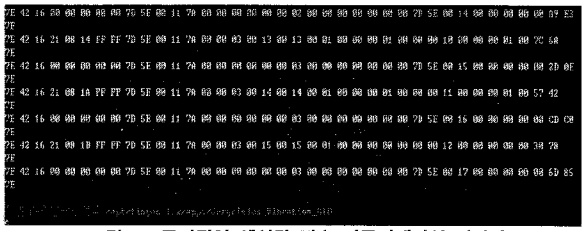
진동 센서 값의 무선 전송을 통해 수신된 메시지의 구조는 <그림 6>과 같다.



<그림 6> 수신된 메시지 구조

2.4.2 실험 결과

<그림 7>은 주기적인 센싱값 전송 어플리케이션에 대한 메시지 분석을 ListenRaw Tool 상에서 살펴본 결과이다.



<그림 7> 주기적인 센싱값 전송 어플리케이션 결과값

<그림 8>은 인터럽트 기반의 센싱값 전송 어플리케이션에 대한 메시지 분석을 ListenRaw Tool 상에서 살펴본 결과이다.

<그림 8> 인터럽트 기반의 센싱값 전송 어플리케이션 결과값

3. 결 론

<그림 7>과 <그림 8>에서의 '주기적인(Periodic) 센싱값 전송 어플리케이션'과 '인터럽트 기반(Interrupt-based)의 센싱값 전송 어플리케이션' 실험 결과값처럼, 수신되어진 메시지의 Vibration 센싱값의 패턴이 크게 다름을 알 수 있다. 즉, <그림 7>의 경우 센서 노드(A)에서 전송되어진 센싱값이 주기적으로 '1'의 값이 전송됨을 볼 수 있다. 이에 비해 <그림 8>의 경우는 센서 노드(A)에서 전송되어진 센싱값이 '1'이 연속적으로 나오는 구간이 발생함을 알 수 있다.

<그림 8>의 경우, 센서 노드(A)의 진동 센서에서 센싱값을 획득한 즉시 메시지를 전송하고, 진동 센싱값이 센싱 되는 경우에 모두 메시지를 전송하기 때문에 <그림 7>의 경우와 같이 1초마다 센싱 하여 그때의 센싱값을 전송하는 패턴과 다르게 나타난 것이다. 즉, 반응속도 측면에서 '인터럽트 기반의 센싱값 전송 어플리케이션'이 더욱 뛰어난 성능을 보이고 있음을 알 수 있다.

이 같은 실험을 반복해서 횟수를 증가시켜보면, 이 같은 실험결과는 '주기적인 센싱값 전송 어플리케이션'과 '인터럽트 기반의 센싱값 전송 어플리케이션'의 신뢰도 측면에서의 비교가 가능하다.

<표 1>은 실험 테스트의 횟수를 증가하며 '주기적인 센싱값 전송 어플리케이션'과 '인터럽트 기반의 센싱값 전송 어플리케이션'의 신뢰도 측면을 살펴보기 위해 진동 센싱 인터럽트 발생 횟수 대비 메시지 수신 성공 횟수를 살펴본 결과이다.

<표 1> 진동 인터럽트 발생에 따른 메시지 수신 횟수

	10회	50회	100회	300회
Periodic	6	32	65	173
Interrupt	10	48	95	287

즉, 진동 인터럽트 발생 횟수를 10회, 50회, 100회, 300회 반복해서 발생하여 주기적인 어플리케이션과 인터럽트 기반의 어플리케이션 각각의 경우에 메시지 수신 횟수를 살펴본 결과 인터럽트 기반의 어플리케이션인 경우에는 인터럽트 발생 횟수에 거의 수렴하는 메시지 수신을 보이는 반면, 주기적인 어플리케이션의 경우에는 50~60%정도의 메시지만 수신됨을 확인할 수 있었다.

따라서 신뢰도 측면에서도 반응속도 측면에서 살펴본바와 같이 '인터럽트 기반의 센싱값 전송 어플리케이션'이 더욱 우수함을 보여주고 있다.

[참 고 문 헌]

[1] David Gay, Philip Levis, David Culler, Eric Brewer. NesC 1.1 Language Reference Manual. May 2003.

[2] J.Hill, R.Szewezyk, A.Woo, D.E.Culler, and K.S.J.Pister. System Architecture Directions for Networked Sensors. In Architectural Support for Programming Languages and Operating Systems, pages 93-104, 2000.

[3] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges for reliable multihop routing in sensor networks. In SenSys '03, Los Angeles, California, Nov. 2003.

[4] http://docs.tinyos.net/index.php/Main_Page

[5] http://www.tinyos.net/tinyos-1.x/doc/tutorial/index.html