碩士學位論文

ALE 미들웨어를 위한 다양한 센서 데이터 스트림 처리에 관한 연구

濟州大學校 大學院

컴퓨터工學科

梁 文 碩

2008年 12月

ALE 미들웨어를 위한 다양한 센서 데이터 스트림 처리에 관한 연구

指導敎授 邊 暎 哲

梁 文 碩

이 論文을 工學 碩士學位 論文으로 提出함

2008年 12月

梁文碩의 工學 碩士學位 論文은 認准함

審査委員長 _____ 印

委 員 _____ 印

委 員 _____ 印

濟州大學校 大學院

2008年 12月

A study of multi-sensor data stream processing on based ALE middleware

Moon-Seok Yang

(Supervised by professor Yung-Cheol Byun)

A thesis submitted in partial fulfillment of the requirement for the degree of Master of Computer Engineering

2008. 12.

This thesis has been examined and approved.

Thesis director, _____

Thesis director, _____

Thesis director, _____

December 2008

Department of Computer Engineering

Graduate School

Cheju National University

감사의 글

어느덧 새로운 각오를 갖고 도전을 시작한지 벌써 2년이 지나 대학원 생활을 마치고 이렇게 졸업논문을 쓰면서 감사의 말을 전하게 되었습니다.
먼저 대학원 생활동안 미려하고 부족한 저를 지도해주신 지도교수님인 변영철 교수님께 감사의 말을 전하고 싶습니다. 그리고 논문의 완성을 위해 지도해주신 안기중 교수님과 곽호영 교수님께도 감사를 드립니다. 또한 많은 가르침을 주신 김장형 교수님, 변상용 교수님, 이상준 교수님, 송왕철 교수님, 김도현 교수님께도 감사를 드립니다.
함께 연구실에서 생활하면서 많은 도움을 주던 영식씨, 동기인 지웅이, 막내 지윤이 그리고 학부생들인 요종이, 가영이, 혜연이, 대오에게도 감사의 말을 전하고 싶습니다. 그리고 연구실 선배인 연미누나, 대학원 선배인 한경복 박사님, 권훈씨, 창영이에게도 고맙다는 말을 전하고 싶습니다. 또한 대학원 생활의 실질적인 도움을 주셨던 정은경 선생님과 이정화 선생님께 감사하다는 말을 전하고 싶습니다.
끝으로 어려움에 있어서 용기를 북돋아주던 친구들과 항상 모자란 저를 지켜봐 주시고 아낌없이 사랑해주시는 부모님 그리고 형, 누나들에게도 감사를 드립니다.

목 차

그림 목차

표 목차

국문초록

ALE 미들웨어를 위한 다양한 센서 데이터 스트림 처리에 관한 연구

컴퓨터공학과 양문석
지 도 교 수 변영철

마크 와이저에 의하여 유비쿼터스 컴퓨팅이 주창된 이후 이를 실현하기 위한 기술로서 USN(Ubiquitous Sensor Networks)과 RFID(Radio Frequency Identification)에 관한 연구가 활발하게 이루어지고 있다. 유비쿼터스 컴퓨팅을 실현하기 위한 기술인 RFID와 USN은 기술적 유사성과 상호 영향에도 불구하고 별개의 연구로 인식되어 RFID와 USN의 기술적인 융합에 대한 연구는 미미하다. 본 논문에서는 USN과 RFID의 융합뿐만 아니라 비용과 확장성 면에서의 효율성을 제공하기 위하여 USN 환경의 다양한 센서 데이터를 RFID 미들웨어서 쉽게 처리할 수 있는 방법을 제안한다. 특히 RFID 미들웨어 내부에서 처리할 수 있도록 다양한 유형의 센서 데이터를 EPC 데이터 포맷으로 변환하는 방법을 제안함으로써 RFID 미들웨어에서 센서 데이터를 효율적으로 처리할 수 있다. 본 방법을 이용하면 국제 표준 스펙에 기반한 RFID 미들웨어가 RFID 태그 데이터뿐만 아니라 일반 센서 데이터를 효율적으로 처리할 수 있기 때문에 사물의 인식 정보와 센서 데이터를 활용한 다양한 유비쿼터스 응용 서비스를 저비용으로 구현할 수 있다.

ABSTRACT

A study of multi-sensor data stream processing on based ALE middleware

YANG, MOON-SEOK
Department of Computer Engineering
Graduate School
Cheju National University

From the time when ubiquitous computing was proposed by Mark Weiser, a lot of people are intensively investigating on the topics related to ubiquitous sensor networks(USN) and radio frequency identification(RFID). USN and RFID, as new technologies for realization of ubiquitous computing, are closely related with each other technically. Nevertheless, these technologies are recognized as a separate research topic, and there are few researches on technical convergence of the two research areas. In this paper, we propose an efficient processing method of various kinds of sensor data which are used in USN environment to provide efficiency from the view point of not only convergence of USN and RFID technologies but also cost and extensibility. Especially, we propose a method of conversion of various sensor data into EPC data format which can be handled in a RFID middleware. Using the proposed method, RFID middleware systems based on international standard specification can handle not only RFID tag data but also general sensor data efficiently, and various types of ubiquitous application services can be implemented at low cost and in a short time.

약어표

USN	Ubiquitous Sensor Network
RFID	Radio Frequency Identification
IT	Information Technology
ALE	Application Level Event
EPC	Electronic Product Code
MIT	Massachusetts Institute of Technology
SAG	Software Action Group
RF	Radio Frequency
API	Application Program Interface
EPCIS	Electronic Product Code Information Service
ONS	Object Name Service
WG	Working Group
GID	General IDentifier
EAN	European Article Number
PML	Physical Markup Language
XML	eXtensible Markup Language
QoS	Quality of Service
URN	Uniform Resource Names

I. 서 론

1. 연구 배경

최근 정보통신 분야의 최대의 화두는 단연 유비쿼터스 컴퓨팅(Ubiquitous Computing)이다. 유비쿼터스 컴퓨팅은 21세기 새로운 IT 혁명으로 불리며, 상상도 하지 못할 정도로 사회·경제·문화 등 모든 분야에 큰 영향을 미치게 될 것이다. 유비쿼터스 컴퓨팅은 다양한 컴퓨터가 현실 세계의 디바이스, 환경 및 사물들 속으로 스며들어 언제, 어디서나, 어떠한 기기로도 통신 서비스를 이용할 수 있는 인간, 사물, 공간 간의 최적의 컴퓨팅 및 네트워크 환경을 구축함으로써 생활 속으로 자연스럽고 편리하게 컴퓨터를 사용할 수 있는 기술을 의미한다[1].

마크 와이저에 의하여 유비쿼터스 컴퓨팅이 주창된 이후 이를 실현하기 위한 기술로서 USN (Ubiquitous Sensor Networks)과 RFID (Radio Frequency Identification)에 관한 연구가 활발하게 이루어지고 있다[2]. USN은 필요한 모든 곳에 센서를 부착하고, 이를 통해 사물의 인식 정보를 기본으로 온도, 압력, 오염, 균열 등의 주변 환경 정보까지 각종 센서를 통하여 실시간 수집하여 관리, 통제할 수 있도록 구성한 네트워크이다[3]. RFID는 각종 물품에 전자 태그를 부착하여 사물의 정보와 주변 환경 정보를 무선 주파수로 전송·처리하는 비접촉식 인식 기술이다[4]. 그러나 RFID와 USN은 기술적 유사성과 상호 영향에도 불구하고 별개의 연구로 인식되어 RFID와 USN의 기술적인 융합에 대한 연구는 미미한 실정이다.

최근에는 RFID 기술을 이용하여 물류, 소매업, 의료, 공장·가정·사무실 자동화, 보안, 재난 방지, 재산 관리 등 다양한 서비스를 제공하기 위한 응용 및 기술들이 연구 개발되고 있다[5, 6, 7]. 이와 같이 RFID 기술을 이용한 유비쿼터스 응용 서비스를 쉽게 구축할 수 있으려면 RFID 태그가 부착된 객체와 응용 서비스 사이에서 교량 역할을 하는 미들웨어가 필요하다[8].

RFID 미들웨어는 하드웨어(리더)와 애플리케이션의 중간에 위치해 다수의 리

더를 관리하고, 이기종 운영체제 간 상호협력이 가능하며, 분산처리의 신뢰성, 네트워크의 독립성, 응용 프로그램 및 서비스간의 상호 운용성 및 투명성을 지원한다. 여러 가지 센서를 관리하고 센서의 프로토콜을 이용하여 데이터를 수집하며, 또한 수집되어 가공되지 않은 데이터로부터 의미 있는 정보, 혹은 응용이 사용하기 쉬운 형태의 정보를 추출하여 응용 서비스에 전달하는 기능을 수행한다[9].

한편, 기존의 국제 표준을 따르는 미들웨어 스펙인 ALE(Application Level Events)[10]에서는 일반적인 관점에서 RFID 리더 장치도 태그를 인식하는 센서이므로 미들웨어에서 RFID 태그 데이터뿐만 아니라 다양한 유형의 센서 데이터를 처리할 수 있을 경우 효율적인 유비쿼터스 서비스를 제공할 수 있다.

사실상의 표준인 ALE 스펙은 유비쿼터스 서비스를 위한 RFID 미들웨어에서의 태그 데이터 처리를 위한 기능과 내용을 기술하고 있다. ALE 스펙에서 기술한 필터링, 그룹핑, 카운팅 등 주요 기능들과 내용들은 RFID 태그 데이터뿐만 아니라 센서 데이터 처리에 필요한 일반적인 내용까지 담고 있다. 즉, 여러 프로젝트에서 연구되고 있는 USN 미들웨어 중 이벤트 기반의 server-side 미들웨어의 기능을 기술하고 있다. 이에 따라 미들웨어에서 RFID 태그 데이터이외에 센서 데이터 처리가 가능해지면 서비스 이용자들은 리더에서 읽은 태그정보를 사용자 요청에 따라 이벤트 기반의 정보를 제공하듯이 센서에서 수집된 정보를 사용자 요청에 따른 서비스를 할 수 있다.

2. 연구 목적 및 방법

본 논문에서는 USN과 RFID의 융합뿐만 아니라 비용과 확장성 면에서의 효율성을 제공하기 위하여 USN의 환경의 다양한 센서를 ALE 기반의 미들웨어에서 처리할 수 있는 방법을 제안한다. 제안하는 방법은 RFID 리더기를 태그 정보를 센싱하는 하나의 센서로 간주하여 리더기이외의 일반적인 센서가 센싱한 정보를 미들웨어가 태그 데이터를 처리하는 것과 같이 센서를 데이터를 처리하도록 하는 것이다. 미들웨어로 입력되는 일반 센서 데이터를 ALE 기반의 미들웨어 내부

에서 사용하는 EPC(Electronic Product Code)[11] 코드처럼 센서 데이터를 EPC 데이터화하여 일반 센서 데이터를 처리한다. 이와 같은 방법으로 사실상의 국제 표준의 ALE 기반의 미들웨어에서 기존의 방법을 변경하지 않고서도 RFID 태그가 아닌 다양한 센서 데이터 스트림을 효과적으로 처리할 수 있도록 한다.

RFID 태그 정보를 통하여 사물의 인식 정보만을 처리하는 것만이 아니라 RFID 태그이외의 센서 데이터를 처리할 수 있는 미들웨어를 개발할 수 있으므로 사물의 인식 정보와 센서 데이터를 활용하여 다양한 응용 서비스를 저비용으로 제공할 수 있다. 또한, 응용과 다양한 센서들의 표준화된 인터페이스를 미들웨어가 제공함으로써 이기종 센서를 저렴한 비용으로 쉽게 연결할 수 있는 범용성을 가질 수 있다.

3. 논문 구성

본 논문의 구성은 다음과 같다. II장에서는 관련 연구 및 기술로 RFID 기술 동향 및 표준화 동향과 USN 미들웨어 동향, RFID 기반 미들웨어 제품 및 솔루션, 센서 데이터 처리를 위한 고려사항 등을 분석하며, III장에서는 센서 데이터 처리를 위한 시스템 구성과 센서 유형, ALE 미들웨어에서 센서 데이터를 처리하기 위한 방법에 대해 설명한다. IV장에서는 III장에서 제안한 방법의 구현 및 실험한 결과를 살펴보고, 마지막 V장에서는 본 연구의 결론에 대하여 설명한다.

II. 관련 연구 및 고려사항

1. RFID 기술 동향

1) RFID 표준화 동향

RFID와 관련된 국제표준화는 사실상의 표준을 제시하고 있는 EPCglobal을 중심으로 활발히 이루어지고 있다. EPCglobal[12]은 기존 MIT Auto-ID[13] 센터에서 개발한 RF 기반의 자동인식 시스템 기술을 표준화하고 상용화하기 위해 2003년 10월에 설립된 비영리기구로서 Auto-ID 센터에서 개발한 기술의 표준화, 상용화 및 EPC 코드의 보급과 관리 등을 목적으로 활동하고 있다.

EPCglobal은 개별물체의 유일식별자인 EPC 기반의 'Internet of Physical Object'를 구성하기 위한 기술 집합을 EPC Network[14]라 정의하고, EPC Network의 구현에 따른 기술요소 분야를 하드웨어, 소프트웨어, 비즈니스 분야로 구분하여 각 분야별 AG(Action Group)을 구성, EPCglobal 가입 업체 중심으로 기술규격 및 표준제정을 추진하고 있다. 특히, RFID 미들웨어 및 네트워크 시스템과 관련된 소프트웨어 인터페이스 및 표준제정은 SAG(Software Action Group)에서 진행되고 있으며, 이들이 다루고 있는 분야는 다음과 같다.

가. Reader Protocol
리더와 호스트(미들웨어)간의 명령 수행/응답 및 태그정보의 교환과 관련된 표준을 정의한다.

나. Reader Management
개별 RFID 리더기에 대한 환경설정, 모니터링 및 시스템 이상 발생 시 알람과 같은 RFID 리더기 관리에 관한 표준적인 기능목록을 정의한다.

다. Filtering and Collection

응용 프로그램이 복수 개의 태그인식 기기(RFID 리더기 포함)로부터 정제 및 요약된 태그인식 데이터를 표준화된 방식으로 획득하기 위해 필요한 데이터 표현 규격 및 리포팅 방식을 정의한 일련의 소프트웨어 API를 정의한다.

라. EPCIS(Electronic Product Code Information Service)

EPC와 관련된 정보를 획득하고, 관리하며 이를 공유하기 위한 외부 인터페이스를 정의한다. 상품 정보를 관리하고 정보제공 요구가 있을 때 이를 PML(Physical Markup Language)로 표시하여 제공하는 시스템이다.

마. ONS(Object Name Service)

EPC와 관련한 EPCIS 위치 정보 검색서비스를 제공하는 ONS에 대한 구조 및 질의 API를 정의한다.

바. Tag Data Translation

Tag data standards에 정의된 EPC 인코딩/디코딩 규칙을 표현하고, 그 규칙에 근거하여 EPC 코드 변환작업을 수행하는 데 필요한 규격을 정의한다.

사. Security

사용자정보, 데이터 암호화 등 EPC Network 전반에 걸쳐 보안 프레임워크를 제공하기 위한 일종의 가이드라인을 정의하고 권고안을 마련한다.

이상과 같이 SAG는 총 7개의 주제에 대해 분야별 WG(Working Group) 활동을 통해 EPC Network를 구성하기 위한 표준규격을 제시하고 있다. 그림 1은 EPCglobal에서 제안한 EPC Network 구조 및 구성요소를 나타내고 있다.

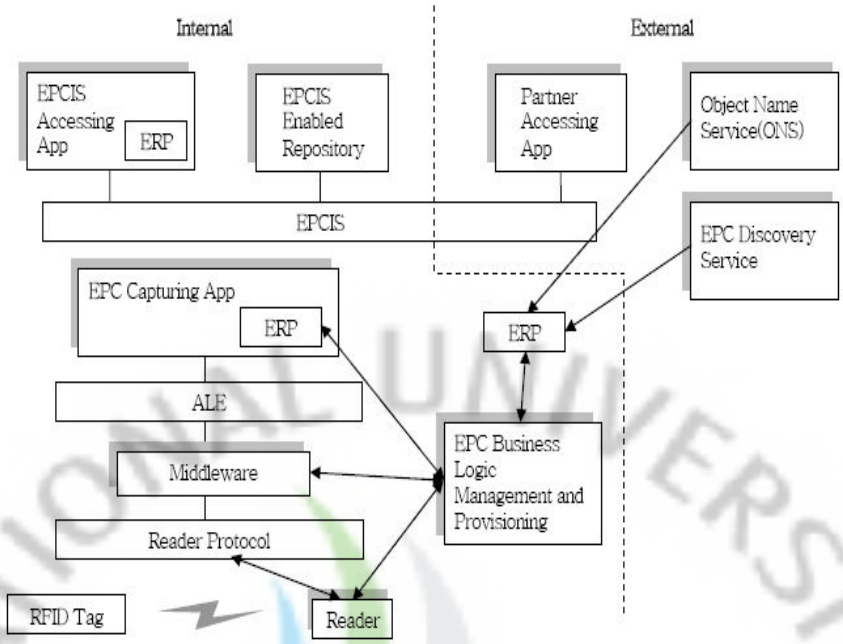

그림 1. EPC Network 구조

각 구성요소에 대해 간략히 살펴보면 다음과 같다. 리더는 리더 인식공간에 존재하는 복수 개의 태그를 인지한다. 이렇게 인식된 태그 데이터는 리더 프로토콜을 통해 리더로부터 미들웨어로 데이터가 전달되게 된다. 미들웨어는 리더로부터 전달된 복수 개의 중복된 데이터를 필터링하는 역할을 수행하게 되고, ALE 인터페이스를 통해 통합/정제된 태그 데이터 목록이 미들웨어에서 EPC Capturing Application으로 전달되게 된다. EPC Capturing Application은 미들웨어로부터 전달 받은 태그인식정보와 기타 비즈니스 관련정보를 EPCIS에 공급하게 되고, 이러한 정보들은 EPCIS-Embedded Repository에 저장/관리되어 향후 질의에 대응할 수 있도록 한다. EPCIS Accessing Application 및 Partner Application은 EPCIS에서 제공하는 인터페이스를 통해 EPCIS에서 관리하고 있는 EPC와 관련된 정보를 질의하고 그에 대한 결과를 활용하여 비즈니스적으로 의미 있는 활동을 수행하게 된다. 기타 EPC Discovery Service는 EPC와 관련된 데이터의 검색 엔진으로서 특정 EPC와 관련된 데이터를 저장하고 있는 EPCIS의 위치정보 목록을 질의의 응답으로 하는 서비스 제공자이다.

다음 그림 2는 EPC Network상에서 EPCglobal의 표준화 대상을 나타내고 있

다. 그림에서 보는 바와 같이 EPCglobal에서는 계층적 구조상에서 인접한 두 계층 간의 인터페이스를 표준화의 대상으로 삼고 있다.

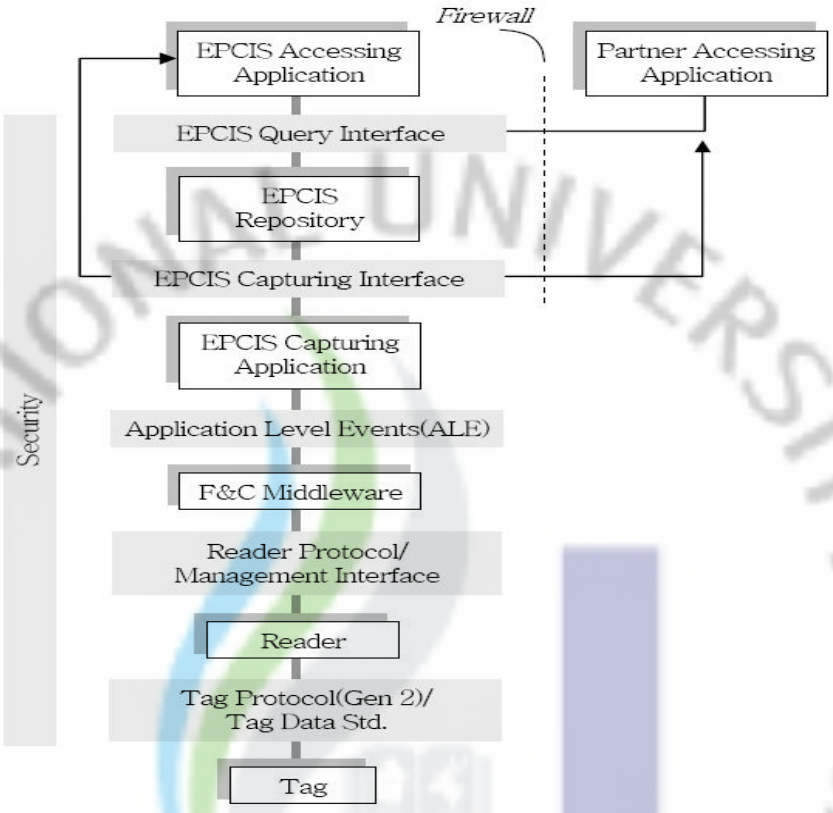

그림 2. EPC Network 표준화 대상

ALE는 일종의 인터페이스로서 RFID 리더로부터 전달된 EPC 데이터를 일정 기간 동안 수집한 후, 중복 혹은 불필요한 정보를 제거한 최종 EPC 목록 및 개수 정보를 일정한 형식으로 표현하여 리포팅하는 일련의 API 형태로 정의하고 있다 [15].

ALE 규격을 이해하는 데 있어 중요한 개념인 이벤트 사이클(event cycle)은 RFID 미들웨어와 응용 애플리케이션간의 최소 연동단위로서, 매 이벤트 사이클 종료 시점에 해당 사이클 동안 수집된 EPC 목록을 응용 애플리케이션에 전달한다. ALE 인터페이스는 이벤트 사이클을 정의하기 위한 API를 정의하는데, 이 API를 통해서 이벤트 사이클의 시/종점, 수집된 EPC 목록에 대한 필터링 및 그

룹핑 조건, 리포팅 형식 등을 설정하게 된다. ALE 클라이언트인 응용 애플리케이션은 원하는 형태의 이벤트 사이클을 정의하여 API를 통해 등록하고, 이를 반복 호출 혹은 구독/발행 방식을 지원하는 API를 통해서 응용 애플리케이션에서 원하는 EPC 목록을 전달받게 된다.

EPCIS는 기업 내/외에 존재하는 서로 다른 애플리케이션 간에 EPC를 매개로 EPC 관련 정보를 공유하고 정보 이력을 관리함으로써 태그가 부착된 객체에 대한 정보에 공통된 시각을 갖도록 하는 데 그 목적이 있다[16]. 그림 2에서 보는 바와 같이 2개의 EPCIS와 관련된 인터페이스가 존재하는데, 하나는 EPCIS Catpuring Interface이고 또 다른 하나는 EPCIS Query Interface이다. 전자는 EPC 인식 정보 및 관련 정보를 수집하기 위한 인터페이스로서, 태그의 발급과 폐기. 태그간의 관계(부착된 객체의 모자관계 등), 특정 지점에서 인식된 EPC 목록의 등록 등 EPC의 인식정보 이외에 관련된 비즈니스적 정보 또한 등록할 수 있도록 API가 정의되어 있다. 후자는 기업 내/외부에 존재하는 응용 애플리케이션이 EPC를 통해 이와 관련된 정보를 질의하기 위한 표준화된 API를 정의하고 있다.

2) RFID 미들웨어

일반적으로 RFID 시스템은 기본적으로 사물에 부착되고 이를 유일하게 구분할 수 있는 정보를 담고 있는 RFID 태그, 태그에 담겨진 정보를 인식하는 RFID 리더기, 마지막으로 리더기로부터 인식된 태그정보를 처리하는 호스트 시스템으로 구성된다[17].

호스트 시스템이란 RFID 리더기로부터 인식된 태그인식정보를 처리하고 응용 프로그램이 이를 활용하는 전반적인 소프트웨어 시스템을 지칭하며, 이중에서 미들웨어는 리더기로부터 인식된 RFID 태그정보를 수집하고 중복된 정보들을 제거하여 의미 있는 정보만을 응용 프로그램에 전달한다. 이러한 RFID 미들웨어는 단순한 태그 인식정보의 전달뿐만 아니라, 서로 다른 형태의 통신 방식 및 프로

토콜을 지원하는 리더기들을 일관된 형식으로 통합적으로 관리하고, 모니터링 할 수 있어야 한다[18].

RFID 미들웨어는 리더에서 계속적으로 발생하는 식별코드 데이터를 수집, 제어, 관리하는 기능을 하며, 모든 구성요소와 연결되어 계층적으로 조직화되고 분산된 구조의 미들웨어 네트워크를 구성하여 서로 통신한다. 미들웨어는 다양한 형태의 리더 인터페이스, 다양한 코드 및 네트워크 연동, 여러 가지 응용 플랫폼에 대해서도 상호 운용성을 보장할 수 있어야 한다[19]. 이를 위해 적어도 다음의 요건들이 만족되어야 한다.

가. 이기종 RFID 리더 시스템 지원 및 관리

RFID 미들웨어는 다수의 이기종 RFID 리더 시스템간의 이질성(예를 들면, 지원하는 태그와 리더간의 프로토콜의 상이함, 리더와 호스트 시스템간의 네트워크 인터페이스의 다양성 등)이 존재하는 환경 하에서, RFID 하드웨어 시스템을 상위계층에서 일관되게 접근이 가능하도록 기능을 제공해야 한다. 리더인식영역에 존재하는 태그정보 수집, 리더기 설정 및 원격제어, 리더 시스템 모니터링 등의 관리가 이뤄질 수 있도록 기능이 제공되어야 한다.

나. RFID 태그 데이터 처리

RFID 태그 데이터가 RFID 리더기로부터 반복적으로 대량의 정보가 유입됨에 따라서, RFID 미들웨어는 이러한 중복된 정보 및 응용 시스템 계층에 불필요한 정보들을 필터링하고 요약하는 기능을 제공해야 한다.

다. 응용 시스템과의 연동

RFID 미들웨어는 정제, 요약된 태그데이터를 데이터 수요자인 기존 응용 시스템에 신뢰성 있게 전송할 수 있는 기능을 제공해야 한다. 또한, 이상의 정제, 요약 과정을 최종 인식된 RFID 태그 정보를 이용하여 태그가 부착된 개별사물에 대한 상세정보를 제공하기 위한 시스템 또한 필수적인 요소라 할 수 있으며, RFID 미들웨어 시스템의 고려 대상이 된다.

RFID 미들웨어는 RFID 정보가 다양한 응용 서비스에서 사용되도록 RFID 태그에 저장되어 있는 데이터를 적절한 장소와 적절한 시간에 응용 서비스로 전달하는 기능을 제공한다. 이때 RFID 리더기로부터 수집된 정보 중 응용 서비스가 필요한 데이터의 형식 및 전달 조건에 따라 요구되는 기능이 달라진다. 이것은 다양한 조건에 따른 필터링 기법, 처리 대상 데이터의 양과 동시에 처리되어야 하는 조건의 수 등을 고려하여 실시간 데이터 처리가 손실 없이 이루어져야 한다. 또한 응용 서비스로 정보를 제공하는 다양한 인터페이스 기능이 가능해야 한다.

2. EPC(Electronic Product Code)

EPC는 새롭게 생성되거나 이미 존재하는 모든 객체들을 유일하게 식별할 수 있도록 주어지는 객체 고유의 코드이다. 고유 일련번호를 기반으로 한 EPC 네트워크를 통해 공급체인 안에서 이동 중인 모든 개별 물품의 현 위치, 이동 경로를 추적 확인할 수 있으며, '차세대 바코드'로 불리고 있다.

전반적인 General EPC의 형태는 그림 3과 같지만, 헤더의 값에 따라 EPC 영역 식별자(Domain Identifier)의 세부 구조 및 그에 해당하는 역할이 결정되므로 각 인코딩 형태에 따라 약간의 차이가 있을 수 있다.

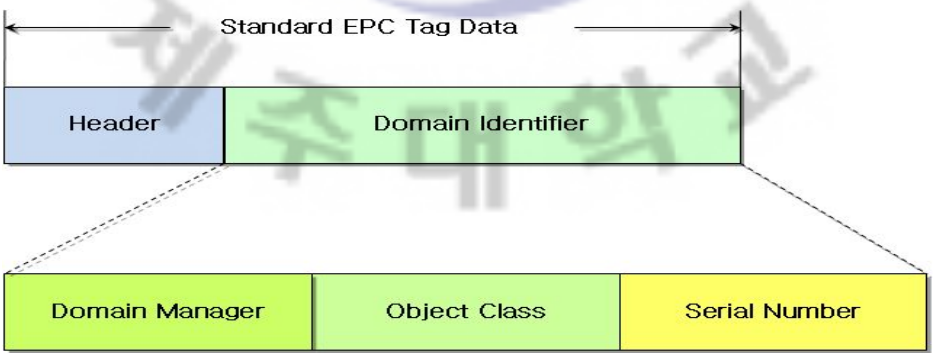

그림 3. General EPC 형태의 예

EPC 코드에는 여러 가지 종류가 있으나, 가장 널리 알려진 96비트 코드(GID, General IDentifier-96)를 기준으로 설명하면 다음과 같다. 96비트 EPC 코드는 96개의 2진수로 표시되며, 아래의 표에서와 같이 헤더→업체코드→상품코드→일련번호 순으로 구성되어 있다.

표 1. GID 코드 구조

MSB(최상위 비트) LSB(최하위 비트)

구분	Header	General Manager Number	Object Class	Serial Number
비트크기	8	28	24	36
세부내용	00110101_2 (헤더값)	268,435,455 (최대표현개수)	16 (최대표현개수)	68,719,476,735 (최대표현개수)

헤더(Header)는 EPC 코드의 종류에 따라 다른 값을 가지며, 96비트 EPC 코드의 경우 8비트로 구성되어 있고, 256개의 서로 다른 EPC 코드를 표시하기 위해 사용된다. 업체코드(Manager Number)는 EAN 바코드의 업체코드에 할당하며 각국 EAN 회원기관이 할당하며 28비트의 용량으로 7개의 숫자(0~9) 및 문자(A~F)를 조합하여 약 2억 6.000만 개의 업체 코드를 할당할 수 있다. 상품코드(Object Class)는 동일 품목의 개별 상품 코드에 해당하며 사용업체가 할당한다. 24비트 용량으로 약 1,600만개 상품에 코드를 부여할 수 있다. 일련번호(Serial Number)는 동일 품목의 개별 상품에 부여되는 고유한 식별번호로서 사용업체가 할당한다. 36비트로 이루어져 있으며 약 680억 개의 개별 상품에 코드를 부여할 수 있다.

3. PML(Physical Markup Language)

PML은 RFID 응용 시스템에서 상호간의 데이터 교환을 위해 물리적 객체의 정보를 표현하기 위한 표준 언어로, 사람과 컴퓨터가 함께 이해할 수 있도록 MIT Aout-ID Center에서 개발한 상품 기술 방식이다. 이는 물체, 시스템, 공정,

그리고 물체와 관련된 환경을 기술하는 XML 기반의 언어이다. 그림 4는 RFID 태그 데이터를 표현하는 PML 문서의 예이다.

```
<pmlcore: Sensor>
    <pmluid:ID>urn:epc:1:4.16.36</pmluid:ID>
    <pmlcore:Observation>
        <pmlcore:DateTime>2002-11-06T13:04:34-06:00</pmlcore:DateTime>
        <pmlcore:Tag>
            <pmluid:ID>urn:epc:1:2.24.400</pmluid:ID>
        </pmlcore:Tag>
        <pmlcore:Tag>
            <pmluid:ID>urn:epc:1:2.24.401</pmluid:ID>
        </pmlcore:Tag>
    </pmlcore:Observation>
</pmlcore:Sensor>
```

그림 4. PML 코드의 예

다양한 센서들로부터 얻어지는 센서 데이터 정보들을 PML이라는 공통 언어로 표현함으로써 여러 가지 업무와 응용시스템 등에 이용할 수 있다. 제품명을 비롯해 센서의 현재 상태, 데이터, 위치 등을 PML로 저장함으로써 사물의 정보를 저장, 관리하는 기능이 가능하다.

4. USN 미들웨어 동향

1) USN 미들웨어

USN 기술은 모든 사물에 컴퓨팅 능력과 통신 기능을 부여해 환경과 상황을 자동으로 인지하도록 함으로써 사용자에게 최적의 서비스를 제공할 수 있게 하는 기술이다[20]. 즉 일상생활에서 사용되는 가전기기, 사무용품등에 컴퓨팅 능력이 있는 센서들을 장착해 이 센서들의 주변의 온도, 습도 등의 환경정보와 사용자의 정보를 인식해 사용자가 필요한 서비스를 제공하도록 한다.

일반적으로 USN은 센서노드, 싱크노드, 미들웨어로 구성되어 있다. 센서 노드는 센서를 통해 주변 환경의 정보를 수집하여 가공·데이터화해 싱크노드에 전

송한다. 즉, 센서노드는 서비스의 요구에 따라, 또는 이미 설정한 조건의 이벤트 발생에 따라 센싱된 정보를 싱크노드에 전달한다. 센서노드로부터 데이터를 수집한 싱크노드는 수집한 데이터를 미들웨어를 통해 원격지의 위치한 서버나 사용자에게 전달하여 데이터를 활용할 수 있게 한다[21].

미들웨어는 다양한 하드웨어 환경을 하나의 플랫폼을 보이게 하고, 이를 이용한 단일 플랫폼에서 다양한 애플리케이션을 구동할 수 있는 환경을 지원하는 소프트웨어를 말하며, USN 미들웨어는 물리적으로 USN 응용 서비스 시스템과 센서노드 하드웨어의 중간에 위치하여 그 둘 간의 통합이 유연하게 이루어지도록 하는 역할을 수행한다.

USN 미들웨어의 기본 기능은 센서노드들로부터 수집되는 센싱 정보를 효율적으로 관리하고, USN 응용 서비스들로부터 주어지는 질의들에 대하여 신속히 응답하는 것으로 매우 단순한 것처럼 보인다. 그러나 USN 미들웨어는 센싱 정보를 관리할 때, 다수의 센서네트워크에 설치된 수많은 센서노드들로부터 연속하여 주어지는 대용량 센싱 정보를 효율적으로 관리하기 위한 방법과 연속적인 센싱 정보 요청을 센서네트워크 및 센서노드의 컴퓨팅, 통신 및 저장 능력 등을 고려하여 처리할 수 있는 방법도 제공하여야 한다. 또한, 주어지는 질의들에 대하여 실시간으로 응답할 수 있어야 하고, 유비쿼터스 환경에서 발생 가능한 일시성, 연속성, 이벤트 질의와 같이 다양한 형태의 질의도 처리할 수 있는 능력을 보유해야 한다. 이외에도 USN 미들웨어는 센서네트워크 구성에 대한 추상화 기능을 지원함으로써 이기종의 센서네트워크 구성에 독립적으로 응용 서비스가 가능하도록 해야 한다[22].

대부분의 USN 미들웨어는 응용 서비스를 지원하기 위하여 서버 시스템에 설치되고, 노드들의 원활한 동작과 성능 향상을 위하여 센서노드와 싱크노드에도 설치된다. 이러한 USN 미들웨어를 설치되는 위치에 따라서 구분하며, 서버 시스템에 설치되는 경우는 server-side 미들웨어, 그리고 노드에 설치되는 경우는 in-network 미들웨어 또는 센서노드 미들웨어라고 할 수 있다[23]. 일반적으로 server-side 미들웨어는 기본 기능으로 이기종 센서 네트워크로부터 수집한 센싱 데이터를 필터링, 통합, 분석해 의미 있는 상황정보를 추출, 저장, 관리, 검색하고 그 정보를 응용 서비스로 전달해, USN 서비스 간 연계, 통합을 용이하게 한다.

이에 반하여, in-network 미들웨어는 애플리케이션과 환경 변화에 따른 센서 노드의 재프로그래밍, 센서 네트워크의 변화 지원, 센싱 데이터의 처리, 저장, 질의 처리, 이벤트 처리 기능 등을 제공하며, 주로 센서노드와 싱크노드 영역으로 국한된다.

2) USN 미들웨어 기술 동향

USN 응용 시스템들은 사람이 직접 접근하기 어려운 지역에 서식하는 동식물들에 대한 생활환경을 감시하거나 그 지역의 기후 변화를 감지하는 u-Environment 서비스[24, 25], 교량 및 건물 등의 구조물의 안전 상태를 실시간 감지하기 위한 u-Structure 서비스[26], 병원에서 환자, 의사, 간호사 및 고가장비들의 위치를 실시간 추적하고 환자의 상태를 실시간 감지하기 위한 u-Hospital 서비스[27], 도로 및 차량에 설치된 센서노드를 이용하여 실시간으로 교통정보를 수집하여 차량 사고를 감지하고 심지어 사고를 미연에 방지할 수 있는 u-Transportation 서비스[28]를 위한 응용 시스템들이 개발되어 오고 있다.

이와 같이 다양한 분야의 USN 응용 서비스를 구축하기 위한 시스템의 개발은 USN 미들웨어 시스템의 발전에도 큰 영향을 끼치게 되었다. 실제로 USN 응용 서비스에 따라서 다양한 형태의 USN 미들웨어들이 소개되었는데, USN 응용 서비스의 QoS 요구조건에 대한 보장을 최우선 목표로 하여 개발된 MiLAN 미들웨어[29], RFID 미들웨어와 유사하게 끊임없이 획득하는 센싱 정보에 대하여 이벤트들을 설정함으로써 USN 응용 시스템에게 원하는 정보를 전송하는 이벤트 기반의 DSWare 미들웨어[30], USN 응용 서비스 변화 및 센서 네트워크 주변 환경 변화에 따라 센서노드 미들웨어의 기능을 무선 통신을 통하여 동적으로 변화시킬 수 있는 Impala 미들웨어[31], 센서 네트워크에 존재하는 센싱 정보들을 분산 데이터베이스의 분산 데이터로 간주하여 USN 응용 시스템의 요구사항을 분산 질의 처리 과정으로 수행할 수 있는 TinyDB[32] 및 Cougar[33] 미들웨어 등이 개발되어 왔다. 표 2는 지금까지 개발된 USN 미들웨어의 주요 특징 및 한계점에 대하여 간략히 제시하고 있다.

표 2. 주요 USN 미들웨어별 특징 및 한계점

USN 미들웨어	주요 특징	한계점
TinyDB (Berkeley)	· 센서 네트워크를 가상의 분산 데이터베이스로 간주 · Server와 in-network 미들웨어가 협력적으로 동작 · TinyOS 기반으로 동작, SQL-like 질의 언어 지원 · 질의 수행 최적화 및 in-network aggregation 지원	· TinyOS 기반의 센서노드에서만 이용이 가능하며, 센서노드에 신규 기능을 추가할 때, 모든 센서노드가 보유하고 있는 질의 처리기 모듈을 수정해야 함.
Cougar (Cornell)	· 모든 센싱정보를 서버에 불러온 다음, DB기반 접근 방식으로 질의를 처리함 · Server-side 미들웨어, SQL-like 질의 언어 지원 · 질의 수행 최적화 지원	· Server-side 미들웨어로서, 서버 시스템이 모든 센싱정보를 유지하고 있어야 하기 때문에 모든 센서노드들이 센싱정보를 모두 서버로 전송해야 함
SINA (Delaware)	· Cougar와 유사함. server-side 미들웨어로, SQL-like 질의 언어를 지원 · 지리적으로 인접한 센서노드들을 계층적으로 cluster로 묶어서 관리함(cluster head 노드 이용)	· Cougar와 마찬가지로 서버 시스템이 모든센싱정보를 유지하고 있어야 하기 때문에 모든 센서노드들이 센싱정보를 모두 서버로 전송해야 함
DSWare (Virginia)	· DB 방식으로 SQL-like 질의 언어를 지원 · In-network aggregation 지원 · 센서노드들에 대한 동적인 그룹 관리 방법 지원	· 특정 제품의 센서노드 하드웨어에 대한 의존적인 미들웨어로서, 이종의 센서노드들에 대한 추상적인 인터페이스 제공 기능이 부족함
Milan (Rochester)	· USN 응용 시스템의 QoS 요구 처리 기능 지원 · QoS 요구와 센서 네트워크의 리소스를 비교 분석하여 센서 네트워크의 lifetime은 최대화하면서 QoS 요구를 최대로 만족시키고자 함	· USN 응용 서비스에 tightly-coupled 되어 있어서 이종의 센서노드들에 대한 추상화를 지원하지 않음 · 모바일 센서노드를 지원하지 않음
Impala (princeton)	· 센서노드 기능의 동적 갱신 지원 · Binary 명령어를 수행할 수 있는 모바일 코드 기술을 이용하여 노드의 기능을 실행 시에 동적으로 변경	· Hewlett-Packard 제품에 의존적인 미들웨어로서, 이종의 센서노드들에 대한 추상화를 지원하지 않음
Mate (Berkeley)	· 센서노드 기능의 동적 갱신 지원(TinyOS 기반) · Byte Code와 Virtual Machine(VM) 기반의 센서노드 기능을 실행 시에 동적으로 변경할 수 있음	· 센서노드가 VM 기반으로 구성됨으로써, 복잡한 기능의 갱신일 때 interpretation 과정으로 인한 추가의 리소스 손실이 있음
COSMOS (KTRI)	· 다양한 유형의 질의 지원(일시성, 연속성, 이벤트) · 대용량 센서 네트워크 환경에 대하여 대량의 동시질의 처리 지원 · 이종의 센서 네트워크에 대한 추상화 기능 지원	· In-network aggregation과 같은 센서노드 미들웨어 기능이 부족함

이와 같이 여러 가지 분야의 USN 응용 서비스를 구축하기 위한 시스템 개발은 응용 서비스에 따라서 다양한 형태의 USN 미들웨어를 개발 및 연구되고 있지만 미들웨어에 대한 표준화 및 센서네트워크 간의 센서 데이터 공유할 수 있는 표준화는 이루어지지 않고 있다.

5. RFID 기반 미들웨어 제품 및 솔루션

현재 개발된 RFID 시스템 및 솔루션으로는 Oracle의 Sensor Edge Server, CapTech의 TagsWare, SUN의 Java System RFID Software 등이 있다.

Oracle의 Sensor Edge Server[34]는 데이터 수집, 이벤트 처리, 데이터 분배(dispatching) 등의 기능을 지원하는 센서 기반 서비스 통합 플랫폼이다. 미리 정의된 필터들을 이용하여 응용이 원하는 데이터를 추출하며, 보다 복잡한 데이터를 처리하고자 할 경우 직접 로직 필터를 작성할 수 있다.

CapTech 사의 TagsWare[35]는 RFID 태그 데이터를 응용에 전달하기 위한 링크, RFID 장비로의 표준 인터페이스를 지원하는 드라이버, 그리고 응용에서 링크와 드라이버의 사용을 지원하는 응용 기반 요소들로 구성된다. 리더 장치로부터 수집된 원시 데이터에서 태그 데이터를 추출하여 평활화(smoothing)한 후 응용에 전달하기 위한 링크 컴포넌트가 체인으로 연결된 구조를 제공한다.

SUN 사의 Java System RFID 소프트웨어[36]는 다양한 센서로부터 오는 센서 데이터 스트림을 이벤트 관리기에서 처리하고, 리더 어댑터, 필터, 로거(logger), 엔터프라이즈 게이트웨이 등으로 구성된다. 델타(delta)와 평활화 질의를 할 수 있고, 필터들을 서로 연결하여 특정 마스크(mask) 조건을 만족하는 EPC 데이터 값을 얻을 수 있을 뿐만 아니라 사용자 정의 필터를 개발할 수 있는 도구를 지원한다.

ETRI의 UbiCore 시스템[37]은 XML 기반 미들웨어 시스템으로 다양한 센서를 관리하며, XQueryStream이라는 XQuery에 기반한 연속 질의 언어를 제공한다. 필터링과 중간 결과 재사용을 통하여 스트림 데이터에 대한 질의 처리 속도를

개선하였고, 연속적으로 생성되어 들어오는 실시간 데이터뿐만 아니라 저장된 이력 데이터에 대한 질의를 지원하며 컨텍스트와 서비스의 연계 정보를 표현하기 위한 마크업 언어인 CSML(Context-driven Service Markup Language)를 제공한다.

이러한 다양한 RFID 미들웨어 및 솔루션들은 다양한 기능 및 데이터 처리 방법을 제공함으로써 데이터 처리의 효율성 면에서 뛰어나다. 하지만 대부분의 미들웨어들은 응용 서비스와 미들웨어 간 독립성을 제공하기 위한 사실상의 표준에 대한 고려가 미진할 뿐만 아니라 RFID 태그 데이터 이외에 다양한 센서로부터 들어오는 센서 데이터 스트림을 처리하지 못한다.

위에서 언급한 RFID 미들웨어이외에 본 논문에서 제안하는 목적과 같이 RFID 태그 데이터이외에 다양한 센서 데이터를 처리하기 위한 문제를 해결하기 위한 연구가 몇몇 이루어지고 있다.

Ubicore[38]의 새로운 버전에서는 다양한 센서를 관리하고, 센서에서 센싱한 데이터를 수집하고, 수집된 데이터를 처리하여 자동적으로 어플리케이션에 제공하는 방법을 제안하였다.

RFID 미들웨어 플랫폼인 REMS(RFID Event Management System)과 RBPTS(Real-time Business Process Triggering System)[39]에서는 ALE와 호환성이 있을 뿐만 아니라 수동적인 리더와 능동적인 리더 등 다양한 타입의 리더를 처리할 수 있다.

ESN(EPC Sensor Network) 아키텍처[40]는 EPCglobal 아키텍처에 기반하여 분산되어 있는 리더와 센서로부터 실시간으로 다양한 데이터를 처리하기 위해 CEP(Complex Event Processing)를 이용하여 RFID와 WSN(Wireless Sensor Network)을 통합하는 방법을 제안하였다.

하지만 이와 같은 연구들은 궁극적으로 RFID 태그 데이터만이 아닌 다양한 센서 데이터를 효율적으로 처리하기 위한 방법을 제안하고 있지만, 기존의 ALE 기반의 미들웨어에 쉽게 적용되거나 플러그인 하는 방법이 아닌 새로운 접근 방법을 제안하고 있다. 본 논문에서는 위의 연구들과는 다르게 기존의 ALE 기반의 미들웨어를 수정 없이 다양한 센서 데이터를 RFID 태그 데이터와 같이 처리 가능하도록 하는 방법을 제안한다.

6. 센서 데이터 처리를 위한 고려사항

본 논문에서는 RFID 미들웨어에서 센서 데이터를 처리하기 위해 다음과 같은 사항을 고려하여야 한다.

RFID 태그 데이터와는 달리 센서들은 현재 데이터 표현을 위한 표준화가 마련되어 있지 않다. 일반적인 센서들은 제작사에 의해 센서가 수집하기 위한 환경 정보에 따라 각각 주변 환경 특성에 따라 설계되고 제작되어 진다. 이에 따라 일반적인 데이터 패킷 구조는 각 센서마다 데이터의 크기부터 다양한 구조를 갖고 있다. 또한 센서 데이터는 주위 환경 정보를 센싱하여 그 결과를 보내는 경우와, 센서 자체의 상태 정보를 보내는 두 가지 타입의 데이터가 있다. 이와 같은 부분을 고려하여 본 논문에서는 다양한 패킷 구조를 갖는 센서 데이터를 처리하기 위해 공통의 패킷 구조를 갖는 프로토콜을 설계할 필요가 있다.

III. 다양한 센서 데이터 스트림 처리 방법

1. 개요

본 연구에서는 ALE 기반 RFID 미들웨어에서 RFID 태그 데이터 이외에 다양한 센서 데이터 스트림 처리를 위한 방법을 제안한다. RFID 리더기를 하나의 센서로 간주하여 센서가 센싱한 정보를 RFID 미들웨어가 태그 데이터를 처리하는 과정과 같이 센서 데이터를 처리한다. 미들웨어로 입력되는 일반 센서 데이터를 ALE 미들웨어 내부에서 사용하는 EPC 포맷으로 자동 변환함으로써 미들웨어 수정 없이 일반 센서 데이터를 처리하기 위한 방법이다.

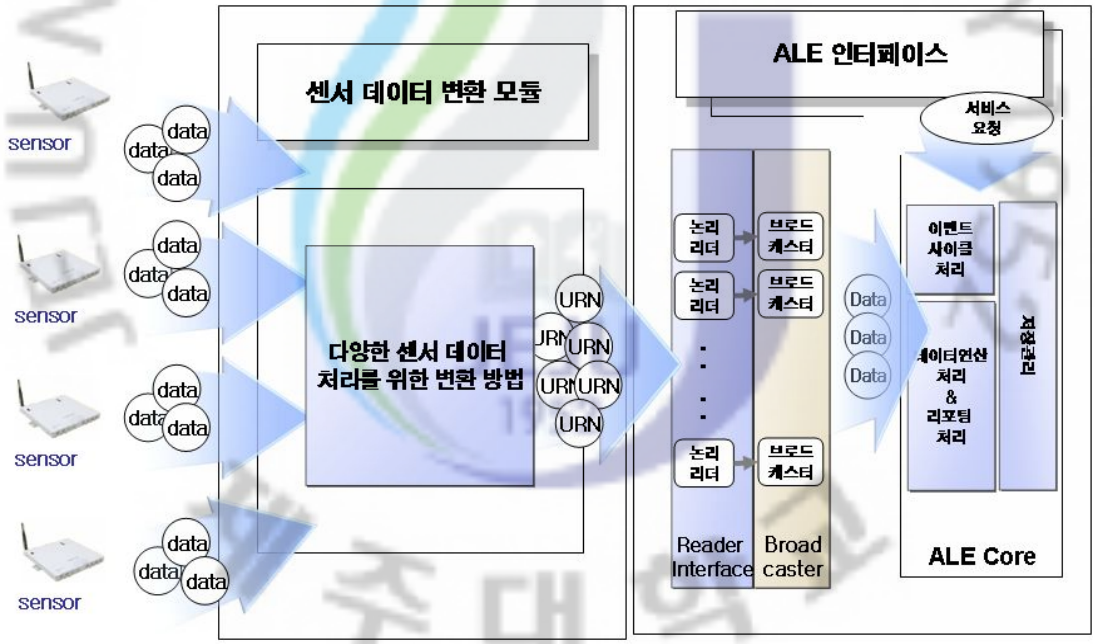

그림 5. 센서 데이터 처리 개념도

그림 5는 ALE 기반 RFID 미들웨어에서 센서 데이터 처리를 위한 방법을 나타내는 개념도 이다. 그림과 같이 센서에서 수집된 데이터들은 미들웨어로 전송되면 센서 데이터를 미들웨어 내에서 처리가 가능한 데이터로 변환을 거쳐 기존

의 미들웨어에서 처리가 가능하게 한다.

ALE 기반 RFID 미들웨어에서의 태그 데이터 처리과정은 리더기를 통해 미들웨어로 전송된 태그 데이터를 EPCglobal에서 정의한 미들웨어 내부적으로 사용하는 PML로 변환하여 처리한다. 태그 데이터를 PML로 변환하는 과정에서는 태그 데이터를 EPC 코드 정의에 따라 URN 형태로 변환을 하여 처리한다. 즉 16진수로 표현된 태그 데이터를 코드 체계에 정의되어진 방법에 따라 URN 형태의 코드로 변환한다. 예를 들어 리더기를 통해 읽혀진 태그 정보가 '8000 0000 4001 0000'일 경우 미들웨어는 데이터를 2진수로 변환하고, 데이터에서 헤더 정보를 추출하고, 헤더에 따른 코드 변환 규칙에 따라 URN 형태인 'urn:epc:tag:sgtin-64:0.0.32.65536'으로 변환하여 처리한다. 따라서 일반 센서 데이터의 경우도 이와 유사하게 센서 데이터를 각각의 센서에 따라 EPC 형태의 코드와 URN 코드를 정의하고, 해당 URN으로 변환하면 기존의 ALE 미들웨어를 수정하지 않고서도 센서 데이터를 처리할 수 있다.

2. 시스템 구성

USN 환경을 위한 센서이외에 일반 센서들은 주변 환경 정보를 수집하기 위한 기능들만을 특화하여 제작되어 있는 경우가 대부분이다. 그래서 시스템을 구성하는데 있어서 일반 센서에 통신 기능을 추가하기 위해 RF모뎀을 연결하여 미들웨어와 통신이 가능하도록 시스템을 구성하였다.

일반 센서 데이터를 ALE 미들웨어서 처리하기 위한 시스템의 구성은 크게 세 부분으로 구성 된다. 주위 환경 정보를 수집하는 센서, 센서와 미들웨어간의 통신을 위한 RF모뎀, 그리고 미들웨어로 구성된다.

센서는 주위 환경 정보를 수집하여 센서와 연결된 RF모뎀(센서노드)을 통해 수집한 데이터를 전송한다. 미들웨어는 센서와 통신을 위한 RF모뎀(마스터노드)과 연결되고 이를 통해 센서노드에서 보내는 데이터가 미들웨어로 입력된다.

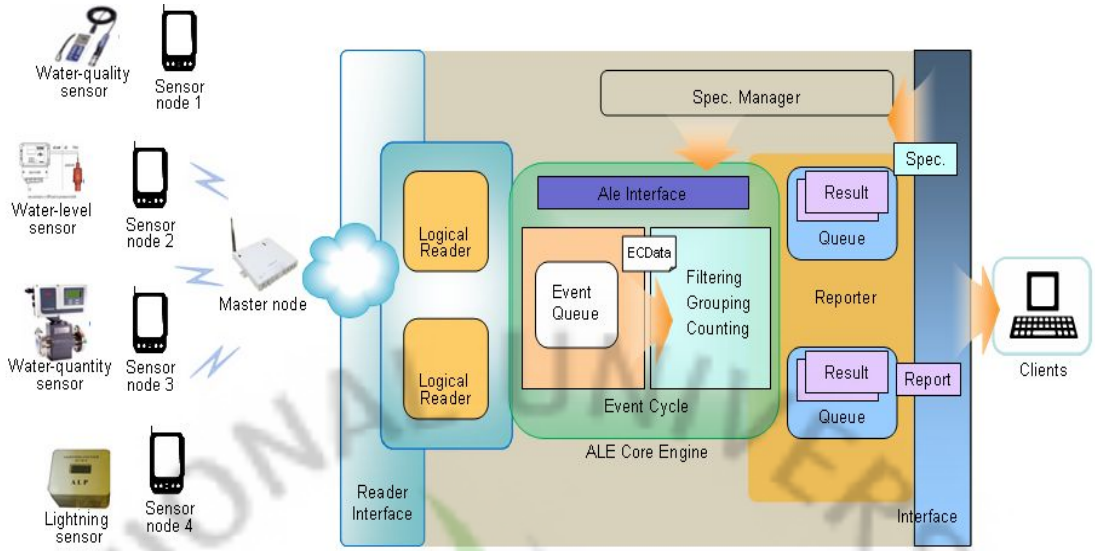

그림 6. 시스템 구성도

본 논문에서는 정수장 및 지하수 관리를 위해 사용되는 수질 센서, 수위 센서 및 유량계 센서 그리고 낙뢰 경보 센서를 이용하여 시스템을 구성하였다. 각각의 센서들은 주위 환경 정보를 센싱하여 RF모뎀을 이용하여 미들웨어로 전송하고 미들웨어는 입력되는 센서 데이터를 EPC 데이터화하여 RFID 태그 데이터와 같은 방법으로 처리한다.

3. 센서 유형 및 분석

센서 데이터는 RFID 태그 데이터와는 달리 센서 데이터를 표현하기 위한 표준화가 마련되지 있지 않고, 센서 제조사별 또는 응용에 따라 다양한 구조의 데이터 프로토콜을 가지고 있다. 다음 표 3에서와 같이 사용되어지고 있는 센서들도 각 센서별 수집하려는 정보에 맞게 제작되어져 있고 데이터 크기 및 구조 또한 다양하다.

표 3. 센서 하드웨어 스펙

구분	제조사	모델	설명
수질 센서	A사	Hach sc1000 Multi-parameter Universal Controller	전기전도도, 온도, 잔류염소, Ph, 탁도 측정값을 반환
수위 센서	B사	PS7000	수위 측정값을 반환
유량계 센서	C사	Win TEC WTM100	순간유량, 누적유량, 순간유속, 적산시간 측정값을 반환
낙뢰 경보 센서	D사	CORONARM-40	낙뢰 경보 값을 반환

수질센서의 경우는 수질정보를 수집하기 위해 여러 가지의 센서로 복합적으로 구성되어 있다. 물의 흐린 정도를 나타내는 탁도 센서, 수소 이온에 감응하는 Ph 센서, 잔류염소를 측정하는 센서, 온도를 측정하는 온도 센서, 그리고 전기전도도를 측정하는 전기전도도 센서 등으로 구성되어져 있다. 표 4는 수질센서의 데이터 구조를 나타낸다.

표 4. 수질 센서 데이터 구조

data		
코드	크기(byte)	의미
0x0A	1	전기전도도 항목
0x값1	4	전기전도도 측정값
0x0B	1	온도 항목
0x값1	4	온도 측정값
0x0C	1	잔류염소 항목
0x값1	4	잔류염소 측정값
0x0D	1	Ph 항목
0x값1	4	Ph 측정값
0x0E	1	탁도 항목
0x값1	4	탁도 측정값
0x0F	1	센서별 상태코드
0x값1	2	전기전도도 & 온도 센서 상태
0x값1	2	잔류염소 & Ph 센서 상태
0x값1	2	탁도 센서 상태
0xCheckSum	1	체크섬
	33(byte)	

수위 측정의 방법은 직접적, 간접적 방법으로 나눌 수 있고 전자에는 직접 관측법, 플로트에 의한 방법이 있으며, 후자에는 압력 측정을 이용하는 방법, 음향을 이용하는 방법 기타 각종 물리 현상을 이용하는 방법이 있다. 본 논문에서 사용한 수위 센서의 수위 데이터의 구조는 표 5와 같다.

표 5. 수위 센서 데이터 구조

data		
코드	크기(byte)	의미
0x0A	1	수위 항목
0x값1	4	수위 측정값
0xCheckSum	1	체크섬
	6(byte)	

유량계 센서는 유량정보를 수집, 측정하기 위해 유체가 흐르는 배관에 Faraday 법칙을 이용하여 자기장을 형성하고 그 자장에 유체가 흐르면 기전력이 발생한다는 Flaming의 오른손 법칙의 원리를 이용하여 만든 유량 계측기인 전자 유량계를 사용한다. 전자 유량계는 온도, 밀도, 압력, 점도, 고형물의 유무와 관계없이 전도도가 5㎲/㎝ 이상이면 유량 측정을 할 수 있기 때문에 상수, 하수, 오수, 폐수, 화학, 정유, 제철, 제지 등 유체 유량을 정밀하게 관리하여야 하는 분야에서 넓게 사용되고 있다. 전자유량계 센서 데이터는 표 6과 같은 구조를 갖는다.

표 6. 유량계 센서 데이터 구조

data		
코드	크기(byte)	의미
0x0A	1	순간유량 항목
0x값1	4	순간유량 측정값
0x0B	1	누적유량 항목
0x값1	4	누적유량 측정값
0x0C	1	순간유속 항목
0x값1	4	순간유속 측정값
0x0D	1	적산시간 항목
0x값1	4	적산시간 측정값
0xCheckSum	1	체크섬
	28(byte)	

낙뢰 경보 센서는 뇌운의 전자계 충격파의 빈도수와 강도를 측정하여 뇌운의 접근 여부를 감지하고 근접한 뇌운에 의한 전계와 뇌방전에 의해 야기된 전계변화를 측정하고 방전전류를 측정하여 뇌운의 이동 및 진행 양상을 정확히 감지한다. 감지한 정보를 바탕으로 거리에 따라 정보를 표현한다.

표 7. 낙뢰 경보 센서 데이터

단계	경보 내용
제1주의보	20~30km의 거리에 뇌운이 발생, 약 30분후, 뇌우 가능성 있음
제2주의보	뇌운이 10km이내에 근접하고, 약 10분후 근방에 낙뢰 가능성 있음
낙뢰경보	인접지역에 낙뢰가 발생할 확률이 높음, 긴급피난의 필요가 있음

낙뢰 경보 센서 데이터의 경우는 표 7과 같이 낙뢰 경보 센서 감지한 정보에 따른 3가지의 경우로 나뉘어 제1주의보, 제2주의보, 낙뢰 경보 등으로 표현한다.

4. 센서 데이터 전송을 위한 프로토콜 설계

RFID 태그 데이터와는 달리 센서 데이터는 데이터 프로토콜에 대한 표준화가 마련되어 있지 않으며, 일반적인 센서의 경우는 제조사에 의해 수집하기 위한 환경 정보에 따라 각각 주변 환경 특성에 따라 설계되고 제작되어 진다. 그래서 센서의 데이터 패킷 구조는 각 센서마다 데이터의 크기부터 다양한 구조를 갖고 있다. 또한 센서 데이터는 주위 환경 정보를 센싱하여 그 결과를 보내는 경우와, 센서 자체의 상태 정보를 보내는 두 가지 타입의 데이터가 있다. 이와 같은 부분을 고려하여 본 논문에서는 다양한 패킷 구조를 갖는 센서 데이터를 처리하기 위해 공통의 패킷 구조를 갖는 프로토콜을 설계 하였다.

본 논문에서 제안하는 프로토콜은 다양한 센서 데이터 처리를 위해 프로토콜의 전반적인 구성은 미들웨어와 센서간의 통신을 위하여 필요한 부분을 주안점으로 설계하였다. 기존의 센서 데이터의 구조를 변경하지 않고 다양한 센서를 활용할 수 있게 확장성 있게 시스템 구성에 있어서 필요로 하는 부분만을 고려하여 설계하였다. 미들웨어로 입력되는 센서 데이터에서 센서를 판별하고 센서에 따라 데이터 파싱 및 처리를 하기 위해 시스템 구성에 있어서 센서를 판별하기 위한 정보를 프로토콜 구성에 포함시켰다. 제안하는 프로토콜의 구조는 그림 7과

같이 크게 헤더, 메인프레임, 테일로 구성된다.

헤더의 요소로는 시스템 구성에 있어서 센서와 미들웨어간의 통신을 위하여 데이터를 보내는 센서노드와 미들웨어와 연결된 마스터노드를 나타낸다. 메인 프레임은 센서에서 센싱하여 전송한 데이터를 변경 없이 사용한다. 테일은 데이터의 마지막을 뜻한다.

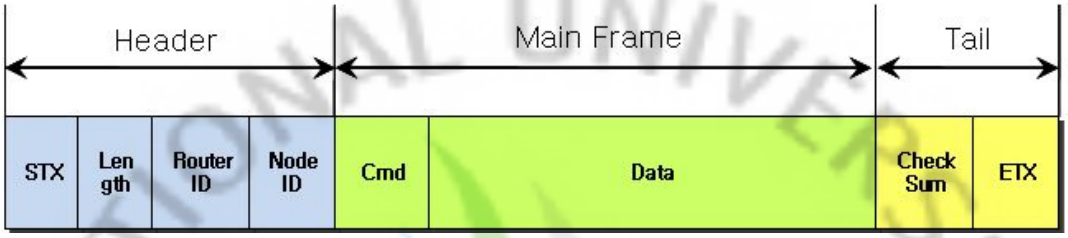

그림 7. 센서 데이터 프로토콜

프로토콜의 세부적인 구성요소는 다음과 같다. 헤더의 구성요소는 표 8과 같이 데이터 패킷의 시작을 나타내는 STX, 데이터 길이를 나타내는 Length, 데이터 전송되는 목적지의 마스터 노드를 나타내는 Router ID, 데이터 보내는 센서를 나타내는 Node ID로 구성한다.

표 8. 헤더의 구성요소

Field Define	Bytes (Sum)	Description or Value	Unit
STX	1	Ox02(start of Frame)	Hex
Length	1	Router ~ CheckSum 까지의 길이	Hex
Router ID	1	Router ID(Ox00 ~ OxFF)	Hex
Node ID	1	Node ID(Ox00 ~ OxFF)	Hex

메인 프레임의 구성은 표 9와 같이 센서에서 전송되어지는 데이터가 주위 환경 정보를 전송하였는지 그렇지 않으면 센서의 상태 정보를 전송하였는가를 구

분하기 위한 요소와 실질적으로 센서에서 보내어지는 데이터로 이루어진다.

표 9. 메인프레임의 구성요소

Field Define	Bytes (Sum)	Description or Value	Unit
Command	1	'D'	char
Data	가변	각각의 센서에서 수집한 low data	Hex

마지막으로 테일은 표 10과 같이 데이터의 에러 체크를 위한 요소와 데이터 패킷의 마지막을 나타내는 ETX로 구성된다.

표 10. 테일의 구성요소

Field Define	Bytes (Sum)	Description or Value	Unit
CheckSum	1	Router ~ Data까지의 합	Hex
ETX	1	프레임의 종료 문자(Ox03)	Hex

센서에 의해 수집된 데이터는 센서노드를 통해 마스터 노드로 전송되고, 마스터노드는 센서노드에서 보내진 데이터를 제안하는 프로토콜 구조에 맞게 미들웨어로 전송한다. 시스템을 구성하는데 있어서 각각의 센서노드는 센서의 종류를 판별하기 위해 센서노드의 ID를 지정한다. 수질 센서노드인 경우 '01 ', 수위 센서노드는 '02', 유량계 센서는 '03', 그리고 낙뢰 경보 센서는 '04'로 지정하고 마스터노드 또한 센서노드에서 전송하려는 마스터노드를 구분하기 위하여 ID를 지정한다.

수위 데이터이의 경우, 센서노드 ID가 '02'이고 전송하려는 마스터노드의 ID가 '0A'라면, 미들웨어로 전송 되어지는 데이터의 구조는 헤더에는 프레임의 시작을 나타내는 STX는 '02', 마스터노드 ID로부터 체크섬까지의 길이, 라우터 ID(마스터노드)인 '0A', 센서노드 ID(센서노드)인 '02'로 이루어진다.

메인 프레임은 수집한 데이터를 전송 한다는 명령어를 나타내는 '44', 수집된 데이터 '1002 8384 0100 0101 0000 0058 6FBD 3FF9 0215 D000 0000 0000 0000 0000 0000 0000 00BE 8B10'등과 같이 이루어진다. 마지막 테일은 체크 섬인 '036B'와 프레임의 끝을 나타내는 EOF인 '03'으로 이루어진다. 즉, 미들웨어로 전송되어지는 전체 수위 데이터의 구조는 '0229 0A02 4410 0283 8401 0001 0100 0000 586F BD3F F902 15D0 0000 0000 0000 0000 0000 0000 0000 BE8B 1003 6B03'가 된다.

제안하는 프로토콜은 기존의 센서 데이터의 구조를 변경하지 않고 시스템 구성에 있어서의 필요한 정보만을 추가하여 프로토콜을 설계하였다. 다양한 환경에서 사용하기 위해 시스템 구성 정보만을 추가하게 됨으로써 처리가 가능하게 함으로써 제안하는 프로토콜이 아닌 다른 프로토콜을 사용하게 되더라도 센서와 미들웨어간의 인터페이스에서 센서노드 정보를 추가만 하면 처리가 가능하도록 하였다.

5. 센서 데이터 처리 방법

본 논문에서는 ALE 미들웨어에서 RFID 태그 데이터이외에 다양한 센서 데이터 처리를 위해 태그 정보를 읽어 미들웨어로 태그 데이터를 전송하는 리더기를 하나의 센서로 간주하여 센서에서 주위 환경 정보를 수집하여 미들웨어로 전송하는 데이터를 RFID 태그 데이터처럼 처리하는 방법을 제안한다.

ALE 미들웨어에서 리더에서 입력받은 태그 데이터를 처리하는 과정은 다음과 같다. ALE 기반 RFID 미들웨어에서 처리되는 태그 데이터는 EPCglobal에서 정의한 EPC 코드 데이터이다. 미들웨어로 입력되는 태그 데이터는 PML로 변환하여 내부적으로 처리하는데 PML로 변환과정에서 그림 8과 같이 태그 데이터는 URN 코드 형태로 변환한다.

그림 8. 태그 데이터의 URN 코드 변환 결과

URN 코드로 변환된 태그 데이터가 갖는 의미는 EPC 코드 중 SGTIN-64 코드, 즉 개개의 물체를 식별하기 위한 코드를 뜻한다. 식별 데이터는 4개의 필도로 구성되며 각각의 필드는 '.'를 사용하여 구분한다. 첫 번째 필드인 '0'은 물류·유통을 의미하는 필터, 두 번째 필드인 '0'은 각 업체마다 부여되는 업체코드를 의미하며, 세 번째 필드인 '32'는 상품 분류에 따른 상품코드를 뜻하며, 마지막 '65536'은 각각의 상품마다 부여하는 일련번호를 의미한다. 이와 같이 URN 코드 형태로 변환된 태그 데이터를 이용하여 업체코드, 상품코드, 일련번호 등 각각의 항목에 따라 사용자 요청에 따라 필터링 및 그룹핑 과정을 거쳐 응용 애플리케이션에 제공한다.

센서 데이터의 경우도 RFID 태그 데이터가 처리되어지는 방법과 유사하게 센서 데이터를 PML로 변환하게 되면 태그 데이터와 같이 기존의 미들웨어 수정 없이 처리가 가능하도록 할 수 있다. 즉, ALE 기반 RFID 미들웨어에서 RFID 태그 정보이외에 다양한 센서 데이터 처리를 위한 방법으로 각각의 센서 데이터마다 EPC 코드와 유사한 센서 데이터에 맞는 코드를 정의하고 센서 데이터를 URN 코드 형태로 변환하여 PML로 변환하면 기존의 ALE 미들웨어에서도 처리가 가능해 진다.

센서 데이터의 변환 과정은 그림 9와 같은 과정을 거쳐 변환한다. 미들웨어로 전송된 센서 데이터를 EPC 코드 형태로 변화하고, EPC 코드 형태로 변환된 데이터는 헤더에 정보에 따라 센서의 URN 코드로 변환한다.

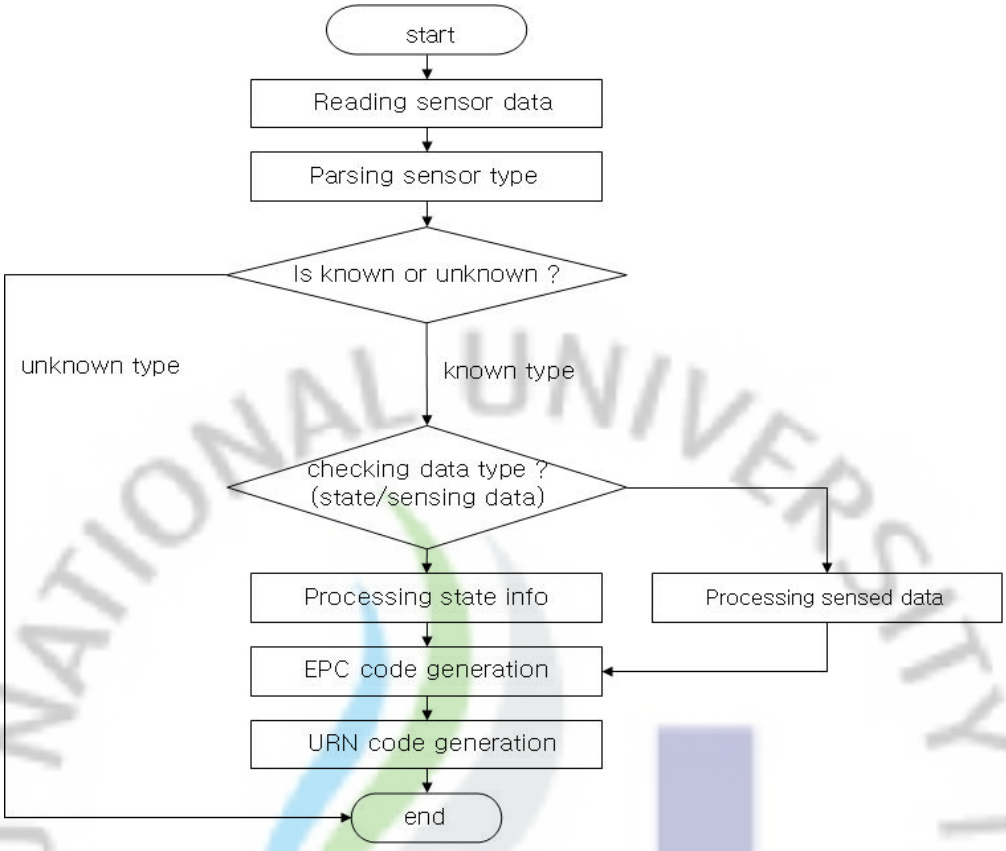

그림 9. 데이터 변환 과정

우선, 미들웨어로 전송된 데이터에서 센서의 종류를 판별하고, 센서의 종류가 미들웨어에 등록된 센서인지를 판별하여 등록되어 있는 센서의 경우 처리가 계속 이루어지지만 등록되어 있지 않은 센서의 경우 과정을 종료한다. 센서의 종류가 판별된 데이터는 센서에서 보내어진 데이터가 센서의 상태 정보인지, 센서가 수집한 정보를 나타내는지를 판별한다. 센서의 상태를 나타내는 경우에는 각각의 센서 상태정보를 나타내는 형태로, 센서가 수집한 정보의 경우는 각각의 센서 수집 정보를 나타내는 EPC 태그 데이터와 유사한 코드 체계로 변환한다.

센서 데이터를 EPC 코드 형태로 변환하는 방법은 다음과 같다. 그림 10과 같이 센서 데이터는 센서별 코드 체계에 따라 센서 타입 정보를 의미하는 헤더와 실제 센서가 전송한 데이터를 저장하는 데이터로 구성된다.

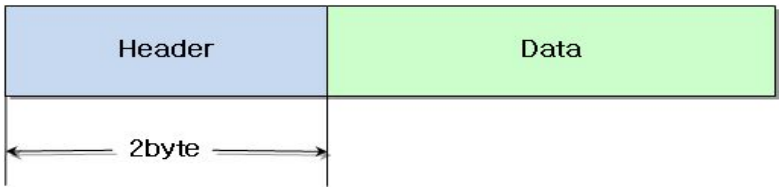

그림 10. 센서 데이터를 위한 EPC 코드 구조

헤더는 각각의 센서의 종류와 센서 데이터 변환 과정에 있어서 필요한 메타 정보를 갖는다. 데이터 부분은 센서에서 주위 환경 정보를 센싱한 결과인 데이터로 구성하고, 각각의 센서마다 수집되는 데이터 항목에 따라 데이터의 크기가 달라진다. 표 11은 본 논문에서 임의로 정의한 센서별 헤더이다.

표 11. 센서별 헤더 정보

센서 종류	헤더 정보
수질 센서	3600 0001
수위 센서	3600 0002
유량계 센서	3600 0003
낙뢰 경보 센서	3600 0004

EPC 코드 데이터의 경우, 헤더의 값에 따라 도메인 식별자의 세부 구조 및 그에 해당하는 역할이 결정되어 각 인코딩 형태에 따라 차이가 있다. 센서의 경우도 EPC 코드와 마찬가지로 헤더 값에 따라 센서 데이터의 세부 구조 및 인코딩 형태가 달라진다.

수질 센서의 경우 센서에서 보내어지는 데이터는 전기전도도, 온도, 잔류염소, Ph, 탁도 등 여러 항목으로 구성되어 있고, 수위 센서의 경우는 오직 수위 값만을 갖는 하나의 항목으로 이루어져 있다. 이와 같이 센서 데이터도 각각의 헤더 값에 따라 데이터의 세부 구조와 인코딩 형태가 달라짐으로써 데이터 변환 과정에 있어서 차이가 발생한다.

센서 데이터를 EPC 코드 형태로 변환한 후, 다음 과정은 URN 코드 형태로 변환하는 것이다. 센서 데이터의 URN 코드 형태는 EPC 코드의 URN 코드와 유사한 형태를 갖는다.

EPC 코드 중 SGTIN-64인 경우 'urn:epc:tag:sgtin-64:0.0.32.65536'와 같은 형태와 같다. 코드 형태는 크게 코드 정보를 의미하는 부분과 태그가 부착된 개체 정보를 나타내는 부분으로 구분되어 있다. 코드 정보에서는 URN을 의미하는 'urn', EPC 코드를 의미하는 'epc', 태그 데이터를 의미하는 'tag', 그리고 코드 종류를 의미하는 'sgtin-64'로 이루어져 있다. 개체 정보에서는 필터, 업체코드, 상품코드, 일련번호 순으로 '.'로 구분하여 이루어져 있다.

본 논문에서 제안하는 센서 데이터의 URN 형태는 그림 11과 같이 EPC 코드의 URN 형태와 유사한 형태로 정의한다.

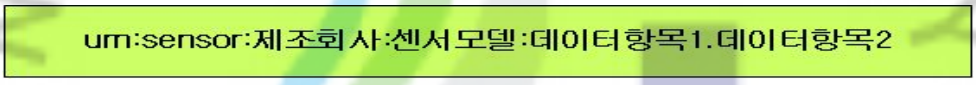

그림 11. 제안하는 URN 코드 체계

코드 정보에서는 URN을 의미하는 'urn', 센서를 의미하는 'sensor', 디바이스인 센서의 제조회사를 가리키는 '제조회사', 그리고 센서 모델명을 나타내는 '센서모델'로 구성한다. EPC코드에서 개체 정보를 의미하는 부분은 실제 센서에서 센싱한 결과인 데이터를 각 센서마다 갖는 항목에 따라 '.'로 구분하여 구성한다.

다음 표 12는 본 논문에서 정의한 센서 데이터의 URN 코드이다.

표 12. 제안하는 URN 코드 체계

센서 종류	URN 체계
수질 센서	urn:sensor:a:sc1000:전기전도도.온도.잔류염소.Ph.탁도
수위 센서	urn:sensor:b:ps700:수위
유량계 센서	urn:sensor:c:wtm100:순간유량.누적유량.순간유속.적산시간
낙뢰 경보 센서	urn:senosor:d:coronarm-40:낙뢰경보

예를 들어 유량계 센서인 경우, 'C'사에서 제작한 'wtm100' 모델의 유량계 센서에서 순간유량 측정값, 누적유량 측정값, 순간유속 측정값, 적산시간 측정값 등을 URN 코드 형태로 변환하면 'urn:sensor:c:wtm200:순간유량측정값.누적유량측정값.순간유속측정값.적산시간측정값'으로 변환하게 된다. 실제 유량 센서의 데이터 변환 과정을 보면 그림 12와 같은 과정을 거쳐 변환 과정이 이루어진다.

그림 12. 센서 데이터 변환 과정

유량계 센서 데이터의 변환 과정은 미들웨어 입력된 원시 데이터에서 센서노드 부분을 통해 센서의 종류를 판별하여 센서 종류에 따른 헤더 값을 할당한다. 유량계 센서에 할당된 헤더와 원시 데이터에서 실제 센서 데이터 부분을 파싱하여 EPC 코드 형태로 변환한다. EPC 코드 형태로 변환된 데이터는 헤더 정보에 따라 URN 코드 변환 규칙에 의해 URN 코드로 변환하게 된다.

이와 같이 센서 데이터를 URN 코드로 변환 가능하게 함으로써 기존의 ALE 미들웨어에서 처리가 가능 할뿐만 아니라 각 데이터 항목별로 EPC 코드 처리와 같이 필터링, 그룹핑 등 ALE에서 정의한 기능들을 사용할 수 있고, 사용자 요청에 따른 데이터를 제공할 수 있다. 또한 URN 코드 내에서 센서의 제조회사 및 센서 타입, 수집된 센서의 정보들을 쉽게 알아볼 수 있고 효과적으로 관리할 수 있다.

IV. 구현 및 실험

1. 개요

본 논문에서는 기존의 ALE 미들웨어를 수정 하지 않고 센서 데이터를 처리하기 위하여 미들웨어 내부적으로 처리할 수 있는 데이터 형식인 EPC 포맷에 맞게 센서 데이터를 변환한다. 이와 같은 방법으로 센서 데이터를 EPC 데이터화하면 미들웨어 내부적으로 수정 없이 센서 데이터 처리가 가능하다. 이와 같은 방법을 위하여 센서 데이터를 EPC 데이터화하는 센서 데이터 변환 모듈을 구현한다. 또한 기존의 개발되어진 미들웨어를 수정하지 않고 플러그인하여 사용할 수 있게 구현하며, RFID 태그 데이터 처리를 위한 모듈과 센서 데이터 처리를 위한 모듈로 구성하여 각각의 모듈을 실행함으로써 RFID 태그 데이터와 센서 데이터 처리를 가능하게 한다. 그림 13은 센서 데이터 변환을 위한 모듈의 구성도이다.

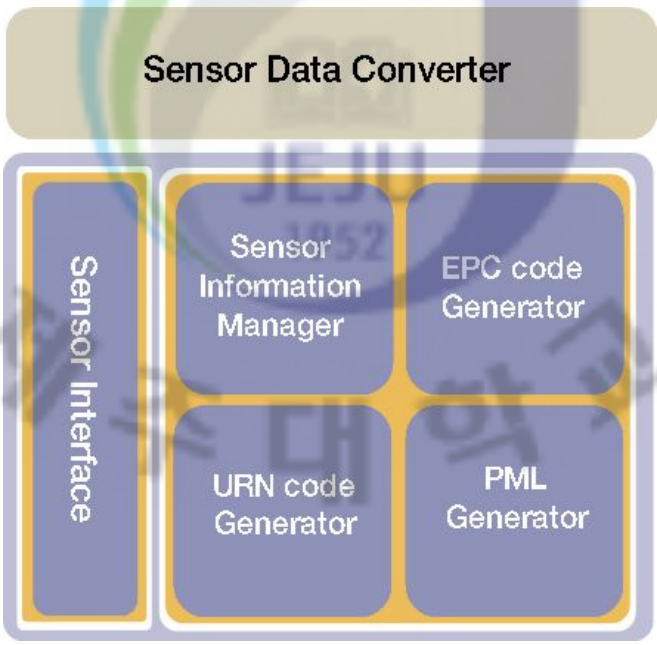

그림 13. 센서 데이터 변환 모듈 구성도

센서 데이터 변환을 위한 구성은 크게 두 부분으로 센서 인터페이스와 센서 데이터 변환부 등으로 나뉜다. 센서 인터페이스(Sensor Interface)는 센서와 미들웨어간의 통신을 위해 센서에서 수집한 데이터를 미들웨어로 전송하기 위한 센서와 미들웨어간의 인터페이스이다. 센서 데이터 변환부는 센서 데이터를 기존의 ALE 미들웨어에서 처리가 가능한 데이터로 변환하는 역할을 한다. 센서 데이터 변환을 위해 센서 데이터 변환부에는 각각의 센서 데이터 변환 정보인 메타정보를 관리하는 센서 정보 관리자(Sensor Information Manager)와 센서 데이터를 EPC 코드와 유사한 형태로 변환하는 EPC 코드 생성기(EPC code Generator), EPC 코드와 유사한 형태로 변환된 데이터를 URN 코드로 변환하는 URN 코드 생성기(URN code Generator) 그리고, 마지막으로 미들웨어 내에서 처리되어 지는 데이터인 PML로 변환하는 PML 코드 생성기(PML Generator)로 구성된다.

센서 데이터의 처리 과정은 센서 정보 관리자에 등록된 센서 변환 정보에 따라 EPC 코드 생성기에 의해 EPC 코드 형태의 센서 데이터로 변환하고, EPC 코드 형태로 변환된 데이터를 URN 코드 생성기에 의해 URN 형태로 변환한 후, 최종적으로 PML 코드 생성기에 의해 ALE 미들웨어에서 처리되는 데이터 형태인 PML로 변환한다.

2. 주요 클래스 다이어그램

센서 데이터 변환을 위한 모듈을 구현한 패키지 다이어그램은 그림 14와 같다. 센서 데이터 변환을 위한 패키로는 센서와 미들웨어간의 인터페이스를 구현한 sensor.inter 패키지와 센서 데이터 변환을 위한 sensor.convert 패키지가 있다.

sensor.convert 패키지는 센서 데이터 변환 처리에 따라 EPC 코드 형태로 변환하는 sensor.covert.epc 패키지와 URN 형식으로 변환하는 sensor.convert.urn 패키지, 그리고 PML로 변환하는 sensor.convert.pml 패키지로 구성된다. 그리고 센서 데이터 변환 정보를 관리하는 sensor.im 패키지와 센서 데이터 변환에 사용되는 유틸들로 구성된 sensor.util 패키지로 이루어져 있다.

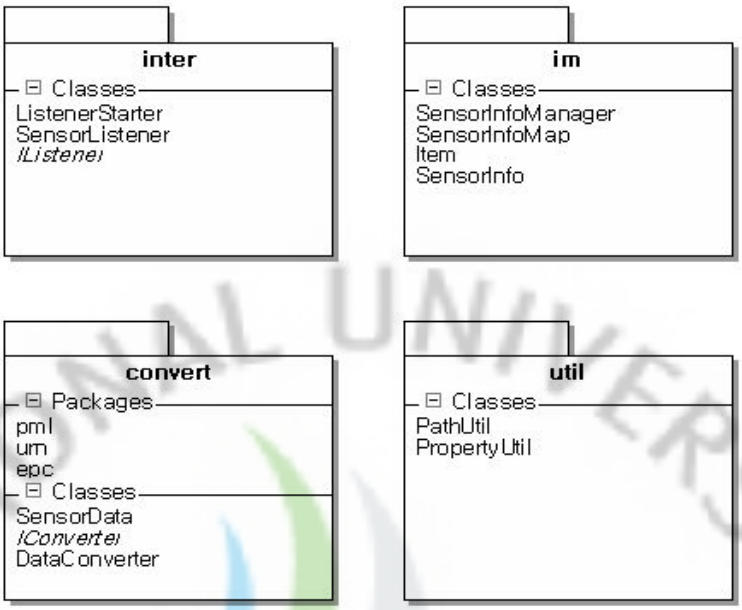

그림 14. 패키지 다이어그램

표 13. 센서 데이터 변환 모듈의 주요 패키지

패키지	설명
sensor.inter	센서와 미들웨어간의 인터페이스를 구현한 패키지
sensor.im	센서 데이터 변환에 필요한 센서 메타정보를 관리하는 클래스를 구현한 패키지
sensor.convert	센서 데이터 변환을 실행하는 클래스를 구현한 패키지
sensor.convert.pml	센서 데이터를 PML로 변환하는 클래스를 구현한 패키지
sensor.convert.epc	센서 데이터를 EPC 형태로 변환하는 클래스를 구현한 패키지
sensor.convert.urn	센서 데이터를 URN 형태로 변환하는 클래스를 구현한 패키지
sensor.util	센서 데이터 변환에 필요한 주요 유틸들을 구현한 패키지

1) inter 패키지

inter 패지지의 주요 클래스들은 센서와 미들웨어간의 인터페이스를 구현한 클래스이다. ListenerStarter 클래스는 미들웨어에 등록된 센서에서 데이터를 전송받기 위한 리스너를 실행하는 클래스이다. IListener 인터페이스는 센서, 리더기 등 미들웨어와의 통신을 위한 인터페이스이며, SensorListener는 센서에서 보내어지는 데이터를 전송받기 위해 IListener 인터페이스를 구현한 클래스이다.

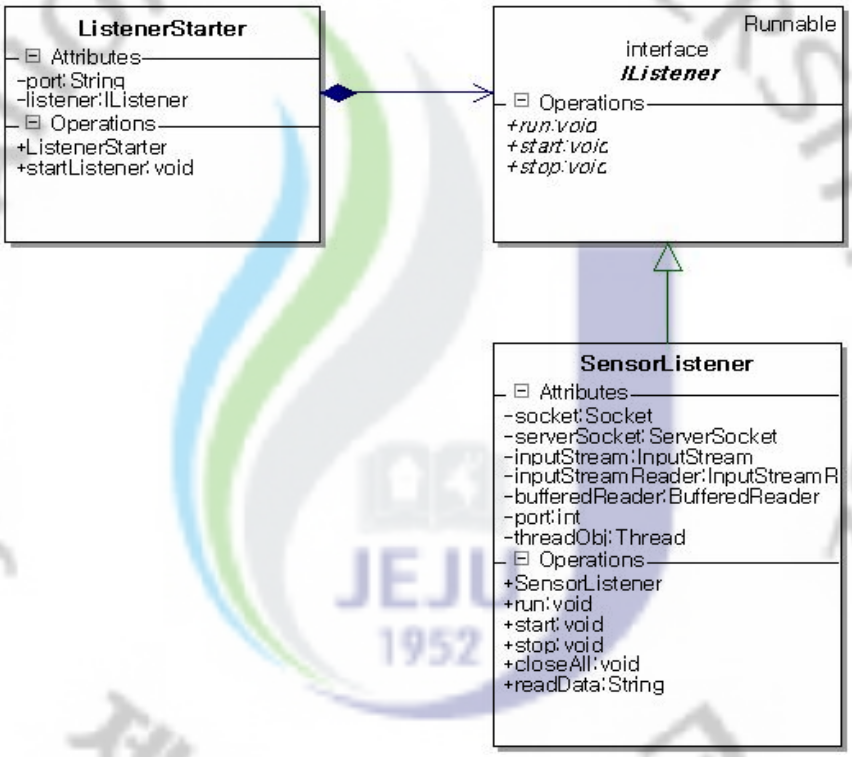

그림 15. inter 패키지의 클래스 다이어그램

표 14. inter 패키지

클래스	설명
ListenerStarter	등록된 센서와 미들웨어간의 인터페이스를 실행하는 클래스
IListener	센서, 리더기 등 미들웨어간의 통신을 위한 인터페이스
SensorListener	센서와 미들웨어간의 통신을 위해 IListener 인터페이스를 구현한 클래스

2) convert 패키지

convert 패키지의 주요 클래스들은 센서 데이터를 미들웨어에서 처리가 가능한 데이터 형태로 변환을 하는 클래스들로 구성되어 있다. 센서 리스너를 통해 미들웨어로 전송된 데이터는 SensorData 클래스의 객체에 할당되고, DataConverter의 클래스는 센서 데이터 변환을 위한 ICoverter 인터페이스를 구현한 클래스의 객체를 생성하여 데이터 변환을 한다.

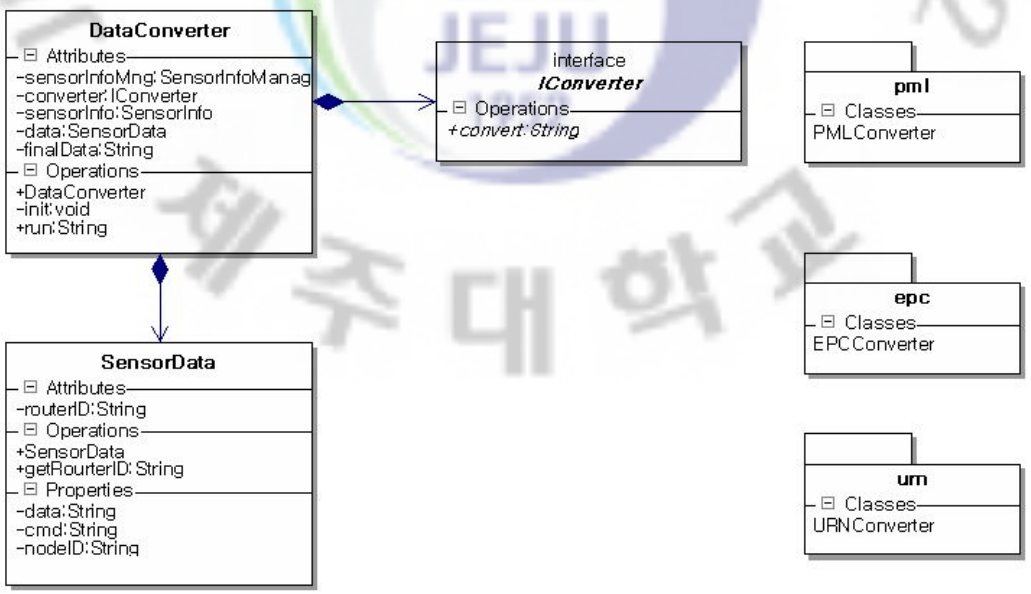

그림 16. convert 패키지의 클래스 다이어그램

pml 패키지, epc 패키지, 그리고 urn 패키지에는 각각의 센서 데이터를 epc, urn, pml 등의 형태로 변환하기 위해 IConverter 인터페이스를 구현한 클래스로 구성되어 있다.

표 15. convert 패키지

클래스	설명
DataConverter	센서 데이터 변환의 전반적인 과정을 처리하는 클래스
IConverter	센서 데이터 변환을 위한 인터페이스
PMLConverter	센서 데이터를 PML 형식으로 변환하는 클래스
EPCConverter	센서 데이터를 EPC 형식으로 변환하는 클래스
URNConverter	센서 데이터를 URN 형식으로 변환하는 클래스
SensorData	센서 데이터를 할당 받는 클래스

3) im 패키지

im 패키지의 주요 클래스들은 센서 데이터 변환에 필요한 센서의 메타정보를 관리하는 클래스로 구성되어 있다. SensorInfo는 각각의 센서마다 데이터 변환에 필요한 센서정보를 갖는 클래스이며, SensorInforMap는 SensorInfo 클래스의 객체를 저장하는 클래스이다. Item 클래스는 각각의 센서마다 구성되는 데이터 항목에 따라 항목에 대한 필드정보를 갖는 클래스이다. 그리고 SensorInfoManager는 데이터 변환에 있어서 각각의 다른 메타정보를 갖는 센서정보를 관리하며 데이터 변환 과정에 도움을 주는 역할을 한다.

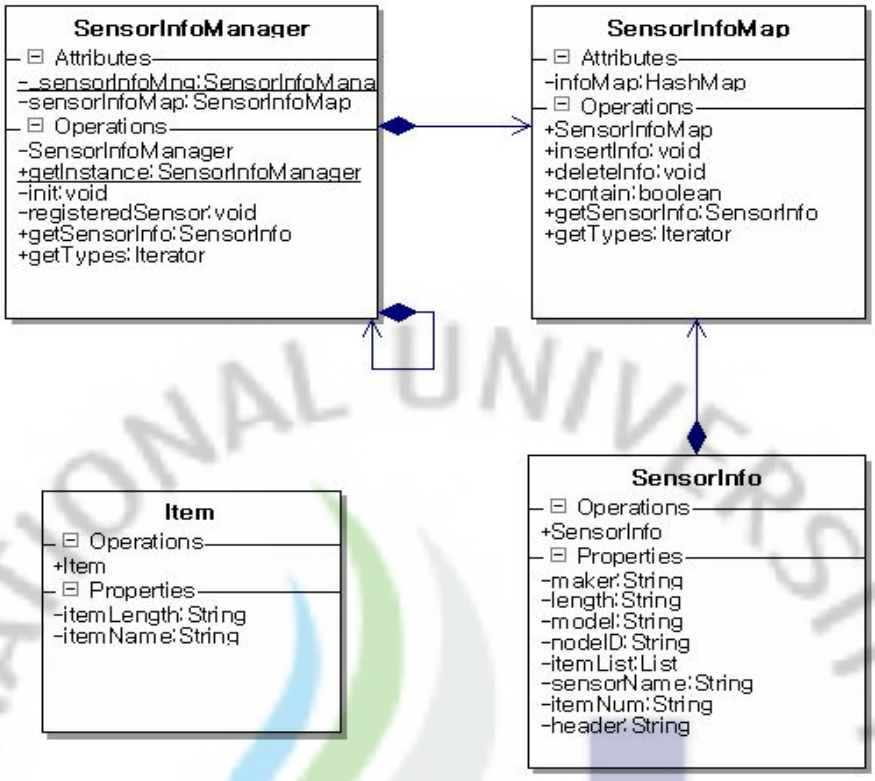

그림 17. im 패키지의 클래스 다이어그램

표 16. im 패키지

클래스	설명
SensorInfoManager	센서 데이터 변환에 필요한 센서의 메타정보를 관리하는 클래스
SensorInfoMap	센서 메타정보를 갖는 SensorInfo 클래스의 객체를 저장하는 클래스
SensorInfo	각 센서마다 데이터 변환에 필요한 메타정보를 갖는 클래스
Item	각 센서마다 구성되는 데이터 항목에 따른 필드정보를 갖는 클래스

```
#### registed sensors
registeredSensors = 4
sensor.type0 = water-quality
sensor.type1 = water-level
sensor.type2 = water-flux
sensor.type3 = lighting

#### water-quality
water-quality.name = water-quality
water-quality.nodeID = 01
water-quality.maker = a
water-quality.model = sc1000
water-quality.itemNum = 5
water-quality.length = 80
water-quality.item0 = electric
water-quality.item0Length = 8
water-quality.item1 = temperature
water-quality.item1Length = 8
water-quality.item2 = chlorine
water-quality.item2Length = 8
water-quality.item3 = Ph
water-quality.item3Length = 8
water-quality.item4 = turbidity
water-quality.item4Length = 8
water-quality.header = 36000001

#### water-level
water-level.name = water-level
water-level.nodeID = 02
water-level.maker = b
water-level.model = ps700
water-level.itemNum = 1
water-level.length = 26
water-level.item0 = waterlevel
water-level.item0Length = 8
water-level.header = 36000002

#### water-flux
water-flux.name = waterflux
water-flux.nodeID = 03
water-flux.maker = c
water-flux.model = wtm100
water-flux.itemNum = 4
water-flux.length = 70
water-flux.item0 = flux
water-flux.item0Length = 8
water-flux.item1 = accumulation
water-flux.item1Length = 16
water-flux.item2 = velocity
water-flux.item2Length = 8
water-flux.item3 = time
water-flux.item3Length = 14
water-flux.header = 36000003

#### lightning
lighting.name = lighting
lighting.nodeID = 04
lighting.maker = d
lighting.model = coronarm-40
lighting.itemNum = 1
lighting.length = 28
lighting.item0 = warning
lighting.item0Length = 4
lighting.header = 36000004
```

그림 18. 센서 데이터 변환을 위한 센서 메타데이터

그림 18은 센서 데이터 변환에 있어 필요한 센서의 정보를 갖고 있는 sensor.conf 프로퍼티 파일로써 각각의 센서마다 갖는 데이터 구성 정보와 데이터 변환에 필요한 정보를 갖고 있다. 센서 네트워크 구성에 있어서의 센싱한 데이터를 보내는 센서를 가리키는 센서노드 ID 정보와, 데이터 변환에 있어서 해당 센서마다 데이터 필드 구성에 따른 데이터 인코딩 정보를 갖고 있는 헤더 정보 또한 이 프로퍼티 파일에 기록된다.

4) util 패키지

util 패지지의 클래스들은 센서 데이터 변환에 있어서 필요한 설정 및 프로퍼티

파일을 로딩하기 위한 클래스이다. 패스 경로 정보를 알기 위한 PahtUtil 클래스와 센서의 메타 데이터를 로딩하기 위한 PropertyUtil 클래스 등이 있다.

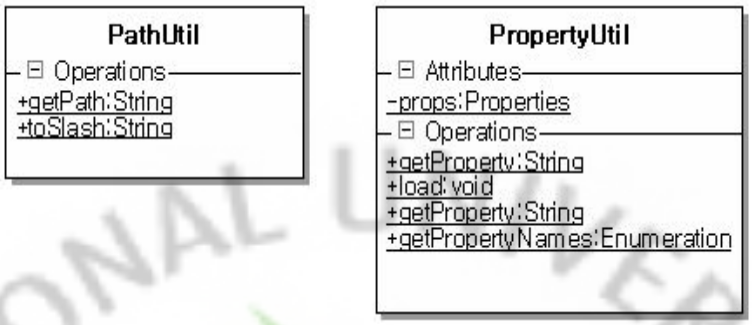

그림 19. util 패키지의 클래스 다이어그램

표 17. util 패키지

클래스	설명
PropertyUtil	센서 데이터 변환을 위한 프로퍼티 파일을 로딩하는 클래스
PathUtil	센서 데이터 변환을 위한 패스 유틸 클래스

3. 실험 데이터

ALE 미들웨어에서 센서 데이터를 처리하기 위해 구현한 결과를 실험하기 위해 수질, 수위, 유량계, 낙뢰 경보 센서에서 수집한 센서 데이터를 바탕으로 실행하였다. 센서 데이터는 수질, 수위, 유량계, 낙뢰 경보 센서에서 수집한 정보를 본 논문에서 제안한 프로토콜에 따라 에뮬레이터에 의해 생성하여 미들웨어로 전송한다.

표 18. 실험 데이터

센서 종류	실험 데이터
수질 센서	0256 0A01 440A 45B1 C56E 0B91 17B6 990C 9D7D AE25 0D190 6927 10E2 AC4C 7C20 F573 2599 ABDC EFFF F03
수위 센서	0229 0A02 44100 2838 4010 0010 1000 0005 86FB D3FF 9021 5D00 0000 0000 0000 0000 0000 0000 000B E8B1 0036 B03
유량계 센서	0245 0A03 4402 3031 3030 3031 3031 3030 3030 3830 2020 2020 302E 3020 6D33 2F68 2020 3134 3937 3537 2E35 3220 6D33 2020 302E 3030 3020 6D2F 7320 3735 3933 3A33 3700 0032 3103 9803
낙뢰 경보 센서	020C 0A04 4421 4646 4646 3030 0DF8 03

표 18은 에뮬레이터에서 생성된 데이터로 ALE 미들웨어로 전송하는 수질, 수위, 유량계, 낙뢰 경보 센서 등 센서 데이터의 예이다.

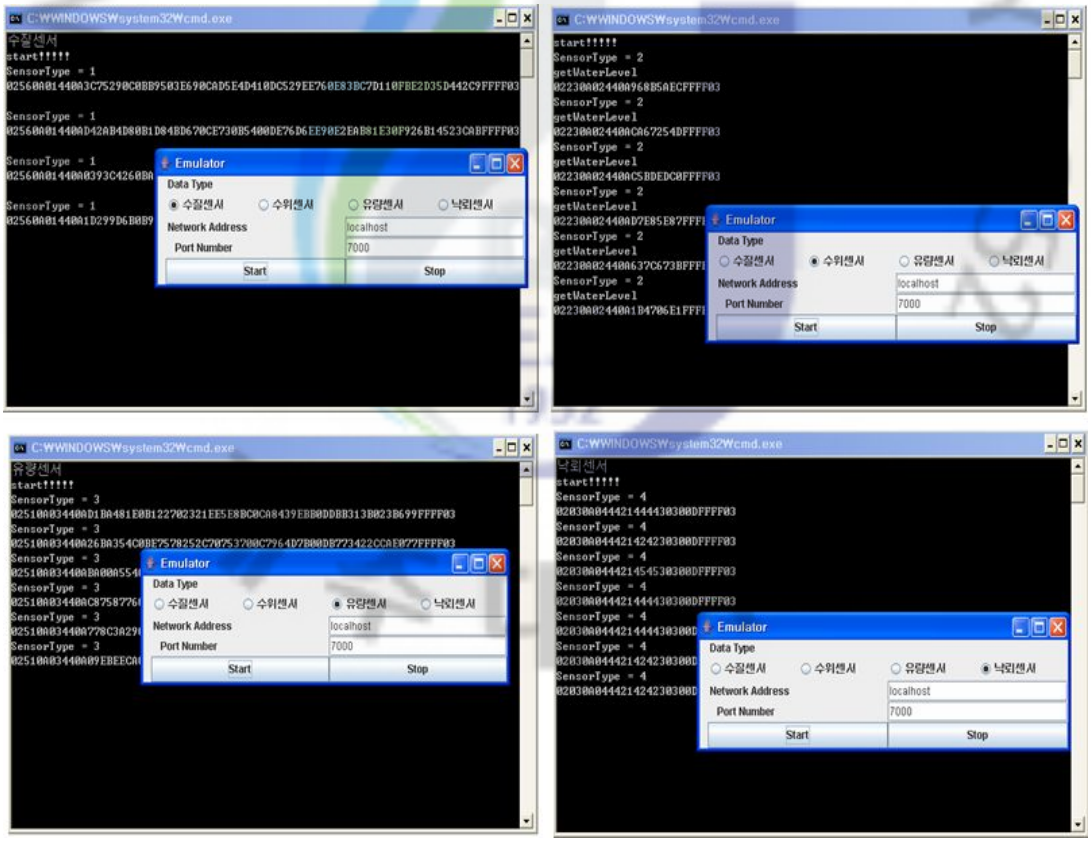

그림 20. 센서 데이터를 생성하는 에뮬레이터

그림 20은 실험을 위해 센서에서 수집한 데이터를 제안하는 프로토콜에 맞게 데이터를 생성하는 에뮬레이터를 실행한 모습으로 각각의 센서 타입을 선택하여 실행하게 되면 수질, 수위, 유량, 낙뢰 경보 등의 데이터를 생성하여 미들웨어로 데이터를 전송한다.

4. 실험 결과

1) 센서 데이터 변환 실험

센서 데이터를 기존의 ALE 미들웨어에서 처리가 가능한 PML로 변환하기 위해 센서에서 수집한 데이터를 URN 형태로 변환된 결과를 각각의 센서별로 정리한 결과는 다음과 같다. 센서 데이터의 변환 과정은 센서에서 수집한 원시데이터를 EPC 코드 형태로 변환한 후, 이를 URN 코드 형태로 변환한다.

표 19. 수질 센서 데이터 변환 결과의 예

수질 센서	데이터 변환 결과
센싱 데이터	0256 0A01 440A 05AA 409D 0BB4 C990 C10C 4350 0903 0DA9 8DA2 770E 4EDC 2560 0F32 33E8 27C1 22FF FF03
EPC 코드 형태	3600 0001 05AA 409D B4C9 90C1 4350 0903 A98D A277 4EDC 2560
URN 코드	urn:sensor:a:sc1000:05AA409D.B4C990C1.43500903.A98DA277.4EDC2560

표 19는 수질 데이터의 변환 결과를 보여준다. 수질 데이터는 전기전도도, 온도, 잔류염소, Ph, 탁도 등의 항목으로 나타낸다. 변환된 URN 형태의 데이터에서는 전송된 센서 데이터가 'a'사의 'sc1000' 모델의 센서에서 수집한 데이터로 첫 번째 필드는 전기전도도, 두 번째 필드는 온도, 세 번째 필드는 잔류염소, 네 번째 필드는 Ph, 그리고 마지막으로 다섯 번째 필드는 탁도를 나타내는 것이다.

표 20. 수위 센서 데이터 변환 결과의 예

수위 센서	데이터 변환 결과
센싱 데이터	0223 0A02 440A 38EC 04BE FFFF 03
EPC 코드 형태	3600 0002 38EC 04BE
URN 코드	urn:sensor:b:ps700:38EC04BE

수위 데이터는 데이터 변환 과정을 통하여 표 20과 같이 변환되었다. 수위 센서 데이터에서는 수위 값을 나타내는 하나의 필드로 구성되어 있어서 'b'사의 'bps700' 모델에서 '38EC04BE'라는 수위 값을 측정한 결과를 보여준다.

표 21. 유량계 센서 데이터 변환 결과의 예

유량계 센서	데이터 변환 결과
센싱 데이터	0251 0A03 440A 4C91 D39D 0BD0 C08B 887C 6870 7B0C 4694 981C 0DDA B468 A17B 812D FFFF 03
EPC 코드 형태	3600 0003 4C91 D39D D0C0 8B88 7C68 707B 4694 981C DAB4 68A1 7B81 2D
URN 코드	urn:sensor:c:wtm100:4C91D39D.D0C08B887C68707B.4694981C.DAB468A17B812D

유량계 센서 데이터의 경우는 수질 데이터와 마찬가지로 유량 데이터를 표현하기 위해 순간유량, 누적유량, 순간유속, 적산시간으로 표현한다. 표 21에서 보듯이 원시 데이터를 URN 코드로 변환된 결과에서는 유량계 데이터의 센서 제작회사. 모델명, 수집한 데이터를 확인할 수 있다.

표 22. 낙뢰 경보 센서 데이터 변환 결과의 예

낙뢰 경보 센서	데이터 변환 결과
센싱 데이터	0203 0A04 4421 4242 3030 0DFF FF03
EPC 코드 형태	3600 0004 4242
URN 코드	urn:sensor:d:coronarm-40:4242

낙뢰 경보 센서의 경우에서도 위의 센서들과 같이 센서에서 수집한 원시 데이터를 URN형태로 변환하여 낙뢰 경보를 나타낸다. 이상에서 살펴본 바와 같이 수질, 수위, 유량, 낙뢰 경보 센서 데이터 등 각 센서별 100개 이상의 센서 데이터를 실험한 결과 데이터 변환이 정상적으로 처리됨을 확인할 수 있었다.

2) 미들웨어에서의 센서 데이터 처리 실험

기존의 ALE 미들웨어에서 센서 데이터 변환 모듈을 연동하는 방법은 센서 데이터 변환 모듈을 'jar'로 만들어 기존의 ALE 미들웨어에 빌드패스 과정을 거쳐 센서 데이터 변환 모듈인 jar 파일을 RFID 미들웨어에 넣는다. 그리고 시스템 설정 파일을 저장하는 디렉토리에 센서 데이터 변환에 필요한 메타정보를 갖고 있는 'sensor.conf'파일을 넣으면 된다. 그림 21은 개발 도구인 이클립스에서 이와 같은 과정을 거쳐 센서 데이터 처리를 하는 모습이다.

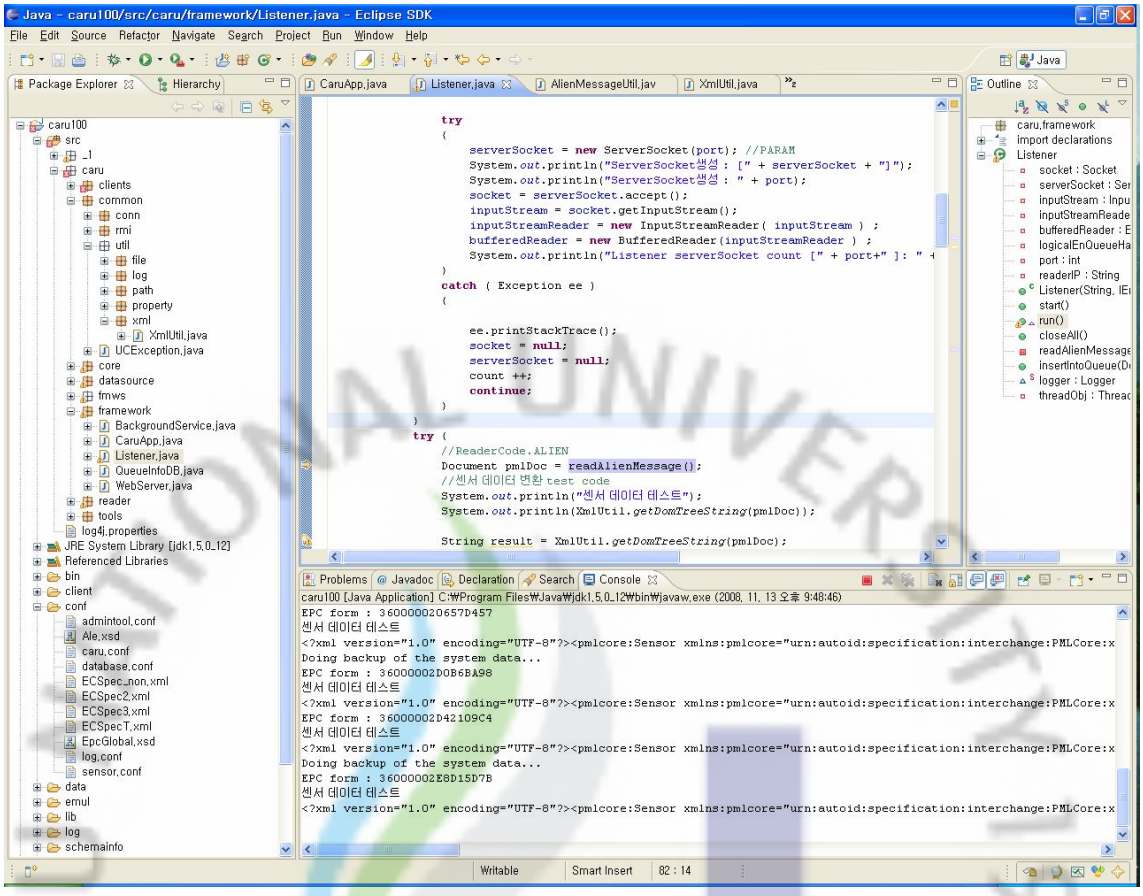

그림 21. 미들웨어에서의 센서 데이터 처리

미들웨어에 센서 데이터 처리 모듈을 넣어서 실행한 결과 다음 그림과 같이 센서 데이터가 PML로 변환되어 처리되어 진다.

```
<?xml version="1.0" encoding="UTF-8"?>
<pmlcore:Sensor xmlns:pmlcore="urn:autoid:specification:interchange:PMLCore:xml:schema:1"
        xmlns:pmluid="urn:autoid:specification:universal:Identifier:xml:schema:1"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="urn:autoid:specification:interchange:PMLCore:xml:schema:1 ../SchemaFiles/Interchange/PMLCore.xsd">
        <pmluid:ID>203.253.213.136</pmluid:ID>
        <pmlcore:Observation>
                <pmluid:ID>0</pmluid:ID>
                <pmlcore:DateTime>2006/10/02 18:03:15</pmlcore:DateTime>
                <pmlcore:Tag>
                        <pmluid:ID>urn:sensor:a:sc1000:02D544EB.21EA4839.E7352015.19C8B001.275BD197</pmluid:ID>
                </pmlcore:Tag>
        </pmlcore:Observation>
</pmlcore:Sensor>
```

그림 22. PML로 변환된 수질 센서 데이터의 예

```
<?xml version="1.0" encoding="UTF-8"?>
<pmlcore:Sensor xmlns:pmlcore="urn:autoid:specification:interchange:PMLCore:xml:schema:1"
        xmlns:pmluid="urn:autoid:specification:universal:Identifier:xml:schema:1"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="urn:autoid:specification:interchange:PMLCore:xml:schema:1 ../SchemaFiles/Interchange/PMLCore.xsd">
        <pmluid:ID>203.253.213.136</pmluid:ID>
        <pmlcore:Observation>
                <pmluid:ID>0</pmluid:ID>
                <pmlcore:DateTime>2006/10/02 18:03:15</pmlcore:DateTime>
                <pmlcore:Tag>
                        <pmluid:ID>urn:sensor:b:ps700:1C2D9348</pmluid:ID>
                </pmlcore:Tag>
        </pmlcore:Observation>
</pmlcore:Sensor>
```

그림 23. PML로 변환된 수위 센서 데이터의 예

```
<?xml version="1.0" encoding="UTF-8"?>
<pmlcore:Sensor xmlns:pmlcore="urn:autoid:specification:interchange:PMLCore:xml:schema:1"
        xmlns:pmluid="urn:autoid:specification:universal:Identifier:xml:schema:1"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="urn:autoid:specification:interchange:PMLCore:xml:schema:1 ../SchemaFiles/Interchange/PMLCore.xsd">
        <pmluid:ID>203.253.213.136</pmluid:ID>
        <pmlcore:Observation>
                <pmluid:ID>0</pmluid:ID>
                <pmlcore:DateTime>2006/10/02 18:03:15</pmlcore:DateTime>
                <pmlcore:Tag>
                        <pmluid:ID>urn:sensor:c:wtm100:B143521A.37C3DA2673942385.194159C7.E4E170E23463CA</pmluid:ID>
                </pmlcore:Tag>
        </pmlcore:Observation>
</pmlcore:Sensor>
```

그림 24. PML로 변환된 유량계 센서 데이터의 예

```
<?xml version="1.0" encoding="UTF-8"?>
<pmlcore:Sensor xmlns:pmlcore="urn:autoid:specification:interchange:PMLCore:xml:schema:1"
        xmlns:pmluid="urn:autoid:specification:universal:Identifier:xml:schema:1"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="urn:autoid:specification:interchange:PMLCore:xml:schema:1 ../SchemaFiles/Interchange/PMLCore.xsd">
        <pmluid:ID>203.253.213.136</pmluid:ID>
        <pmlcore:Observation>
                <pmluid:ID>0</pmluid:ID>
                <pmlcore:DateTime>2006/10/02 18:03:15</pmlcore:DateTime>
                <pmlcore:Tag>
                        <pmluid:ID>urn:sensor:d:coronarn-40:4242</pmluid:ID>
                </pmlcore:Tag>
        </pmlcore:Observation>
</pmlcore:Sensor>
```

그림 25. PML로 변환된 낙뢰 경보 센서 데이터의 예

위의 결과와 같이 PML 형식으로 변환된 센서 데이터는 기존의 ALE 기반의 RFID 미들웨어에서 처리가 가능하게 된다. URN 형태로 변환된 센서 데이터로 인해 미들웨어에서 제공하는 ALE 인터페이스를 이용하여 사용자 및 어플리케이션에서는 각 센서마다 원하는 요구사항에 따라 필터링 및 그룹핑 등의 결과를 서비스 받을 수 있다.

3) 센서 데이터 처리를 위한 미들웨어 성능 평가

다음에 그림들은 ALE 미들웨어 실행시의 CPU 사용량, 동작하는 쓰레드의 개수 등 현재 자바 가상 머신에서 사용하는 자원 정보를 보여준다. 센서 데이터 처리를 위해 ALE 미들웨어를 실행하였을 때의 자원 소모 및 메모리 사용량은 그림 26과 그림 27을 보면 확인할 수 있다.

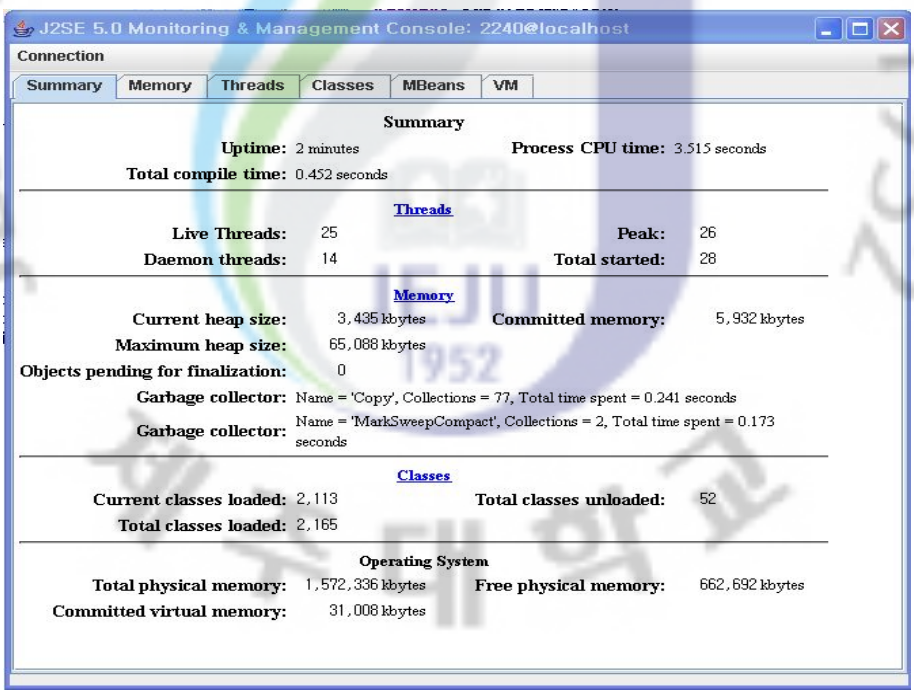

그림 26. ALE 미들웨어의 Summary

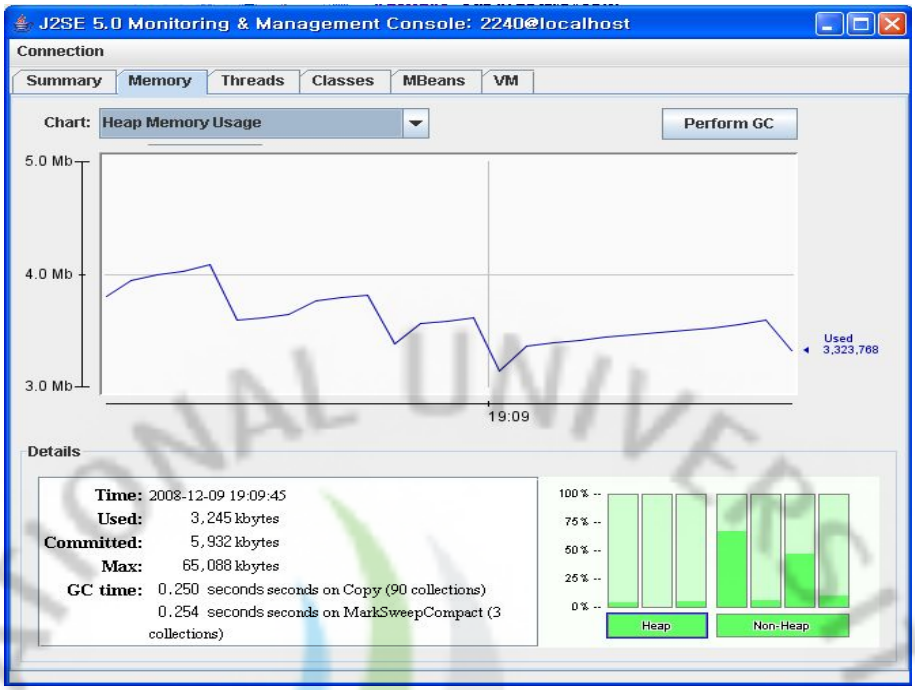

그림 27. ALE 미들웨어의 메모리 사용량

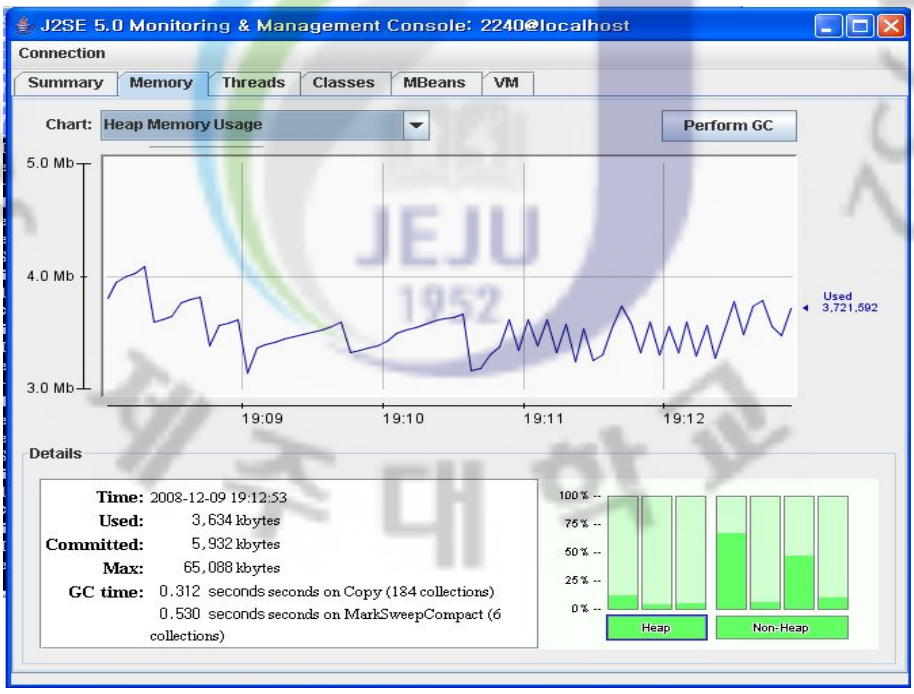

그림 28. 센서 10개를 연결한 ALE 미들웨어의 메모리 사용량

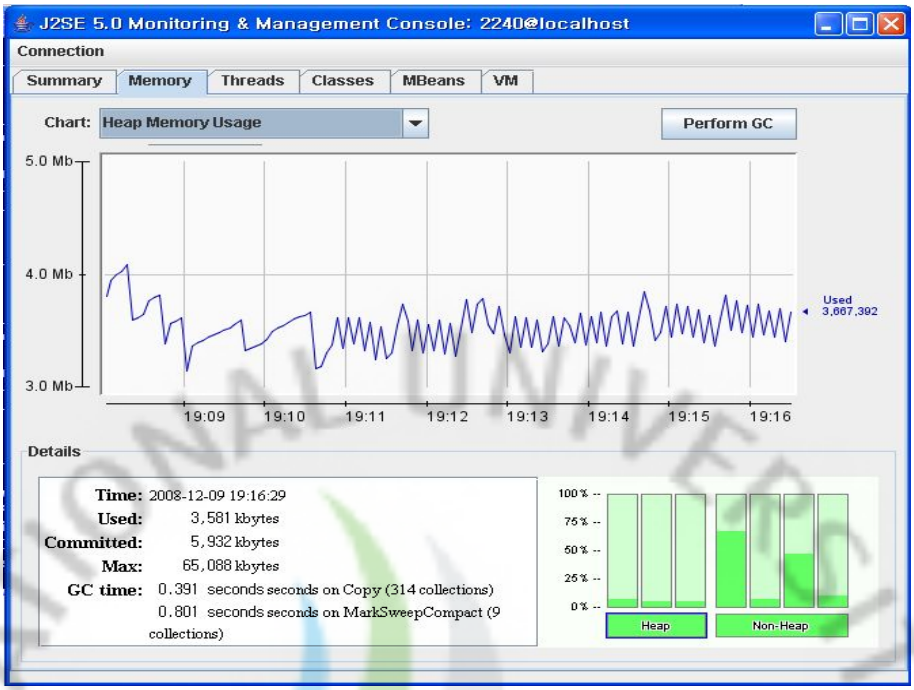

그림 29. 센서 50개를 연결한 ALE 미들웨어의 메모리 사용량

그림 28은 ALE 미들웨어 실행 후 센서 10개를 연결하여 센서에서 센싱한 데이터를 처리하는 경우의 메모리 사용량으로 기존의 미들웨어만을 실행하였을 경우와 비교하여 메모리 사용량이 증가하는 모습을 보인다. 그러나 그림 29를 보면 센서의 개수를 10개에서 50개로 늘렸을 경우의 메모리 사용량의 변화는 데이터 처리에 있어서의 변화는 크게 차이가 나지 않는 모습을 확인 할 수 있었다.

제안하는 방법에서의 센서 데이터 처리를 위한 알고리즘 복잡도는 센서 데이터 처리를 과정에 있어서 입력된 데이터와 미들웨어에 등록된 센서 타입을 비교하여 처리하게 되어 있어 등록된 센서의 개수에 따라 알고리즘 복잡도가 정해지므로 제안하는 방법의 알고리즘 복잡도는 O(n)이 된다.

V. 결론 및 토의

본 논문에서는 국제 표준의 ALE 미들웨어를 기반으로 기존 방법을 변경하지 않고서도 RFID가 아닌 다양한 센서 데이터 스트림을 효과적으로 처리할 수 있는 방법을 제안하였다. 리더를 센서로 간주하여 리더에서 읽힌 태그 데이터를 미들웨어가 처리하듯이 센서에서 수집된 정보를 태그 데이터처럼 처리가 가능하도록 하였다.

사실상의 표준인 ALE 스펙은 유비쿼터스 서비스를 위한 RFID 미들웨어에서의 태그 데이터 처리를 위한 기능과 내용을 기술하고 있다. ALE는 리더 또한 태그 정보를 인식하는 하나의 센서로 간주하며, 스펙에서 기술한 필터링, 그룹핑, 카운팅 등 주요 기능들과 내용들은 RFID 미들웨어뿐만 아니라 USN 미들웨어에서도 필요한 일반적인 내용까지 담고 있다. 이에 따라 RFID 미들웨어에서 태그 데이터이외에 센서 데이터 처리가 가능해지면 서비스 이용자들은 리더에서 읽은 태그정보를 사용자 요청에 따라 이벤트 기반의 정보를 제공하듯이 센서에서 수집된 정보를 사용자 요청에 따라 서비스를 받을 수 있다. 즉, ALE 미들웨어가 USN 미들웨어 중 이벤트 기반의 server-side 미들웨어로서의 역할을 한다.

RFID 태그 정보를 통하여 사물의 인식 정보만을 처리하는 것뿐만 아니라 RFID 이외의 센서 데이터를 처리할 수 있는 미들웨어를 개발할 수 있으므로 사물의 인식 정보와 센서 데이터를 활용하여 다양한 응용 서비스를 저비용으로 제공할 수 있다. 다양한 응용과 센서들의 표준화된 인터페이스를 미들웨어가 제공함으로써 이기종 센서를 저렴한 비용으로 쉽게 연결할 수 있는 범용성을 가질 수 있다.

USN 및 RFID를 성공적으로 구현하기 위해서는 사용자와 응용 환경에 맞는 사용자 중심의 적절한 기술 적용이 필수적이다. 본 연구를 통하여 RFID/USN 융합에 기반 한 응용 서비스 제공을 위한 기반 기술을 확보할 수 있을 뿐만 아니라 기존의 국제 표준에 근거한 방법을 확장함으로써 상대적으로 적은 비용으로 최대의 효과를 얻을 수 있을 것으로 기대된다.

참고문헌

[1] 홍원표, “유비쿼터스 컴퓨팅 개념과 센서네트워크”, 조명전기설비학회지, 제19권 제4호, 2005.

[2] M. Weiser, J. S. Brown, "The coming of age of calm technology", Xerox PARC, 1996

[3] 송석현, 신상철, "RFID/USN 표준화 동향 및 이슈", 정보과학회지, 제22권 제12호, 2004.

[4] 장병준, 안선일, 이윤덕, "RFID/USN 기술개발 동향", 한국정보학회, 정보과학회지, 제23권 제2호, 2005.

[5] Intel, “Building the Digital Supply Chain: An Intel Perspective,” Intel Corp., 2005.

[6] METRO, “The METRO Group Future Store Initiative,“ http://www.futurestore.org.

[7] http://www.infoworld.com/article/04/07/23/HNrfidimplants1, 2004.

[8] 황종성, “유비쿼터스 사회 구현을 위한 IT 전략 연구,” 한국정보사회진흥원, 2006.

[9] 홍연미, 변영철, “ALE 기반 RFID 미들웨어 설계 및 구현,” 한국해양정보통신학회논문집, 제11권, 제4호, pp.648-655, 2007.

[10] EPCglobal, The Application Level Events (ALE) Specification Version 1.0,”September 2005.

[11] EPCglobal, "EPC Tag Data Standards Version 1.1 Rev.1.27", 2005.

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

[13] Auto-ID Lab, http://www.autoidlabs.org.

[14] EPCglobal Inc., "EPCglobal Architecture Framework Final Version", 1 July 2005.

[15] K. Traub and S. Bent, etc., The Application Level Events Specification, Version 1.0, EPCglobal Working Draft, Oct. 14, 2004.

[16] K. Traub and S. Rehling, etc., EPC Information Service Version 1.0 Specification, EPCglobal Working Draft, Oct. 12, 2004.

[17] 황재각, 정태수, 김영일, 이용준, ETRI, "RFID 미들웨어 기술 동향 및 응용", 전자통신동향분석, 제20권 제3호, 2005.

[18] 오세원, 박주상, 이용준, "RFID SW기술과 표준화 동향" 한국통신학회지. 제24권 제7호, 2007.

[19] 성종우, 김대영, "RFID와 USN 통합 인프라스트럭처를 위한 EPC 센서 네트워크" 한국통신학회 논문지, 제23권 제12호, 2006.

[20] 강정훈, 황태호, 송병철, "유비쿼터스 센서 네트워크 기술", 방송공학회지, 제10권,

[21] 김민수, 김광수, 이용준, "USN 미들웨어 특징 및 기술개발 동향", ERRI 주간기술동향 통권 1284호, 2007.

[22] 김대영, 성종우, 송현주, 김수현, "센서 네트워크 미들웨어 기술", 전자공학회지 , 제32권 제7호, 2005.

[23] 김민수, 이용준, 박종헌, ETRI, "USN 미들웨어 기술개발 동향", 전자통신동향분석, 제22권, 제3호, 2007.6

[24] A. Mainwaring, J. Polastre, R. Szewczyk, and D.Culler, Wireless Sensor Networks for Habitat Monitoring,"ACM, Sensor Networks and Applications, Sep. 2002, pp.88-97.

[25] A. Baptista, T. Leen, Y. Zhang, A. Chawla, D. Maier, W. Feng, W. Feng, J. Walpole, C. Silva, and J. Freire,"Environmental Observation and Forecasting Systems:Vision, Challenges and Successes of a Prototype,"Encyclopedia of Physical Science and Technology(R.A. Meyers, Ed.), Academic Press, ThirdEdition, Vol.5, pp.565-581.

[26] N. Xu et al., "A Wireless Sensor Network for Structural Monitoring," ACM Sensys, Nov. 2004, pp.13-24.

[27] Aleksandar Milenkovi?, Chris Otto, and Emil Jovanov,"Wireless Sensor Networks for Personal Health Monitoring: Issues and an

Implementation,"ELSEVIER, Computer Communications, In Press, Corrected Proof, Available online 6 Mar. 2006.

[28] 김민수, 이은규, 장병태, "USN 기반 차세대 텔레매틱스 서비스 연구 동향," IITA, 주간기술동향, 통권 1207호, 2005. 8.

[29] Shuoqi Li, Sang H. Son, and John A. Stankovic,"Event Detection Services Using Data Service Middleware in Distributed Sensor Networks,"Information Proc. in Sensor Networks, Apr. 2003, LNCS 2634, pp.502-517.

[30] T. Liu and M. Martonosi, "Impala: A Middleware System for Managing Autonomic," Parallel Sensor Systems, Proc. ACM SIGPLAN Symp. Principles and Practice of Parallel Programming, 2003, pp.107?118.

[31] S.R. Madden, M.J. Franklin, and J.M. Hellerstein,"TinyDB: An Acquisitional Query Processing System for Sensor Networks," ACM TODS, Vol.30, No.1, 2005, pp.122?173.

[32] Yong Yao and J.E. Gehrke, "The Cougar Approach to In-Network Query Processing in Sensor Networks,"SIGMOD RECORD, Vol.31, No.3, Sep. 2002.

[33] 김민수, 이은규, "유비쿼터스 환경에서의 센서 데이터베이스 기술," IITA, 주간기술동향, 통권 1187호, 2005. 3.

[34] Enterprise Information Architecture for RFID and Sensor-Based Services, Oracle White Paper, 2006.

[35] TagsWare, Agile RFID Solution, http://www.tagsware.com.

[36] A. Gupta, M. Srivastava, "Developing Auto-ID Solutions using Sun Java System RFID Software", 2004.

[37] 이훈순, 최현화, 김병섭, 이명철, 박재홍, 이미영, 김명준, 진성일, "UbiCore : XML 기반 RFID 미들웨어 시스템," 정보과학회논문지, 제33권, 제6호, pp. 578-589, 2006.

[38] H. S. Lee, S. I. Jin, "An Effective XML-Based Sensor Data Stream Processing Middleware for Ubiquitous Service", Lecture Note in Computer

Science, vol. 4704, Springer, Heidelberg, 2007, pp.844-857

[39] T. S. Cheong, Y. G. Kim, Y. J. Lee, "REMS and RBPTS: ALE-compliant RFID Middleware Software Platform", International Conference on Advanced Communication Technology, 2006, pp.699-704

[40] W. Wang, J. W. Sung, D. Y. Kim, "Complex Event Processing in EPC Sensor Network Middleware for Both RFID and WSN", 11th IEEE Symposium on Object Oriented Real-Time Distributed Computing, 2008, pp. 165-169.