모바일 환경에서의 센서 프레임워크에 대한 제안

최은영*, 배두환**
*삼성전자 소프트웨어연구소
**한국과학기술원 전산학과
e-mail : choiey@samsung.com

A Proposal of Sensor Framework Architecture in the Mobile Environment

* Eun-Young Choi, ** Doo-Hwan Bae
* Software Laboratories, Samsung Electronics Co., Ltd.
** Div. of Computer Science, EECS, Korea Advanced Institute of Science and Technology

요 약

최근 유비쿼터스가 활성화되면서 센서에 대한 관심이 높아지게 되었고, 모바일 환경에서의 센서 프레임워크에 대한 정의가 명확히 이루어지지 않은 상황에서 지금까지 센서 데이터를 얻어오는 주체는 어플리케이션에 있었다. 본 논문에서는 모바일 기기에서 여러 가지 센서를 관리하고 센서 데이터를 가져오기 위한 센서 프레임워크를 레이어드 아키텍처로 제안하고 제안된 센서 프레임워크에 대한 효용성에 대해 논의한다.

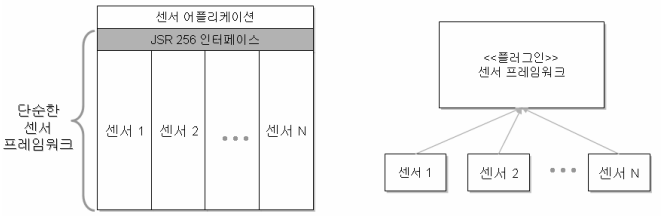
1. 서론

편재형 컴퓨팅에서 컨텍스트는 다가오는 정보화 사회에서 강력한 영향을 줄 수 있는 새로운 서비스 개발을 위한 열쇠이다[1]. 컨텍스트 정보는 센서를 통해서도 얻어질 수 있는데 센서 기술의 발달로 센서는 더 작고 강력해짐으로 모바일 기기에 탑재되는 것이 가능해졌다. 모바일 기기에 내장된 센서는 사용자의 상황 정보를 실시간으로 직접 가져올 수 있는데 이렇게 수집된 컨텍스트 정보는 더 지능적인 서비스를 제공하게 할 수 있을 뿐 아니라 사용자의 패턴을 분석하여 미래의 행동을 예측할 수도 있다.

특히 모바일 기기에 다양한 센서가 탑재됨에 따라 이를 효율적으로 관리하기 위한 센서 프레임워크가 필요하게 되었는데 아직까지 모바일 센서 프레임워크에 대한 정의가 뚜렷이 되어 있지 않아 이를 정의하고 구현하기 위한 작업이 필요하게 되었다. 본 논문에서는 모바일 환경에서의 센서 프레임워크를 레이어드 아키텍처로 제안하고 구현한 결과에 대해 논의한다.

2. 단순한 센서 프레임워크

기존에 정의된 표준으로 모바일 센서 어플리케이션을 위한 인터페이스를 정의해 놓은 JSR256[3]이 있지만 이는 전체 센서 프레임워크를 커버하지 않고 어플리케이션 레벨에서만 기술되어 있어서 센서 프레임워크를 정의한다고 보기는 어렵다. 그러나 전체적으로 센서의 기능을 충실하게 정의하고 있다는 면에서 JSR256 을 어플리케이션 인터페이스로 사용하여 간단한 센서 프레임워크로 나타내어 보면 그림 1 과 같다.



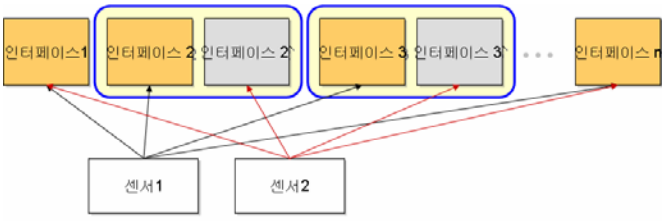
(그림 1) JSR256 을 사용한 간단한 센서 프레임워크 아키텍처

이 센서 프레임워크는 플러그인 구조로 필요한 센서를 컴포넌트로 설치 및 제거 할 수 있으며 JSR256 인터페이스 아래에 위치 센서나 온도 센서 등 다양한 센서 모듈들을 가지고 있는 구조이다. 각 센서 모듈은 JSR256 인터페이스를 통해 어플리케이션과 통신한다.

3. 단순한 센서 프레임워크에 대한 분석

겉으로 보기에 그림 1 의 센서 프레임워크는 큰 문제를 가지고 있는 것 같지 않지만 여러 센서 모듈들을 개발하면서 몇몇 논점들이 제시되기 시작했다. 그 중 큰 부분을 차지한 것은 센서 모듈들간의 코드 중복 문제이다. 센서 모듈 구현 시 각각의 센서는 JSR256 의 모든 인터페이스를 구현할 필요가 있는데 그림 2 의 센서 1 과 센서 2 가 JRS256 인터페이스 1 과 n 처럼 기존에 구현되어 있던 인터페이스를 그대로 사용하는 경우도 있었지만 인터페이스 2 와 3 와 같이 따로 구현해서 사용하는 경우도 있다. 그러나 따로 구현되어야 하는 인터페이스는 각 센서 모듈마다 특성

이 다르게 때문에 정해져 있지 않다.



(그림 2) 공통으로 사용되는 인터페이스와 새로 구현되는 인터페이스

각각 따로 구현되어 사용되는 인터페이스에 대한 코드를 보면 중복되는 부분이 많았는데, 예를 들면 센서마다 따로 구현된 SensorConnection 인터페이스에서 getData()를 구현하는 부분에서 각 센서에는 (1) 센서로 센서 데이터를 요청 (2) 가져온 센서 데이터를 IData 로 변환 (3) 아웃풋 리스트로 데이터를 패칭해주는 공통적인 작업이 있음에도 단순한 센서 프레임워크 구조에서는 각 센서의 특성에 맞추어 이 인터페이스를 따로 구현해야만 한다.
또한 이벤트 관련 부분도 모든 센서는 공통적으로 동기와 비동기 이벤트를 처리함에도 불구하고 이를 위한 별도의 모듈이나 인터페이스가 존재하지 않아 센서마다 각각 구현되었다.

4. 간단한 센서 프레임워크의 분석 결과

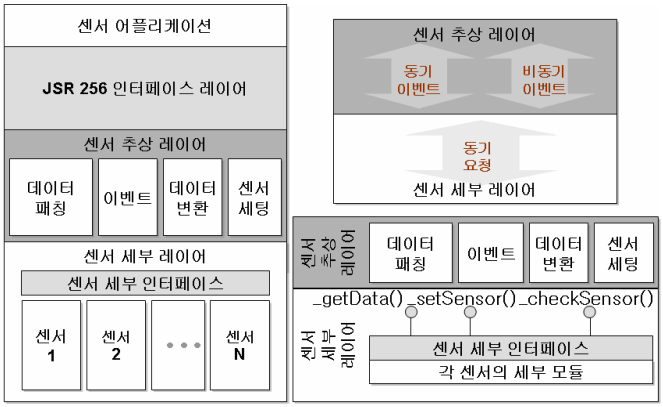
앞에서 분석된 문제들로 인해 코드 중복이 일어나게 되었는데 이를 총 합산해보면 전체 센서 모듈 코드의 20%에 달한다. 이는 센서 모듈들 간에 공통된 부분들을 추출해서 처리하면 더 효율적으로 센서 개발을 할 수 있을 뿐 아니라 중복된 코드가 없는 센서 프레임워크 개발이 가능하다는 것을 의미한다. 각기 구현된 인터페이스들 사이에서 공통적으로 뽑을 수 있는 부분을 정리하면 표 1 과 같다.

<표 1> 센서 모듈들간의 공통 부분

인터페이스 및 기능	세부 항목	설명
이벤트	동기 이벤트	어플리케이션이 센싱 된 데이터를 요청할 때 즉시 결과 값을 처리하는 경우.
	비동기 이벤트	어플리케이션이 센싱 된 데이터를 요청할 때 setDataListener() 또는 Condition 에 대한 센싱 데이터를 요청하는 경우.
Sensor Connect-ion 인터페이스	데이터 변환	각 센서는 고유의 데이터 특성을 가지고 있음. 센서로부터 가져온 데이터를 센서 프레임워크의 포맷에 맞도록 변환.
	데이터 패칭	센서 프레임워크 포맷에 맞도록 Data 로 변환한 후 getData()의 리턴 값으로 데이터를 패칭해주는 작업.
Sensor-Info 인터페이스	센서 세팅	센서들은 각각의 고유한 특성을 갖는다. 채널 수를 비롯하여 각 채널의 이름, 정확도, 단위, 데이터 타입 등에 대한 세팅이 다르다. 이러한 부분에 대한 세팅하는 부분이 필요.

5. 개선된 센서 프레임워크에 대한 제안

각 센서는 공통적으로 처리해야 하는 부분도 있지만 각각의 고유한 특성이 고려되어야 하는데 이러한 점을 감안하여 개선된 센서 프레임워크를 그려보면 그림 3 과 같다.



(그림 3) 개선된 센서 프레임워크 아키텍처

이 센서 프레임워크는 JSR256 인터페이스를 그대로 유지하면서 센서 모듈들간의 공통된 부분을 지원하는 레이어와 센서 고유의 특성을 반영할 수 있는 레이어로 나뉘어져 있다.

5.1. JSR256 인터페이스 레이어

이 레이어는 앞의 단순한 센서 프레임워크와 같이 JSR256 인터페이스를 따르는 레이어이기 때문에 기존에 개발되었던 센서 어플리케이션은 변경될 필요가 없다.

5.2. 센서 추상 레이어

이 레이어는 센서 모듈들간의 공통적인 기능들을 포함하고 있다. 특히 이벤트 처리 부분에 있어서 그림 4 와 같이 하여 이전에는 각 센서 모듈이 개별적으로 구현해야 했던 부분을 공통으로 처리해 주었다.
동기 이벤트의 경우 센서 추상 레이어는 즉시 센서 세부 레이어로부터 버퍼 크기 만큼 센서 데이터를 가져오고, 비동기 이벤트의 경우에는 센서 추상 레이어가 순환문을 돌면서 DataListener 나 ConditionListener 를 체크하여 각 리스너가 만족될 때까지 센서 데이터를 가져온다.
모든 이벤트를 센서 추상 레이어에서 처리해 줌으로 각 센서 모듈 개발자는 이벤트를 구현하지 않아도 되고 추상 레이어가 요청할 때마다 데이터를 주기만 하는 구조로 변경되었다.

```
if synchronous event then
    while as buffer size
        data <- _getData(buffer size)
if asynchronous event then
    while until asynchronous event is over
        data <- _getData(buffer size)
        check DataListener
        check ConditionListener
        start timer as sensor rate
```

(그림 4) 동기 및 비동기 이벤트 처리를 위한 프로세스

5.3. 센서 세부 레이어

이 레이어는 센서의 세부적인 특성을 반영하기 위한 것으로 센서 추상 레이어와는 센서 세부 인터페이스를 통해 데이터를 주고 받는다. 각 센서 개발자들은 그림 3 과 같이 미리 정의된 세부 센서 인터페이스에 대해 구현하고 센서의 특정한 부분들, 예를 들어, 네트워크를 사용하는 센서의 경우 네트워크 연결을 위한 모듈과 같은 특정한 부분들에 대한 부분을 구현하면 된다.

센서 추상 레이어가 센서의 공통적인 부분을 처리하여 전체적으로 센서 프레임워크를 간결하게 만들었다면, 이는 이전에 어떤 인터페이스를 새롭게 구현해야 하는지 정의되지 않아 센서 개발자들에게 혼란을 주었던 구조를 개선한 것으로 센서 세부 레이어는 센서의 특정한 부분에 대해 정확히 구분하고 각 센서가 구현해야 하는 인터페이스를 명확히 정의했다는 점에서 의미가 있다.

6. 개선된 센서 프레임워크의 효용성

표 2 는 센서 프레임워크의 개선 전후에 대한 상황을 각 항목별로 기술해 놓은 것이다. 개선 후 기존 센서 어플리케이션은 그대로 활용할 수 있고 전체적으로 센서 프레임워크가 더 간결해지고 센서 개발이 더 용이하게 되었음을 확인할 수 있다.

또한 수치적으로 개선 후 센서 개발 생산성을 비교해 보면 개선 후 센서 개발이 12%까지 향상된 것을 볼 수 있었다.

7. 결론

본 논문에서는 JSR256 을 사용한 간단한 센서 프레임워크를 살펴보고 모바일 환경에서의 센서 프레임워크를 어플리케이션 개발 시각이 아니라 센서 개발자의 시각에서 전체 프레임워크를 분석하고 공통 부분을 추출하여 개선된 레이어 개념의 센서 프레임워크를 제안하였다. 개선된 센서 프레임워크는 센서 개발의 생산성을 향상시켰고 각 센서의 기능을 분명히 구분하여 정의함으로 중복되는 부분 없이 더 간결한 구조를 가졌다. 이러한 센서 프레임워크는 편재형 컴퓨팅에서 센서를 통해 컨텍스트 정보를 수집하는 데 효율적으로 사용될 수 있다.

<표 2> 센서 프레임워크 개선 전후 비교

비교항목	개선 전	개선 후
인터페이스 및 클래스	개선 전후를 비교해볼 때 개선 후 기존에 있던 인터페이스나 클래스는 그대로 사용하고 추가로 추상 레이어에 있는 인터페이스만 추가	
구현해야 하는 인터페이스	JSR256 의 일부 인터페이스 중 개발하려는 센서와 맞아떨어지지 않는 부분을 따로 구현	세부 레이어 인터페이스와 각 센서를 위한 특정 부분들만 구현
이벤트	동기와 비동기 이벤트 둘 다른 고려	추상 레이어에서 요청이 올 때 센서 데이터를 전달.
시간 및 노력	개선 후 개발 시간과 노력이 감소.	
개발자들이 느끼는 센서 프레임워크에 대한 접근성 정도	개선 후 이벤트에 대한 고려가 필요 없고, 세부 레이어에서 정의한 인터페이스만 구현하면 되기 때문에 접근성이 용이해 짐.	

참고문헌

[1] Joëlle Coutaz, James L. Crowley, Simon Dobson, and David Garlan, CONTEXT is KEY, Communication of ACM, 2005, 53

[2] Matthias Baldauf and Schahram Dustdar, A Survey on Context-Aware Systems Int. J. Ad Hoc and Ubiquitous Computing, Vol. 2, No. 4, 2007

[3] Mobile Sensor API, JSR256 ver. 1.0, JSR256 Expert Group (http://jcp.org/en/jsr/detail?id=256)