공 학 석 사 학 위 논 문

센서장치의 태그화를 위한 센서 리더 에뮬레이터

2009 년 2 월

부 경 대 학 교 대 학 원

컴 퓨 터 공 학 과

이 화 춘

공 학 석 사 학 위 논 문

센서장치의 태그화를 위한 센서 리더 에뮬레이터

지도교수 송 하 주

이 논문을 공학석사 학위논문으로 제출함.

2009 년 2 월

부 경 대 학 교 대 학 원

컴 퓨 터 공 학 과

이 화 춘

이화춘의 공학석사 학위논문을 인준함.

2009 년 2 월 23 일

주 심 공학박사 서 경 룡 (인)

위 원 이학박사 신 상 욱 (인)

위 원 공학박사 송 하 주 (인)

목 차

그 림 목 차

표 목 차

센서 장치의 센서 태그를 위한
센서 리더 에뮬레이터

이 화 춘

부 경 대 학 교 대 학 원 컴 퓨 터 공 학 과

요 약

최근의 RFID 응용은 태그를 이용한 사물의 식별 이외에 빛, 온도, 습도와 같은 센서 장치를 함께 활용하는 것이 일반적인 추세이다. 이 때, RFID 태그는 리더를 통해 이벤트처리 방식으로 사용되는 반면, 센서 장치는 함수호출 방식으로 사용된다. 따라서 RFID응용개발자의 입장에서는 두 가지 데이터 처리 방식을 혼용하여 프로그램을 개발해야 한다. 이에 본 논문에서는 센서 장치를 센서 태그 형식으로 사용할 수 있게 하는 센서 리더 에뮬레이터를 제안한다. 센서리더 에뮬레이터를 사용하면 단일한 방식으로 RFID 태그와 센서 장치를 접근할 수 있어 RFID 응용프로그램을 효과적으로 개발할 수 있다. 또한 태그의 위치가 고정된 응용일 경우에는 고가의 센서태그와 센서리더 대신 사용될 수 있으므로 저비용으로 높은 효과를 제공할 수 있다.

A Sensor Reader Emulator for Converting Sensor Device into Sensor Tags

Hwa Chun Lee

Department of Computer Engineering, The Graduate School,
Pukyong National University

Abstract

It is normal that the new RFID system is utilized by tag which is not only recognizable things but also light, temperature and humidity. This is, it is used to method of event but sensor device is used to method of function call. Therefore, RFID experts should develop both programs for data processing. This is following a dissertation which propose to Sensor Reader Emulator by method of sensor tag. It is possible develop RFID system Program that Sensor Reader Emulator allows to unitary system both RFID tag and sensor device. And it is high-performance and cutting down expenses for using them where tag positions of fixed system use to Sensor Reader Emulator instead of a high-priced sensor tag and sensor reader.

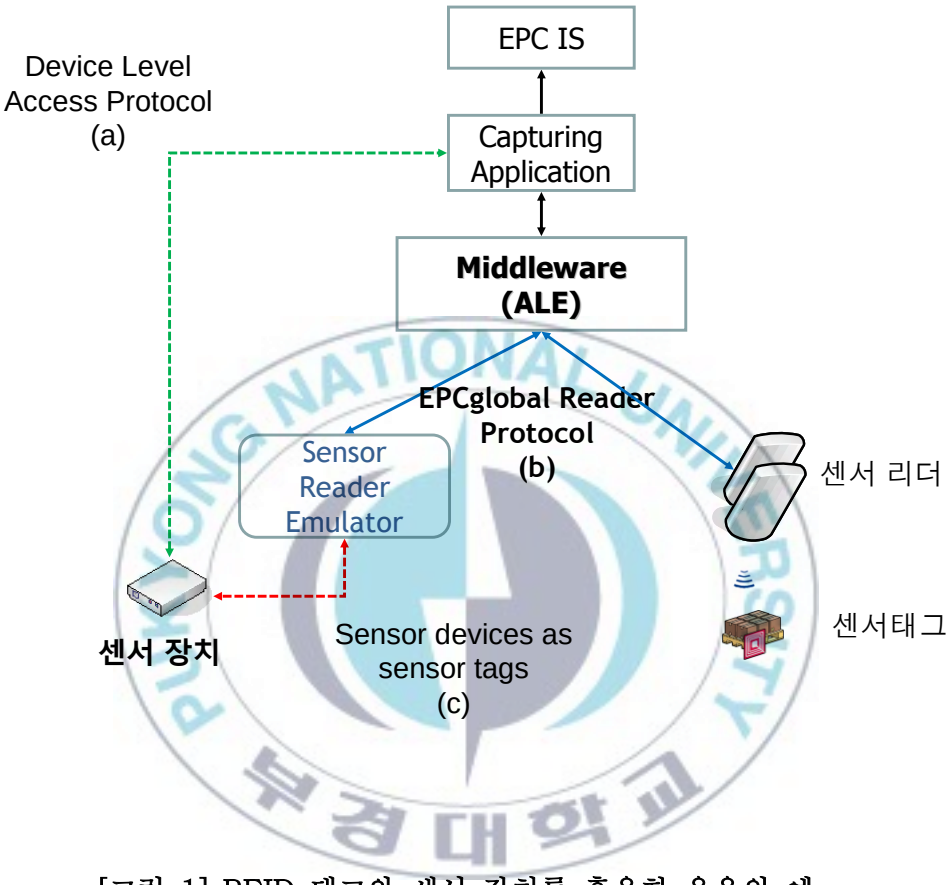
1. 서 론

1.1 연구의 필요성

RFID 기술의 응용범위가 초기에는 무선식별이라는 응용분야에 집중되었으나 최근에는 온도, 습도, 압력 등과 같은 센서 기능과 결합하여 다양한 응용분야에 적용이 확대되고 있다. RFID 미들웨어 또한 식별기능 이외에 센서 태그에 대한 처리 기능을 포함해야 할 필요가 있다. 기존의 식별전용 리더는 TagID에 대한 판독이 주요한 기능이다. 센서의 경우에는 기존 RFID 리더와는 판이하게 다른 호스트 인터페이스를 가진 장치이다. 따라서 RFID 미들웨어가 RFID 태그, 센서태그, 센서장치를 어댑터 형식으로 일관되게 사용할 수 있는 확장된 어댑터 프레임웍의 개발이 필요하다.

그림 1은 이러한 복합적인 RFID 응용에서 태그로부터 리더, 그리고 각종 RFID 미들웨어에 이르기까지의 시스템 구조를 나타낸 것이다. 여기서 RFID 응용 시스템의 핵심이라 할 수 있는 RFID 미들웨어는 RFID 리더를 통해 보내는 각종 센싱 정보를 이벤트(Event) 형식으로 입력 받아 내부적으로 처리한 후 리더 프로토콜(Reader Protocol)[1]로 상위 레벨(Capturing Application)에 전달한다(그림 1의 b 경로). 반면 일반 센서 장치는 Capturing Application[2]의 레벨에서 디바이스 드라이버에 대한 함수호출 형식으로 접근하게 된다(그림 1의 a). 따라서 개발자의

입장에서는 전달 경로와 전달 방식이 다른 센싱 정보를 동시에 처리해야 하는 부담이 따른다.



[그림 1] RFID 태그와 센서 장치를 혼용한 응용의 예

본 논문에서는 이러한 비효율을 없애기 위해서 그림 1의 (c)에서 보인 것처럼 센서 장치를 센서태그처럼 접근할 수 있게 하는 센서리더 에뮬레이터(Sensor Reader Emulator, SRE)를 제안한다.

본 논문에서는 본 절에서 제시하고 있는 부하를 없애기 위해서 그림1의 (c)와 같은 구조로 센서장치를 마치 리더는 아니지만 하나의 리더를 다

루는 것과 같이 구성 하면 보다 효율적으로 사용할 수 있다. 또한 (a)의 구조는 센서장치를 접근하는 것이 Device level로 접근하지만 본 논문에서 제시하는 (c)와 같은 센서 리더 에뮬레이터를 거쳐서 접근하게 되면 센서장치가 마치 센서태그처럼 사용 할 수 있게 한다. 이로 인해 센서장치에서 얻은 정보는 센서리더에서 얻은 정보와 동일한 정보를 상위 계층으로 전달 할 수 있어 단일한 데이터 처리 방식에 단일한 메시지 포맷을 이용하여 센서와 리더를 모두 접근 할 수 있다.

현재 센서리더에 사용되고 있는 센서태그는 고가의 가격으로 인해 연구나 개발이 힘들다. 그래서 이를 대신할 매개체 역할이 필요하다. 고가의 센서태그 대신에 기존에 쓰이고 있는 센서(온도계, 습도계, 압력계 등)들을 활용한다면 센서태그 비용을 줄일 수 있다. 이를 활용하기 위해서는 기존의 센서에서 측정한 데이터와 가상 센서태그 값과 맵핑 시켜주는 센서 리더 에뮬레이터가 필요하다. 따라서 본 논문에서는 이런 가상 센서 리더를 지원하기 위한 설계와 구현 방안을 제안하고 설계한다.

1.2 연구의 목적 및 방법

본 연구는 EPCglobal[3]에서 제시하고 있는 표준 리더 인터페이스와 프로토콜에 대해서 조사한다. 또한 센서장치를 RFID 센서태그와 동일한 방식으로 접근할 수 있도록 지원되어야 한다. 그러기 위해서는 Reader Protocol을 지원해야 하고 센서장치의 공통 인터페이스를 정의해야 한다.

이렇게 공통으로 정의된 인터페이스를 이용해서 각 센서장치 별로 어댑터를 간단히 구현하면 EPCglobal Reader Protocol 형식으로 사용할 수 있게 된다. 그리고 각 센서 장치마다 관리할 수 있는 모니터링 기능이 추가되면 설정파일 설정이나 EPC 코드부여 등 동작 여부의 판단이 쉽게 이루어질 수 있다.

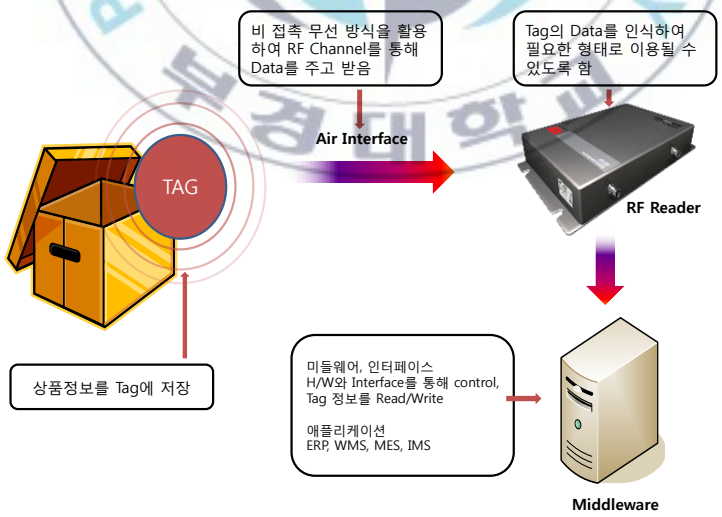
1.3 논문의 구성

본 논문은 다음과 같은 내용으로 구성된다. 2장에서는 관련 연구를 조사하고, 3장에서는 본 논문의 주제인 Sensor Reader Emulator의 아키텍처와 중요 구성 클래스에 대해 알아본다. 4장에서는 구현 및 적용결과에 대해서 살펴본다. 마지막으로 5장에서는 결론과 향후 연구방향에 대해 기술한다.

2. 관련연구

2.1 RFID

RFID 기술[4]은 바코드 시스템과 마그네틱 카드 시스템이 우리 생활에 밀접하게 이용되고 있으나 생산 방식의 변화, 소비자 의식의 변화, 문화 및 기술의 진보, 바코드와 마그네틱 카드의 단점 해소 요구에 의해 개발된 시스템[5]이다. 즉, 무선으로 사람, 물건, 동물 등을 인식, 추적, 식별 할 수 있는 기술이다.



[그림 2] RFID 개념도

RFID 시스템 그림 2와 같이 태그, 리더, 그리고 태그로부터 읽어 들인 데이터[6]를 처리할 수 있는 데이터 처리 시스템으로 구성된다.

RFID 태그는 리더로부터 전원을 공급받거나, 데이터를 수신 또는 송신하기 위한 안테나, 태그의 ID 및 사용자가 임의로 읽고 저장할 수 있는 메모리를 포함하고 있는 장비를 말하며, 각 종 사물에 부착하여 리더기를 통하여 인식되는 과정[7]이 필요하다.

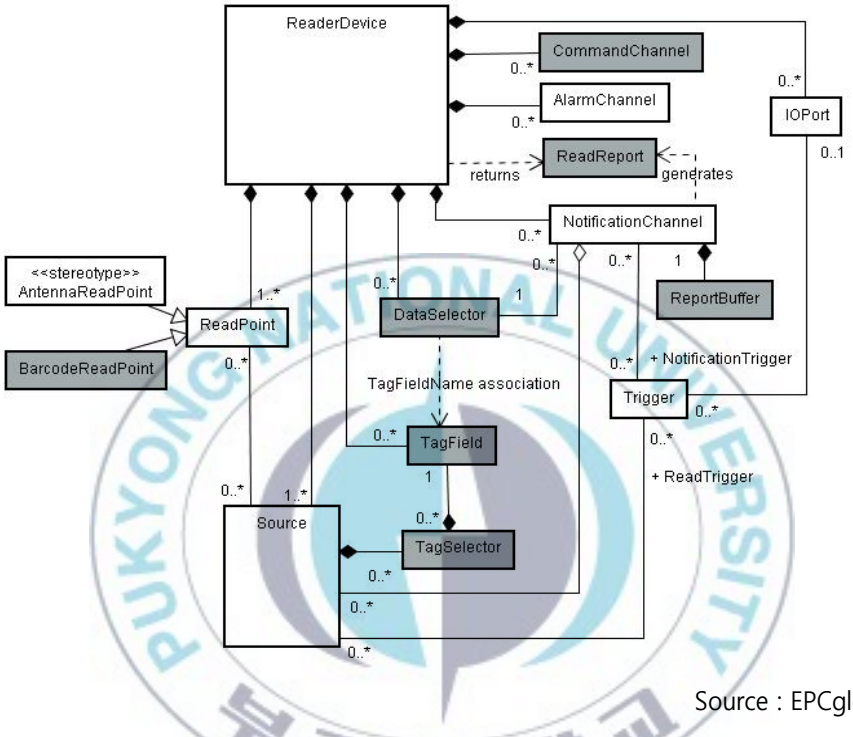
RFID 리더는 소프트웨어 애플리케이션이 비 접촉 식의 RFID 태그로부터 데이터를 읽거나 쓰기 위해 개발된 장치이다. RFID 리더는 HF 인터페이스를 통하여 능동형 태그로 명령을 전송하며, 태그로부터 응답 데이터를 수신하여 디지털 데이터로 복호화하는 기능을 한다.

Middleware는 일반적으로 컴퓨터 시스템에서 미들웨어[8]란 내부시스템과 외부시스템의 중간에 위치하여 대량의 데이터를 효율적으로 처리하고, 서로 다른 애플리케이션에서 데이터 교환이 가능 하도록 해주는 소프트웨어이다. 네트워크상에 연결된 다양한 하드웨어, 응용시스템, 통신망 환경, 운영체제 등의 특성에 관계없이 서로 통신이 가능하도록 소프트웨어적인 기능을 제공한다. 즉, 다양한 이기종 환경에서 응용 프로그램과 운영환경 간에 원활한 통신을 가능하게 해주는 역할을 담당한다.

2.2 Reader Protocol & Reader Management

다음 그림3은 RP와 RM의 Object UML이다. 그림에서 검게 표시된 부

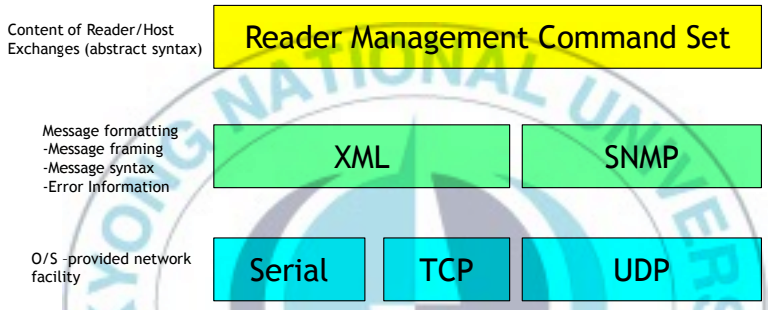
분은 RP부에서만 필요한 부분이고 흰 바탕은 RP와 RM 공통으로 필요한 Object들이다. 이 UML에서 나타내고 있는것과 같이 ReaderDevice Object에서 모든 Object들을 관리하고 있는 것을 알 수 있다.



[그림 3] RP & RM Object UML

EPCglobal Reader Protocol 1.1의 구조는 크게 네 가지 하위 시스템으로 구성되며 여러 개의 안테나로 구성된 Source에서 읽은 태그ID 중 관심 있는 태그만을 필터링(Filtering)하고, 태그 이벤트를 생성 후 상위 미들웨어가 요구하는 조전에 맞는 태그 정보를 그림 4와 같이 MTB (Massaging / Transport Bind)를 통하여 전송한다. 이러한 구조를 통해

기본적인 리더의 기능을 정의 하고 있으며, EPC Class 태그에 대한 기능들을 포함 하고 있다. EPC Class 0, 1을 위한 기본적인 READ, WRITE, KILL, LOCK 명령과 최근 화제가 되고 있는 C1G2 (Class 1 Generation 2)를 위한 Session 기능과 강화된 보안 관련 기능을 포함한다.



[그림 4] Message Transport Binding

✓ Reader Layer

리더 계층은 리더와 호스트 사이에서 보내어지는 메시지에 들어갈 수 있는 내용 및 메시지 형식을 정의한다. 이 계층은 OSI의 표현 계층 및 응용 계층으로 구성된다. 이 규약에 따르면 이 계층에서 여러 개의 MTB를 사용할 수 있지만 특정 인스턴스에서는 한 번에 한 개의 MTB만 사용할 수 있다. 또한 MTB와 무관

하게 리더에서는 한 번에 한 호스트하고만 대화할 수 있다.

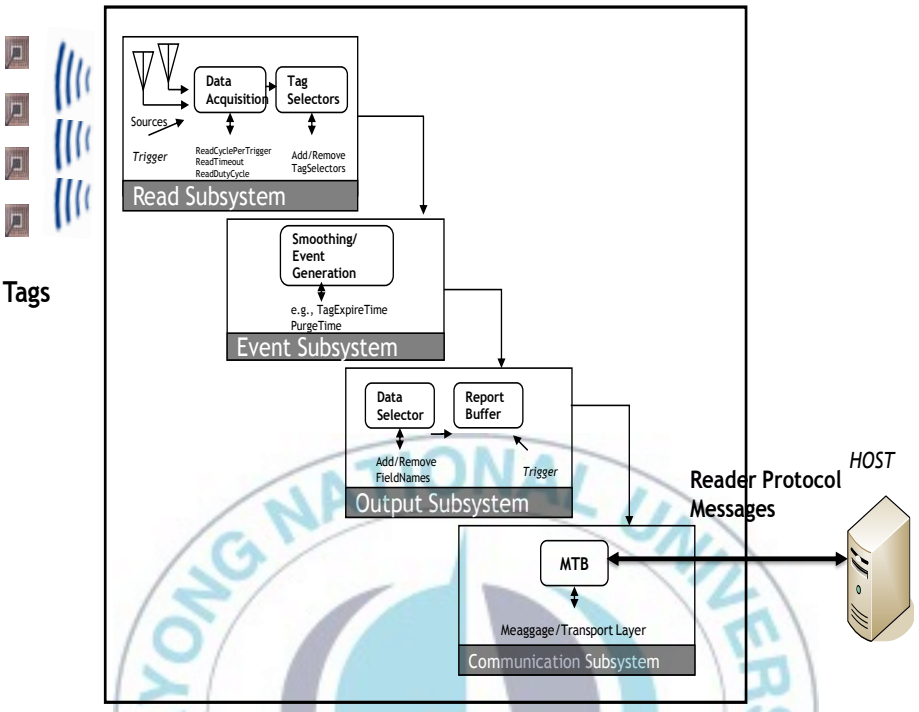
✓ Massage Layer

이 계층은 전송 계층 위에 있으며 연결과 보안을 관리하고 호스트 명령 및 리더 응답, 그리고 통지를 패키징해서 전송 계층과 주고받을 수 있도록 하는 일을 맡고 있다. 암호화나 인증, 세션 관리 등은 모두 이 계층에서 이루어 진다. 이 계층은 호스트와 리더 사이의 대화가 시작되고 끝나는 방법을 기술하며 메시지가 왔다 갔다 하는 프레임이나 엔빌로프의 모양을 정의한다. 논리적으로 OSI의 세션 계층과 동일하다.

✓ Transport Layer

최하위 계층으로, OS에서 제공하는 서비스 또는 네트워킹을 지원하기 위해 필요한 하드웨어를 기술한다. 즉, OSI의 물리, 데이터 링크 및 네트워크 계층에 대응된다고 볼 수 있다. 전송 계층은 카테고리 5 케이블을 통한 이더넷 TCP, 트위스티드 페어 선을 이용한 RS485또는 무선 블루투스 구현을 비롯한 리더와 호스트 사이의 물리적 연결 및 네트워킹 연결 계층이다.

2.2.1 Read Layer Scenario



[그림 5] Reading Pipeline

✓ Read Subsystem

소스 단계에서 데이터 수집 단계로, 데이터 수집 단계에서 읽기 필터링 단계로 데이터가 넘겨진다. 이 절차가 끝나면 읽기 필터링 단계에서 Event Subsystem으로 데이터를 전달한다. Read Subsystem은 상태가 없는 시스템이기 때문에 태그를 읽을 때마다 그 태그를 처음 읽는 것처럼 움직인다.

✓ Event Subsystem

단순히 몇 번 읽히지 않은 태그와 아예 사라진 태그를 구분할 수 있도록 데이터에 대해 평탄화 필터를 적용하는 일을 담당한다. Read Subsystem에는 상태가 없기 때문에 태그를 읽을 때마다 무조건 보고한다. 평탄화 및 이벤트 생성 단계에서는 시간이 지나가도 상태를 저장하기 때문에 전에 읽었던 것과 지금 읽은 것을 비교할 수 있고, 결과적으로 새 태그가 등장했다거나 전에 있었던 것과 지금 읽은 것을 비교할 수 있고, 결과적으로 새 태그가 등장했다거나 전에 있었던 태그가 없어졌다거나 하는 유의미한 이벤트만을 전달하는 기능을 한다.

✓ Output Subsystem

리더에서 보고할 데이터를 결정하고, 데이터를 버퍼링하고, 호스트에서 설정한 트리거에 의해, 또는 호스트의 직접 요청에 의해 그 보고서를 전송하는 일을 담당한다.

✓ Communication Subsystem

원하는 네트워크 프로토콜과 메시지 포맷의 조합(Message Transport Binding) 사용하여 호스트 측과 태그 정보를 교환한다. MTB를 구현하여 Output Subsystem 의 Report Buffer에 있는 데이터를 호스트로 넘겨주는 역할을 한다. MTB는 다양한 네트워크 환경에서 유연한 메시지 전달이 가능하도록 한다.

2.2.2 Reader Protocol 1.1

다음 구성은 Reader Protocol 1.1의 중요 Object들이다.

- ✓ ReaderDevice Object

  리더 디바이스 오브젝트는 모든 리더 오브젝트의 기본 컨테이너가 되는 객체로써 하나의 리더를 대표한다 리더 디바이스는 적어도 하나의 지정된 명령 채널(Command Channel)을 가지고 있으며 추가적으로 리더 디바이스의 속성들과 네트워크에 필요한 객체들을 가지고 있다. 이 객체들은 Sources, DataSelectors, NotificationChannels, Triggers, TagSelectors 들로써 리더 디바이스가 생성될 때 내부적으로 같이 생성된다.

- ✓ Source Object

  기본적으로 태그 정보를 읽어 들여서 그를 바탕으로 만들어진 이벤트를 상위계층에 전달하는 것이 목적이다.

- ✓ DataSelector Object

  데이터 셀렉터 오브젝트는 노티피케이션 채널(Notification Channel)에 의해 유지될 수도 있고 태그를 읽는 명령의 파라미터로써 제공될 수도 있다. 만약 데이터 셀렉터에서 지정할 필드가 유효한 값을 가지고 있지 않을 경우에는 해당 필드는 무시되

어서 노티피케이션 정보등에 포함되지 않는다.

✓ Notification Channel Object

노티피케이션 체널은 기본적으로 하나의 호스트에게 노티피케이션 리포트를 제공하는 통로이다. 노티피케이션 채널은 일련의 소스들(Sources)의 목록을 유지하고 있으며 이들 소스들로부터 받는 이벤트를 모아서 호스트에 리포트 한다. 이 때 데이터 셀렉터(Data Selector)로부터 리포트 하는데 어떤 필드들이 필요한지 정보를 받아서 해당 항목만 리포트 하게 된다. 따라서 노티피케이션 체널은 데이터 셀렉터와 명시적으로 연결을 맺고 있어야 하므로 현재 연결된 데이터 셀렉터를 얻을 수 있는 메소드가 지원되어야 한다. 노티피케이션 채널에서 호스트 쪽으로 노티피케이션 리포트를 제공하는 시점은 트리거(Trigger)에 의해서 결정된다. 그러나 호스트 쪽에서 임의의 시점에 데이터가 필요한 경우에는 노티피케이션 채널에서 제공하는 readQueuedDate()를 이용할 수 있다.

✓ TagSelector Object

태그 셀렉터는 소스(Source) 오브젝트가 멤버로써 가질 수도 있고 필요하지 않은 경우 하나도 가지지 않을 수도 있는 오브젝트이다. 기본적으로 소스는 태그 셀렉터를 하나도 가지지 않고

있다. 태그 셀렉터는 리더로부터 읽혀 들어오는 태그들의 정보를 1차적으로 걸려내는 필터링 역할을 담당하며 단순한 비트단위의 패턴매칭 방법을 사용한다.

✓ Trigger Object

트리거는 source에 적용될 때는 ReaderTrigger로 NotificationChannel에 적용될 때는 NotificationTrigger로 동작한다.

2.2.3 Reader Management 1.0.1

다음 구성은 Reader Management 1.0.1[9]의 중요 구성 Object들이다.

✓ AlarmChannel Object

리더에서 호스트로 비 동기화된 메시지를 보내는 역할을 한다. 발생하는 메시지는 오직 호스트에게만 보내고 Reader Protocol의 NotificationChannel을 다루는 역할을 한다. 모든 Alarm 메시지는 AlarmChannel을 통해서 전송이 된다.

✓ ReadPoint Object

자료를 취득하기 가능한 물리적인 존재를 가진다. 현재 표준으로 정의되어있는 Reader Management에서는 Barcode와 RF

Reader Antenna를 지원 하는데 현재 버전에서는 Antenna만 지원한다.

✓ AntennaReadPoint Object

Antenna를 사용하여 물리적으로 실행되는 ReadPoint이다. 각 AntennaReadPoint는 Reader의 상태를 감지하고 통계정보를 유지한다. Reader가 태그를 read/write/lock/kill 등의 명령의 성공/실패 여부를 판단하여 그 명령의 count를 유지하고 통계정보를 가지고 있는다. 만약 실패일 경우에는 Alarm을 발생하게 된다.

✓ AlarmControl Object

Reader의 현재상태 조작상태들을 알려고 할 때 그 상태 정보를 유지하고 있는 Object다. 모든 Object에 AlarmControl을 지원한다.

✓ TTOperationalStatusAlarmControl Object

Reader의 상태가 바뀌었을 때 바뀌기 전 상태와 바뀐 상태의 정보를 유지하고 있다.

✓ Alarm Object

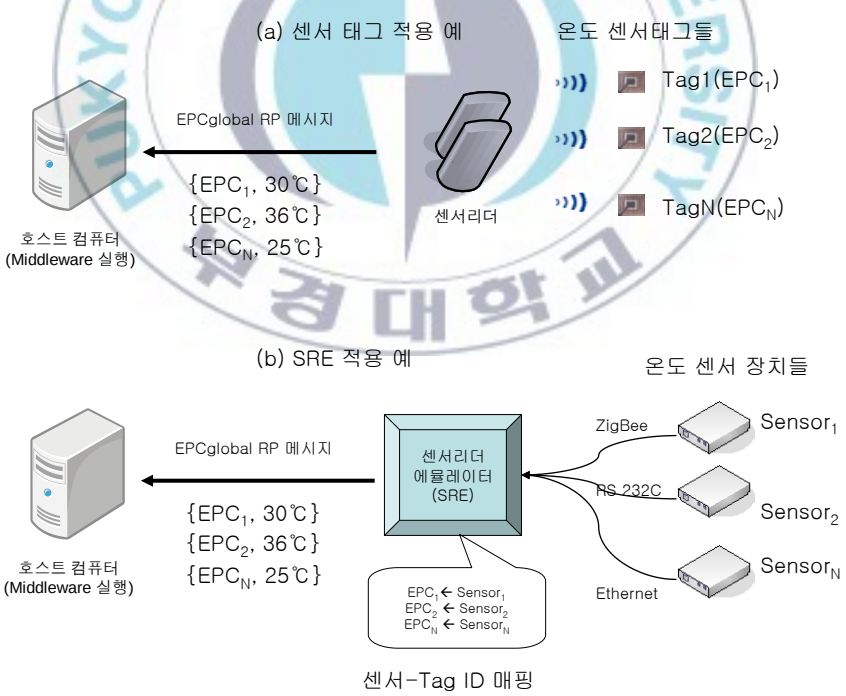
관리 시스템과 Host 어플리케이션에 보내는 메시지이다. Alarm

Object는 추상적인 방법으로 이들의 내용을 Alarm 메시지로 지정한다. AlarmChannel이 실행될 때 Alarm객체가 같이 동작한다.

3. 센서 리더 에뮬레이터의 설계와 구현

본 장에서는 센서리더 에뮬레이터의 개념을 살펴보고 센서장치를 EPCglobal의 표준 프로토콜을 지원하도록 센서 리더 에뮬레이터의 설계 단계를 기술한다.

3.1 센서 리더 에뮬레이터 개념



[그림 6] Sensor Reader Emulator 개념도

위 그림 6은 SRE의 기능을 센서리더와 비교하여 개념적으로 나타낸

것이다. 센서리더는 무선인식 기술을 이용하여 센서태그들로부터 태그ID와 센싱 값을 전달받고, 이것을 이벤트 형식으로 변환한 다음 호스트 컴퓨터에서 실행되는 미들웨어에 전달한다. 전달되는 메시지 포맷은 리더마다 다를 수 있으나 EPCglobal의 표준을 지원하는 리더의 경우에는 EPCglobal Reader Protocol 또는 LLRP 프로토콜[10]에 따라 메시지를 전송할 것이다. SRE의 경우에는 센서장치가 지원하는 통신 프로토콜(ZigBee, RS232C, Ethernet, TCP/IP 등)을 통해 센싱 값만을 전달 받는다. 그리고 센서 장치 별로 미리 부여된 태그ID(EPC)를 검색하여 센서태그[11]와 동일한 형식의 이벤트를 만든 다음 호스트 측으로 전달한다. 따라서 호스트 측에서는 센서리더와 SRE에서 전달되는 메시지를 구분하지 않고 RP 메시지 형식으로 동일하게 사용할 수 있다. SRE를 사용하여 센서장치를 이용하면 다음과 같은 이점이 있다.

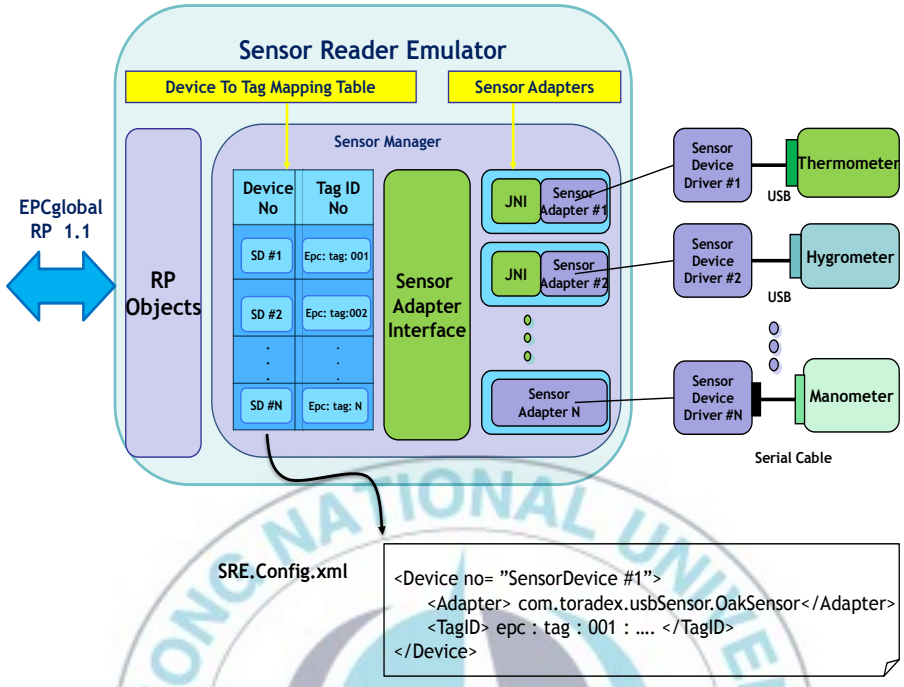
- ✓ 개발자 입장에서는 RFID 리더와 센서 장치를 구분하지 않고 리더형식으로 모든 장치를 접근할 수 있다.
- ✓ 기존의 센서 장치를 센서태그로 대체하거나 장치에 변경을 가하지 않고서도 RFID 미들웨어와 연동할 수 있다.
- ✓ 각 센서 장치에 대해 태그 ID(TAG ID)를 부여하여 전역적으로 식별할 수(globally identifiable) 있다.
- ✓ 응용 프로그램이 센서의 인터페이스를 직접 호출하여 사용하지 않는다. 따라서 센서 장치와 응용 프로그램간의 의존성을 감소시킬 수 있다. 센서 장치 측에서 발생할 수 있는 여러 가지 변

경 사항(예를 들어 센서 장치의 추가, 장치의 설정 변경 등)에 맞추어 응용 프로그램을 변경하는 일을 줄여 준다.

- ✓ 무선인식을 사용하기 어렵거나 불필요한 환경이라면 첫째, 값비싼 센서태그 대신 저렴한 센서 장치로 동일한 효과를 누릴 수 있다. 둘째, 무선인식을 사용하는 센서리더 보다 안정적으로 센싱 데이터를 수집할 수 있다.

3.2 센서 리더 에뮬레이터 구조

센서 리더 에뮬레이터는 실질적인 리더는 아니지만 EPCglobal의 Reader Protocol 1.1을 이용해서 마치 리더처럼 보이게 하는 것이 궁극적인 목적이다. 또한 센서장치가 하나가 아닌 여러 개가 붙을 수 있는데 이러한 경우를 대비해서 내부적으로 여러 개의 센서장치가 붙더라도 일관된 형식으로 센서 값을 처리 할 수 있도록 한다.



[그림 7] Sensor Reader Emulator 아키텍처

그림 7은 본 논문에서 제안하는 센서 리더 에뮬레이터의 아키텍처이다. 실질적인 센서장치가 한 개 또는 그 이상 붙을 수 있도록 각 센서장치에 해당하는 SensorAdapter를 추가적으로 구성해야 한다. 그리고 센서장치의 Vendor specific protocol은 일반적으로 C 언어로 이루어져 있어서 JNI(Java Native Interface) 인터페이스를 이용한다. JNI 인터페이스를 이용하여서 각각의 Sensor Adapter를 일관되게 Sensor Adapter Interface와 연결 시켜 준다. Sensor Adapter단에서는 위와 같이 JNI 인터페이스가 필요한 경우도 있겠지만 불필요하다면 거기에 해당하는 Sensor Adapter를 구성하면 된다. Sensor Adapter Interface는 서로

다른 Sensor Adapter가 있을 경우를 대비해서 일관되게 인식 할 수 있도록 연결해주는 통로가 될 것이다. HashTable을 사용하여 각각의 센서에 고유한 태그 식별자를 부여한다.

Sensor Reader Emulator의 중요구성은 다음과 같다.

- ✓ Sensor Device Driver

  실질적인 센서장치를 말한다. 이러한 센서장치는 한 개 또는 그 이상이 올 수가 있다. 본 연구에서는 Oak Sensor를 사용하였다.

- ✓ SensorAdapter

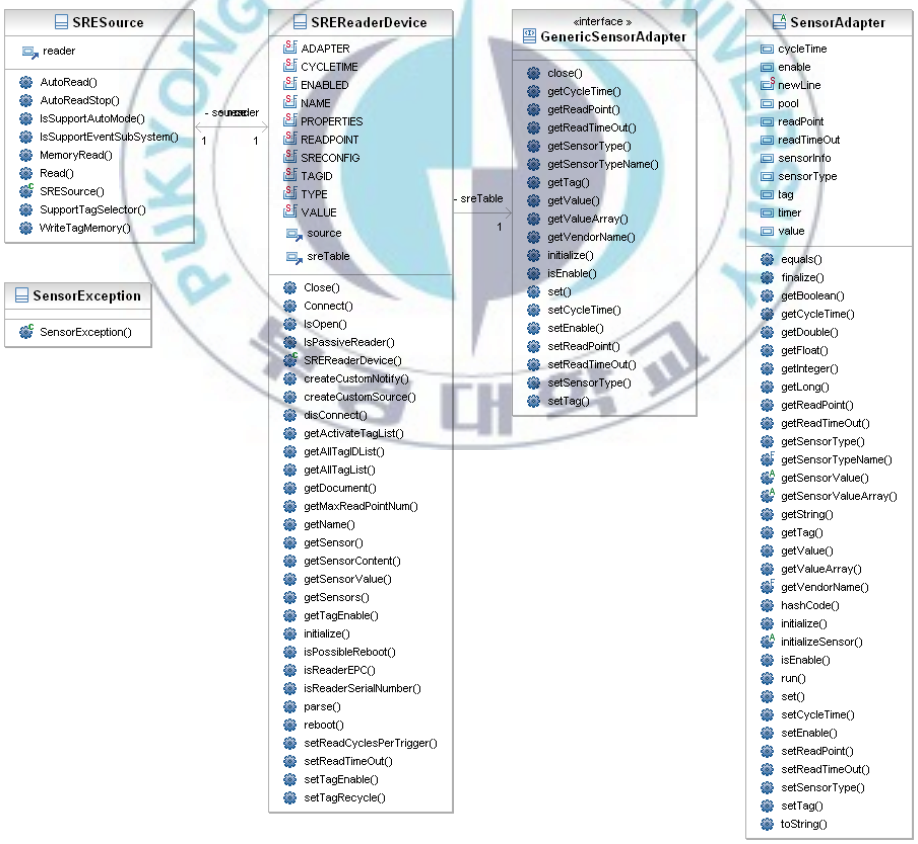
  센서장치와 연결을 하기 위한 Class로써 각 회사마다 프로토콜이 틀리기 때문에 각각의 Sensor Device마다 이러한 Adapter를 만들어야 한다. 다른 센서장치가 연결되었을 때는 제품명 + Sensor 라고 칭하게 되면 여러 개의 센서장치가 있더라도 유지 및 보수에 효율적이다.

- ✓ Sensor Adapter Interface

  N개의 센서장치가 있다고 가정할 때 해당하는 Adapter도 N개가 된다. 이러한 N개에 해당하는 Adapter들은 각 각의 다른 인터페이스를 사용하고 있기 때문에 서로 다른 인터페이스를 일관된 형식으로 인식 할 수 있도록 구현된 인터페이스이다.

3.3 주요 클래스

SRE는 개념적으로 RP의 스펙을 구현하기 위한 RP변환부와 센서어댑터관리부라는 두 개의 모듈로 구성된다. 센서어댑터관리부에서 측정된 센싱 값은 해당 센서장치에 부여된 TAG ID와 함께 RP 메시지 변환부로 전달된다. RP변환부는 전달 받은 TAG ID와 센싱값을 4단계의 서브 시스템을 통해 RP의 이벤트형 메시지로 변환하여 호스트로 전달한다.



[그림 8] Class Diagram

그림 8dms 센서 리더 에뮬레이터의 Class Diagram이다. GenericSensorAdapter는 인터페이스로서 SensorAdapter가 실행한다. SensorAdapter 클래스는 추상 클래스로써 센서장치를 컨트롤 할 수 있는 기본적인 역할은 여기서 담당하게 된다. 센서장치의 기본 정보나 센싱값 추출하고 기본적으로 센서장치의 Adapter를 만들기 위해서 상속을 해주는 역할 등을 하고 있다.

3.3.1 SRECustomProtocol Class

Host에서 받는 각 각의 CommandMessage를 실제 장비에 명령을 보낼 수 있는 메시지로 정의하는 역할을 한다.

표 1 SRECustomProtocol class

원형 함수	설명
READ	“Read” : One Time으로 태그 수집 명령
READALL	“Get AllTagList” : 모든 센서 목록을 포함하는 수집 명령
QUIT	“Quit” : Sensor Device와의 접속 종료 명령
REBOOT	Reboot” : Sensor Device의 재 부팅 명령
GETREADERNAME	“Get ReaderName” : Sensor Device의 이름 호출 명령
GETONETAG	“Get A Tag” : 한 개의 지정된 Sensor에 접근해서 센서 값을 가져오는 명령

3.3.2 SREReaderDevice Class

Middleware와 Sensor Device를 연결하고 관리 명령을 수행하는 부분이다.

표 2 SREReaderDevice Class

원형 함수	설명
connect()	Sensor device와 연결을 한다.
reboot()	Sensor Device의 재 부팅을 한다.
getAllTagList()	모든(활성화/비활성화) 태그와 센서 값을 상위로 올려준다.
getAllTagIDList()	모든 태그 ID를 상위로 올려준다.
getSensorValue()	지정된 센서의 값을 올려준다.
getActivateTagList()	활성화 되어있는 모든 태그와 센싱 값을 상위로 올려준다.

3.3.3 SRESource Class

상위에서 명령이 내려진 요청이 SRESource에서 받아 들여진다. 이렇게 받은 Data는 SREReaderDevice로 보내어져서 Sensor Device로 접근한다.

표 3 SRESource Class

원형 함수	설명
read()	태그를 읽는 명령
autoRead()	자동읽기 시작
autoReadStop()	자동읽기 정지

3.3.4 GenericSensorAdapter Class

Sensor Device의 초기화, 종료, 값을 읽어 들이는 부분을 담당하는 모듈로서 인터페이스를 제공한다.

표 4 GenericSensorAdapter Class

함수원형	설명
Initialize()	Sensor를 초기화한다.
getValue()	실제 Sensor값을 가져오거나, 캐싱되어진 값을 가져온다.
getValueArray()	실제 Sensor의 모든 값을 가져온다.
getTag()	명명된 태그를 가져온다.
setEnable()	Sensor의 활성화 여부를 지정한다.
isEnable()	Sensor 활성화 여부를 가져온다.
getVendorName()	Middleware에서 정의된 FiledName을 가져온

	다.
close()	SREReaderDevice를 종료한다.
setCycleTime()	지정된 Cycle Time 동안 Buffer에 담아 두고 있다가 SREReaderDevice에서 필요할 때 가장 최근 값을 넘겨준다.

3.3.5 SensorAdapter

추상 클래스로서 Sensor Device를 컨트롤 할 수 있는 기본 적인 역할을 담당하고 있다.

3.3.6 생성파일 구조

본 논문에서 사용한 Oak sensor에 대한 정보를 가지고 있다. <adapter>에서 OakSensorAdapter가 지정된 경로를 찾고, <tagID>에서 센서 장치의 TagID를 맵핑 한다. 기본적으로 Cycle Time은 3000 ms로 지정한다.

표 5 SREConfig.xml 예

```
<?xml version="1.0"?>
<!DOCTYPE devicelist SYSTEM "SREconfig.dtd">
<devicelist>
        <device>
        <adapter>com.lit.rpsys.customreaderdevices.oaksensordevice.OakSe
nsorAdapter</adapter>
                <tagID>010000A8900016F000169DC0</tagID>
                <readPoint>ANT1</readPoint>
                <cycleTime>3000</cycleTime>
                <enable>true</enable>
                <sensorType>LIT:HUMIDITY</sensorType>
                <properties>
                    <property name="id" value="T7B270013" />
                    <property name="channel" value="1"/>
                </properties>
        </device>
        <device>
        <adapter>com.lit.rpsys.customreaderdevices.oaksensordevice.OakSe
nsorAdapter</adapter>
                <tagID>0100B2E0200029D00081D39F</tagID>
                <readPoint>ANT2</readPoint>
                <cycleTime>3000</cycleTime>
                <enable>true</enable>
```

```
                <sensorType>LIT:TEMPERATURE</sensorType>
                <properties>
                        <property name="id" value="T7B270013"/>
                        <property name="channel" value="2"/>
                </properties>
        </device>
</devicelist>
```

3.4 센서값의 전달

그림 9는은 RP 변환부와 센서어댑터관리부간의 센싱 정보 전달과정을 나타낸 것이다. RP 변환부에서 TAG ID를 읽어 오는 기능은 Source 객체가 담당하며, 그 시작은 RP 변환부의 Read Trigger에 의해 시작된다. 다음은 Trigger의 Read 요청 이후 수행되는 절차이다.

① Source 객체는 센서어댑터관리자에 현재 등록된 모든 센서들의 리스트를 질의한다(getSensorIds).

② 센서어댑터관리부는 등록된 센서어댑터들의 인터페이스를 이용하여 TAG ID들을 알려준다(4. TAG IDs).

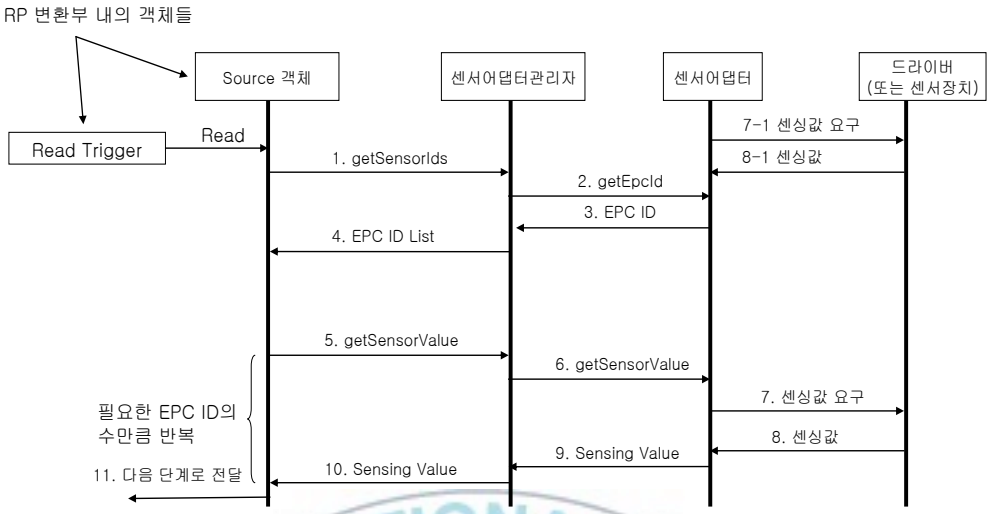
③ Source 객체는 전달 받은 센서들의 TAG ID 들 중에서 필요한

만큼 반복해서 센서어댑터 관리자에 getSensorValue를 호출한다. 이때 전달되는 TAG ID는 ②에서 얻어진 모든 TAG ID들이 대상이 될 수도 있고 Source 내에서 Event Smoothing과정을 거쳐 선택된 일부 TAG ID들에 대해서만 호출될 수도 있다.

④ 센서어댑터 관리자는 TAG ID에 해당하는 센서어댑터를 찾아 getSensorValue를 호출한다.

⑤ 센서 어댑터는 자신의 내부에 캐쉬되어(caching, 그림 5의 7-1과 8-1) 있는 센싱값을 전달하거나, 센서 장치에 요청하여 센싱값을 알아낸다(그림 5의 7~8). 센싱값은 getSensorValue호출시 전달되는 fieldValues 객체에 '{태그필드명, 센싱값(문자열)}' 로 삽입된다. 예를 들어 해당 센서의 값이 태그 필드 'LIT_SRE:TEMP' 로 주어진 것이고 온도 값을 전달하는 것이라면 { 'LIT_SRE:TEMP' , '36.5' }의 쌍이 추가된다.

⑥ 위 과정을 거쳐 TAG ID들과 센싱값을 전달받은 Source 객체는 전달받은 센싱값을 포함한 태그필드 값을 TAG ID의 쌍과 함께 다음 RP 처리과정으로 전달한다.



[그림 9] 센서어댑터에서 RP변환부로 센싱정보가 전달되는 과정

3.5 이벤트 생성

일반적으로 RFID 리더의 읽는 다소 불안정하다. 즉 리더의 읽기가능 영역 내에 존재하더라도 태그가 읽혀지지 않는 경우가 있는 반면 의도하지 않던 영역에 존재하는 태그가 읽히는 경우도 허다하다. RP에서는 이러한 불안정성을 극복하고 동일한 태그가 지나치게 자주 계속해서 보고되는 것은 막기 위해 이벤트 서브시스템을 도입하였다. 이것은 새로 읽혀진 태그와 사라진 태그들은 즉각 보고하되 계속해서 읽혀지는 태그들은 간헐적으로 보고하도록 하는 기법이다.

SRE에서는 센서와의 접속을 통해 센싱값을 성공적으로 읽어 올 수 있는 경우에는 태그가 읽힌 것으로 그렇지 않은 경우에는 태그를 읽지 못한 것으로 모델링한다. 따라서 통신 에러, 센서 장치의 이상 등으로 인해 센서어댑터가 readCycleTime 내에 센싱값을 전달받지 못하면 태그가 읽히지 않은 것으로 간주된다. 센서장치들의 경우 Ethernet, TCP/IP, RS232C, Bluetooth, ZigBee 등과 같은 일반적인 통신 프로토콜을 통해 센서어댑터 연결되므로 RFID 태그 읽기에 비해 매우 안정적인 통신이 이루어진다. 따라서 RFID 태그와는 달리 매우 단조로운 이벤트 만이 보고될 것이다.

4. 적용결과

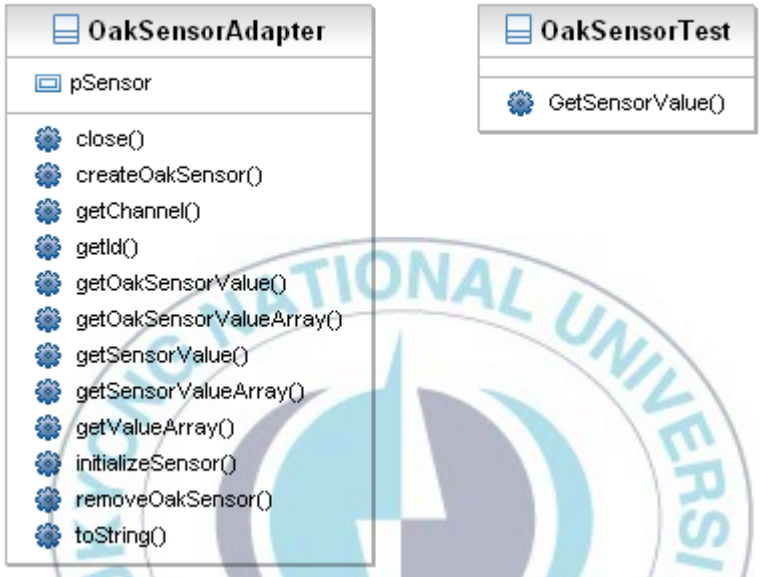
본 논문에서 제안하는 센서 에뮬레이터의 실용성을 확인 하기 위해 그림 10에 나와있는 사진과 같은 USB 센서장치[12]에 적용해 보았다.

[그림 10] Oak Sensor Device

본 연구에서 사용한 USB 센서장치는 Toradex사의 Oak Sensor Device를 이용하였다. 이 장치는 온도와 습도를 센싱할 수 있다. 이러한 장치에서 온도와 습도를 인식하여 미리 설정해놓은 설정파일에서 ECP코드를 맵핑해서 마치 리더에서 읽어 들인 정보를 상위 미들웨어로 전달

할 수 있다.

다음 그림 11은 본 연구에서 사용한 Oak Sensor Device를 SRE를 이용해서 미들웨어로 올려주기 위한 Adapter Class diagram이다.



[그림 11] Oak Sensor Adapter의 Class

다음 표는 위 Oak Sensor Adapter의 중요 함수들이다.

표 6 Oak Sensor Adapter 함수

함수원형	설명
close()	Oak Sensor장치를 종료
createOakSensor()	Oak Sensor장치의 시작
getChannel()	ex)온도 or 습도

getId()	설정파일에 미리 설정해둔 EPC TagID를 가져온다.
getOakSensorValue()	Oak Sensor장치에서 값을 가져온다.
getSensorValue()	Cycle Time에 설정되어있는 시간동안에 마지막으로 읽어 들인 센싱 값을 가져온다.
initializeSensor()	초기화 담당
removeOakSensor()	Oak Sensor장치 제거

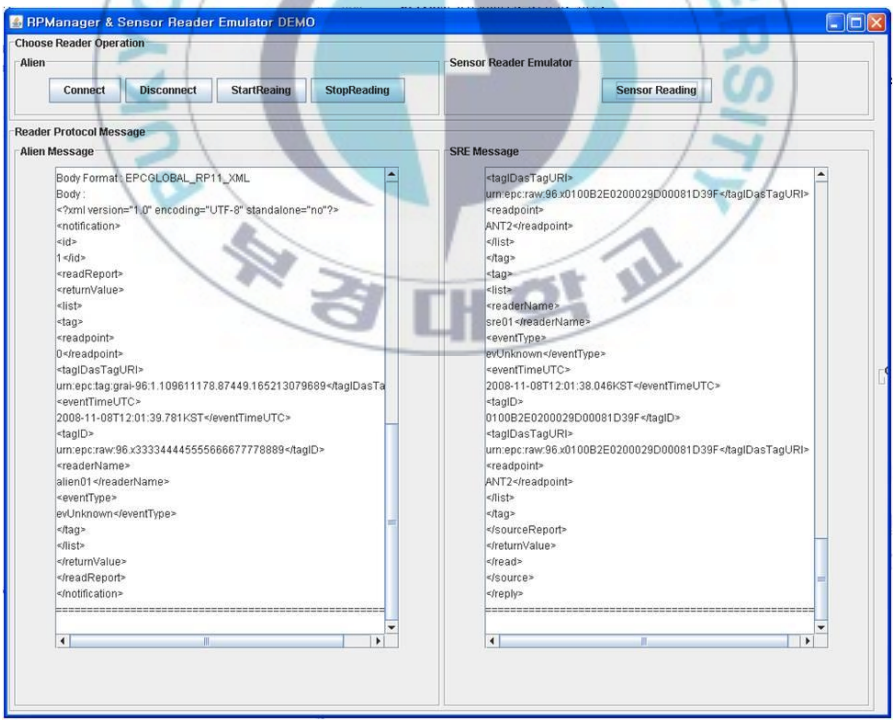
위 표6과 같이 어떠한 센서장치가 연결되더라도 간단하게 인터페이스에 준수하여 설계하면 마치 센서장치가 센서리더에서 읽은 Tag와 동일한 역할을 할 수 있다.

표 7 설정파일의 필드

필드명	설명
adapter	센서장치의 클래스파일의 위치를 미리 지정
tagID	EPC코드를 미리 부여하여 센싱값과 맵핑하는역할
cycleTime	기본적으로 Cycle Time을 지정해두면 그 시감만큼 센싱값을 읽어들여 버퍼에 넣어두는 역할

type	ex)온도 / 습도/ 압력
properties	Channel을 지정하는 역할 ex)온도 : channel-1 습도 : channel-2

위 표 7과 같이 설정파일을 설정하면 TagID를 이용해서 센싱값을 맵핑 수 있다. 또한 channel을 이용해서 온도를 읽을 것인지 습도를 읽을 것인지 설정도 가능하다.



[그림 12] 구현 결과

위 그림 12는 본 연구에서 만든 GUI화면이다. 왼쪽의 메시지는 Alien9800 리더에서 태그를 읽은 메시지이고, 오른쪽 메시지는 본 연구 센서 리더 에뮬레이터를 통한 센싱 값을 읽은 메시지이다. 두 메시지는 EPCglobal에서 제안하는 국제 표준인 Reader Protocol메시지와 동일하게 전달되는 것을 확인 하였다. 따라서 본 논문에서 제안하는 것과 같이 구현한 결과 센서장치가 마치 하나의 센서리더와 같은 동작을 할 수 있다는 것을 확인하였다.

5. 결론

Sensor Reader Emulator는 센서장치에 대한 새로운 접근 방식이다. RFID 리더와 센서장치를 동시에 사용하는 경우 응용프로그램에서는 리더에 접근하기 위해서 Reader Protocol을 사용하고 센서장치에 접근방식은 디바이스 드라이버를 이용한 함수 호출 방식을 사용한다. 개발자의 입장에서 이것은 복잡도를 증가시키는 요인으로 볼 수 있다. 이에 본 논문에서는 센서장치를 센서태그 형식으로 사용할 수 있게 하는 SRE 시스템을 제안한다. 이 프로그램을 사용하면 RFID 리더와 센서 장치들을 단일한 인터페이스를 사용하여 접근할 수 있음으로써 여러가지 장점을 제공하게 된다. 현재는 EPCglobal RP 1.1 표준에 따라서 센서 장치를 접근할 수 있으며 추후에는 LLRP 1.01 프로토콜도 추가적으로 지원할 예정이다. 또한 아직 구현되지 않은 Gen2 태그의 기능 중에서 지원이 가능한 항목들 – User Memory[13][14] 지원, Security 지원 – 에 대한 연구가 필요하다.

참고문헌

[1] EPCglobal, Reader Protocol Standard Version 1.1 Ratified Standard June 21, 2006

[2] EPCglobal, EPC Information Service (EPCIS) Version 1.0.1 Specification, September 21, 2007

[3] EPCglobal, http://www.epcglobalinc.org

[4] 안재명, 이종태, 오해석, ㈜리테일테크 기술연구소 공저 "EPCglobal Network기반의 RFID 기술 및 활용", 2007년 2월

[5] EPCglobal, Object Naming Service(ONS) Version 1.0 Ratified Specification Version of October 4, 2005

[6] EPCglobal, Tag Data Standards Version 1.3 Ratified Specification March 8, 2006

[7] EPCglobal, Radio-Frequency Identity Protocols Class-1 generation-2 UHF RFID Protocol for Communications at 860 MHz-960MHz Version 1.2.0

[8] EPCglobal, The Application Level Events Specification, Version 1.1 Ratified Specification as of February 27, 2008

[9]] EPCglobal, Reader Management 1.0.1 Specification, May 31, 2007

[10] EPCglobal, Low Level Reader Protocol(LLRP), Version 1.0.1

Ratified Standard with Approved Fixed Errata August 13, 2007

[11] 빌 글로버, 히만슈 바트 공저, 서환수 역, 실무자를 위한 RFID 이해와 활용, 2007년 2월

[12] Toradex Oak USB sensor device, http://toradex.com/En/Products/Oak_USB_Sensors

[13] ISO/IEC, ISO/IEC 15961(Data Protocol: Application Interface)

[14] ISO/IEC, ISO/IEC 15962(Data Protocol: Data Encoding Rules and Logical Memory Functions)

감사의 글

2년이라는 시간이 지나고 이렇게 감사의 글을 적고 있으니 그 동안의 기억들이 새록새록 나네요. 좀 더 높고 넓은 꿈을 위해 시작했던 대학원에서의 생활들 크고 작은 힘든 시간을 지나 한 발짝 성장한 모습으로, 이제 많은 분들의 격려와 도움을 받아 이 작은 결실과 함께 마무리 하려고 합니다.

먼저, 부족한 모습이었던 저에게 2년이라는 시간 동안 저의 앞길을 등불처럼 밝혀주신 송하주 교수님께 감사의 말씀을 드립니다. 또한 같은 연구실 연구원들 정신적지주신 명환 형님, 많은 정보를 알려주시는 태순 형님, 좋은 길로 인도해주시는 영규 형님, 긍정적인 종민군, 인사성 밝은 태용군, 똑똑이 성진군, 노력파 승혁군, 의리파 영민군, 얼짱 영준군, 막내 주형군에게도 감사의 말을 전합니다. 그리고 2년 동안 서로 정보교류나 행사가 있을 때 격려해준 대학원 동기와 선 후배님들께도 감사의 말을 전합니다.

마지막으로 2년 동안 저의 뒤에서 아무 말 없이 묵묵히 지켜보시고 힘을 주신 아버지, 어머니, 동생에게도 이 글을 받칩니다.

이렇게 한 지면에 저에게 있어 고마움과 감사의 마음을 표현하지는 못하지만 항상 마음으로 감사함을 느끼고 있습니다. 훗날 이 감사함을 잊지 않고 깊이 간직하며 사회에서도 열심히 하는 재목이 되어 보답하겠습니다.

2009년 2월

이 화 춘