TinyDB와 라인트레이서를 활용한 TinyOS기반의 센서 데이터 처리 모듈 설계 및 구현

정회원 이 상 훈*, 문 승 진*

The Design and Implementation of Sensor Data Processing Module Based on TinyOS Utilizing TinyDB and LineTracer

Sang-hoon Lee*, Seung-jin Moon* ***Regular Members***

요 약

유비쿼터스 컴퓨팅 기술에 대한 관심이 하드웨어, 통신, 데이터베이스 등의 다양한 분야에서 매우 높게 나타나고 있음에 따라 무선 센서 네트워크에 대한 연구가 활기를 띠고 있다. 특히 최근에 새로 보급되는 스마트 센서들은 각 센서 노드들이 정보를 실시간으로 수집하고 분석할 수 있어, 이에 따른 데이터 처리 역시 중요한 이슈로 부각되고 있다. 본 논문에서는 이동 경로를 주행할 수 있는 라인트레이서를 설계 및 구현하여 쿼리 프로세싱 시스템인 TinyDB를 활용해서 라인트레이서에 센서 노드를 탑재하여 이동 경로 주변의 데이터를 수집하였고, 정지해 있는 특정한 위치에서의 데이터와 비교하였으며, 수집된 데이터를 저장하여 웹 서버에서 나타내었다. 또한 사용자의 편의를 위해 임베디드 보드에서 웹 서버로의 접근을 통해서 터치 식으로 쉽게 데이터를 볼 수 있도록 구현하였다.

Key Words : TinyOS, TinyDB, Lego MindStorm, LineTracer

ABSTRACT

The study of sensor network database is beginning to liven up as we are interested in Ubiquitous Computing technology in hardware, communication, database and so on. Especially, as new smart sensors have capabilities of real-time information gathering and analysis of each sensor node, data processing becomes an important issue in Ubiquitous Computing. In thesis, we have applied TinyDB(query processing system) to carry sensor node with line tracer which can follow the fixed path. After we gathered data around path, we have processed data in TinyDB GUI, gathered data, displayed data on a web server. Also we have a web browser on an embedded board for convenient user interface and implemented touch screen such that users can operate with a finger.

Ⅰ. 서 론

최근에 새로 보급되는 스마트 센서들은 저렴하고 작은 크기에도 불구하고 새로운 정보를 취득하기 위한 센싱 부분, 취득한 정보를 저장하고 분석하기 위한 컴퓨팅 부분, 분석된 정보를 제공하기 위한 무선통신 부분 그리고 전원 부분으로 구성되어 각 센서 노드들이 정보를 실시간으로 수집할 수 있게 되

* 수원대학교 컴퓨터학과 실시간 임베디드 멀티미디어 데이터베이스 연구실 ({vkfka1, sjmoon}@mail.suwon.ac.kr)
논문번호 : KICS2006-08-352, 접수일자 : 2006년 8월 25일, 최종논문접수일자 : 2006년 9월 21일

었다. 이러한 스마트 센서들에는 저 전력, 적은 코드 사이즈, 최소한의 하드웨어 리소스를 사용하는 임베디드 OS가 필수적인데 현재 이러한 OS로 주목받고 있는 OS가 바로 TinyOS이다[1-3]. TinyOS는 UC 버클리에서 진행해온 스마트 더스트(Smart Dust) 프로젝트에 사용하기 위하여 개발된 컴포넌트 기반 임베디드 운영체제이고, TinyDB는 이러한 TinyOS 센서 네트워크로부터 정보를 추출하기 위한 쿼리 프로세싱 시스템이다[4, 5].

본 논문에서는 위에서 언급한 TinyOS와 TinyDB를 기반으로 포팅 된 센서 노드를 레고 마인드 스톰을 기반으로 설계 한 라인트레이서에 탑재하여 이동을 시킴으로써 정지해 있는 센서 노드와 비교하여 움직이는 센서 노드를 위한 데이터 처리 모듈을 설계하였다. 기존 TinyOS를 탑재한 노드에서 특정한 주위 환경의 탐지를 하는 경우 원하는 위치에서 노드를 고정시켜야 안정된 탐지를 할 수 있고, 그 외에 다른 위치를 탐지하기 위해서 일정한 간격으로 수 많은 노드들을 분산시킴으로써 탐지를 하였으나, 노드들의 환경적 원인으로의 분실, 위치변화를 알기가 쉽지 않을 뿐 아니라 노드의 수 증가라는 문제를 야기 시킬 수도 있다. 본 논문에서의 데이터 처리 모듈 및 설계는 특히 한 개의 노드를 일정한 간격으로 라인트레이서를 이용해서 이동시켜 특정한 정지노드들의 데이터를 분석하여 비교함으로써 정지노드들의 역할을 대신하는 노드의 수적인 효율성을 추구하였고, 데이터 측정을 위한 시스템의 편리한 사용을 위해서 웹서버를 만들어 임베디드 장비에서 데이터 표현이 가능하도록 데이터 처리 모듈을 구현하였다. 먼저 고정경로에서 노드를 운반하기 위해 작은 로봇으로 설계된 라인 트레이서를 구현하였고 하나의 노드는 움직이고, 다른 노드들은 고정함으로써 각 노드들을 구분하였다. 그리고 각각의 노드로부터 수집된 고정노드의 데이터와 움직이는 노드의 데이터를 비교하였다. 특히 움직이는 노드에서 각 노드의 데이터를 수집하기 위해 규칙적인 시간으로 데이터를 수집하였고, 이 데이터는 웹서버에 저장되어 임베디드 보드에서 사용자가 쉽게 데이터를 볼 수 있도록 구현하였다. 전체적인 구성은 첫째, 움직이는 센서를 위한 데이터 처리 모듈에 관해서 설명하였고, 2장에서는 본 논문에서 사용된 TinyOS, TinyDB, Lego Mind Storm에 관해서 설명하였고, 데이터 처리 모듈 구조와 개발된 레고 마인드 스톰, 그리고 TinyDB의 활용에 관해서는 3장에서 설명하였다. 그리고 4장에서는 처리된 데이터를 표현 및 분석하였고, 5장에서는 결론 및 향후 연구과제에 관하여 설명하였다.

Ⅱ. 관련연구

2.1. TinyOS와 TinyDB의 특징

TinyOS는 센서 네트워크를 위해서 특별히 설계된 오픈 소스 임베디드 운영체제이다[7]. 이 운영체제는 nesC 언어에 의해 제공되는 컴포넌트 기반 모델이다[8]. TinyOS는 메모리와 디바이스 관리를 포함하고, 통신 레이어 들을 위하여 태스크 스케줄링과 프로토콜 스케줄링을 포함하고 있고, 모든 모트들은 ROM에 200K로 프로그램 된 작은TinyOS를 가지고 있고, 그 위에 애플리케이션을 실행한다[3-4].

그림 1은 TinyOS의 통신 스택을 보여준다. TinyOS는 애플리케이션 레이어, 데이터 링크 레이어, 물리 링크 레이어에서 네트워크 라우팅을 구현하고, 다른 플랫폼들과 Mica2, Miica와 같은 센서 보드들과 PC에 포팅 될 수 있다.

TinyDB는 TinyOS 센서 네트워크로부터 정보를 추출하기 위한 쿼리 프로세싱 시스템이다. TinyDB는 사용자가 임베디드 C코드를 작성하지 않는 대신에 원하는 데이터를 추출하기 위해서 간단한 SQL 인터페이스를 제공한다. 즉 TinyDB는 노드에서 데이터를 수집하고, 필터링을 통해 모으고 효율적으로 전달한다[5]. TinyDB의 특징은 첫째, 센서 네트워크에서 받아진 센서의 종류를 기술하기 위해 메타 데이터 카탈로그를 제공한다. 둘째, 사용자가 다루기 쉽게 원하는 데이터를 기술하는 평서 쿼리 언어를 사용한다. 셋째, 네트워크 망을 가지고 있다. 넷째, 동시에 같은 셋의 노드들을 실행할 때 멀티플 쿼리를 할당한다. 다섯째, TinyDB 센서 네트워크를 확장하기 위해서, 단순히 새로운 노드에 표준 TinyDB 코드를 다운로드 하고, 리셋만 하면 된다. 또한 TinyDB가 포팅 된 노드들은 서로 쿼리를 공유한다[10-11].

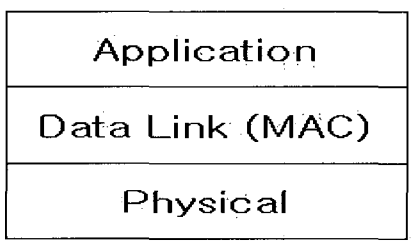

그림 1. TinyOS 통신 스택

2.2. 레고 마인드 스톰

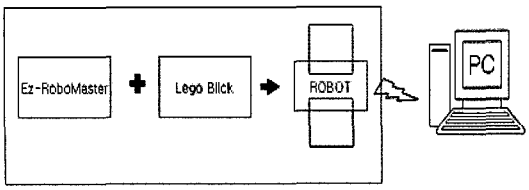

그림 2. 레고 시스템

본 논문에서 레고 마인드 스톰은 라인트레이서를 설계하기 위해 이용한 도구로써 레고에 마인드 스톰을 더한 합한 것을 의미한다. 특정한 목적을 위해 만들어진 로봇이 기구적인 문제로 정상적인 작동이 어렵다고 할 때, 다시 수정하기란 너무나 많은 시간과 비용이 들었기 때문에 마인드 스톰은 빠른 시간에 구조물을 만들고 기타 비용을 줄이기 위한 목적으로 레고와 MIT(메사추세츠 공과대학)의 15년간의 공동연구로 만들어졌다[12, 13].

그림 2는 레고 시스템의 기본 구조를 보여준다. 레고 마인드 스톰은 로봇 제작을 위한 브릭들로 이루어져있다. 기본적인 브릭들은 레고 테크닉 시리즈

의 규격을 따르고 있으며, 마인드 스톰 시리즈는 테크닉 시리즈와 함께 추가적인 전자 장치들을 포함하고 있다. 기본적인 레고 시스템의 구성은 그림 2와 같이 I/O 기능을 가진 프로그래밍 된 Ez-RoboMaster가 레고로 만든 기구 부에 합쳐져 완성된 로봇이 된다.

Ⅲ. 이동 센서 데이터 처리 모듈

이번 장에서는 이동 센서 데이터 처리 모듈의 설계 및 구현에 관해서 설명하였다. 먼저 데이터 프로세싱 모듈 개발 환경에 관해서 설명하였고, 3.2절에서는 라인 트레이서의 알고리즘과 구현에 관해서 설명하였고, TinyDB 활용은 3.3절에서 나타내었다. 마지막으로 3.4절에서는 웹 서버에서 이동 센서 데이터를 나타내기 위해서 임베디드 보드인 X-Hyper255B 보드에서 웹 브라우저를 통한 데이터를 나타내어 사용자가 터치 식으로 사용하도록 구현하였다.

3.1. 데이터 처리 모듈 개발 환경

그림 3은 이동 센서 데이터 처리를 위한 개발 환경을 보여준다. 구성 환경은 데이터 감지를 위한 라

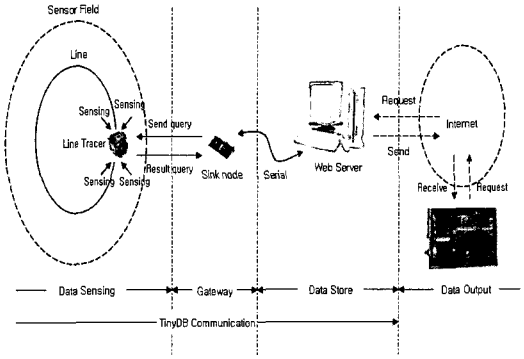

그림 3. 데이터 처리 모듈 구조도

인트레이서, 센서노드, 게이트웨이 역할을 하는 싱크노드, 데이터를 저장하기 위한 웹 서버, 사용자에게 데이터를 나타내기 위한 임베디드 보드로 구성된다. 본 논문에서는 데이터 통신을 위해서 Micaz 모트를 사용한다. Micaz 모트는 Atmel ATmega128L 마이크로프로세서를 탑재한 매우 작은 센서 노드이고, 데이터를 송수신 할 때 사용된다. 그리고 주위의 빛을 감지해서 Micaz로 전달하기 위해 MTS310 센서보드를 사용한다. X-Hyper255B 임베디드 보드는 pxa255 마이크로프로세서를 탑재하고 있고, 사용자에게 데이터를 나타내기 위해서 사용된다. 웹 서버에서 Micaz와 MTS310센서 보드로 구성된 센서 노드에게 데이터를 요청하기 위해서는 싱크노드를 통해서

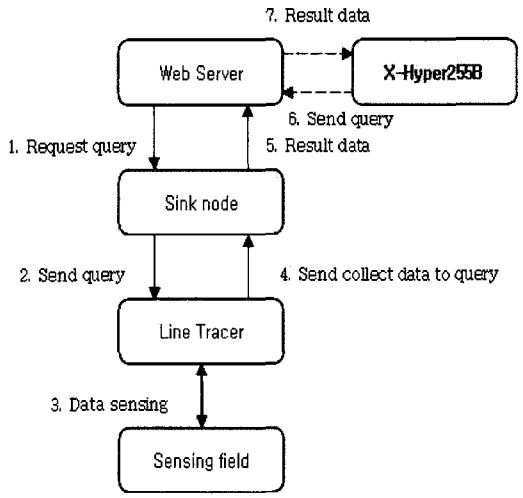

그림 4. 모듈의 데이터 흐름도

쿼리를 전송하게 되고, 센서 노드에서는 쿼리에 맞는 조건을 가진 데이터를 센싱하게 되고, 센싱된 데이터는 싱크노드를 통해 웹 서버에 전달되게 된다. 또한 전송된 데이터는 인터넷을 통해서 X-Hyper255B 임베디드 보드에 나타나게 된다.

그림 4는 모듈의 데이트 흐름 도를 보여준다. 1. 웹 서버에서 싱크노드에 쿼리를 요청하고, 2. 싱크노드는 센서 노드가 탑재된 라인트레이서에 쿼리를 보내게 된다. 3. 라인트레이서는 센싱 필드에서 데이터를 감지하게 되고 4. 감지된 데이터를 수집해서 싱크 노트에 보내게 된다. 5. 웹 서버는 결과를 수신해서 저장하고 6, 7. X-Hyper255B 임베디드 보드에서는 사용자가 데이터 수신을 원하면 보여주게 된다.

3.2. 라인트레이서

본 논문에서 구현한 라인트레이서는 크게 센싱부, 탑재 부, 모터 부, Ez-RoboMaster 부로 구성되어 있다. 센서파트는 라인을 감지해서 주행을 하기 위해서 중요한 부분을 맡고 있고, 탑재 부는 센서 노드를 탑재하기 위해 만든 레고 브릭들을 말한다. 모터 부는 바퀴를 움직이기 위해서 사용되는 두 개의 모터로 구성되어 있고, Ez-RoboMaster 부는 Atmega128 마이크로프로세서가 내장된 보드이다. Ez-RoboMaster는 시리얼 포트를 통해서 프로그래밍되어 컴파일 된 바이너리 파일을 다운로드 해서 각 레고 브릭과 모터의 센서를 제어하는데 사용된다. 즉 라인트레이서는 센서 노드를 탑재하여 정해진 경로를 이동하면서 요청되는 데이터를 수집하는 역할을 하게 된다.

표 1. 라인트레이서 알고리즘

```
In a Line Tracer that load node;
SysInit()
initialization sensor1, sensor2 by 0,
LgmSetPow,
while Sw1GetStat
LgmSetDir, LgmRun
LgsReadType sensor1,
DisplaySensor
If sensor1's value is bigger than line perception value
LgmSetDir Port_M1 backward, Port_M2 forward
else move
LgsReadType sensor2,
DisplaySensor
If sensor2's value is bigger than line perception value
LgmSetDir Port_M1 forward, Port_M2 backward
else move
```

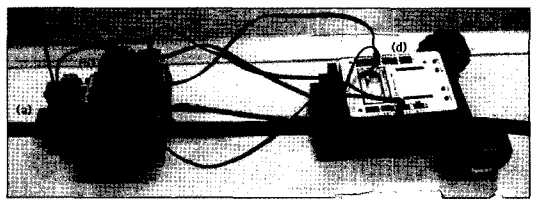

그림 5. 라인트레이서

표 1은 본 논문에서 사용된 라인트레이서의 알고리즘을 보여준다. 라인트레이서가 움직이게 되면 sensor1과 sensor2의 값을 초기화 시키고, 서로 다른 센서를 통해서 읽어 들인 라인의 값을 미리 정의한 값과 비교하게 된다. 만약 미리 정의한 값보다 sensor1의 값이 크다면 왼쪽 모터를 역회전 시키고, 오른쪽 모터를 정 회전 시켜서 라인트레이서를 왼쪽으로 이동시키게 된다. 반대로 sensor2의 값이 크다면 왼쪽 모터를 정 회전 시키고, 오른쪽 모터를 역회전 시켜서 라인트레이서를 오른쪽으로 이동시키게 된다. 그림 5는 본 논문에서 사용된 라이트레이서를 보여준다. (a)가 sensor1, sensor2이고 (b)는 Micaz 모트와 MTS310 센서보드로 구성된 센서노드이고, (c)는 모터, (d)는 Ez- RoboMaster를 보여준다.

3.3. TinyDB 활용

TinyDB는 TinyOS 센서 네트워크로부터 정보를 추출하기 위한 쿼리 프로세싱 시스템으로 ACQP(acquisitional query processing)이다[10]. ACQP은 센서를 새로운 장소에 위치시키거나, 센서의 배터리 교체 등과 같은 시간적, 물리적 비용에 대한 소비를 감소시키기 위한 목적을 가지고 있고, 스마트 센서가 언제, 어느 장소에서, 얼마나 자주 데이터가 쿼리 프로세싱 오퍼레이터에 습득되고 전달되는가를 제어하는 가에 대한 개념을 가지고 있는데, TinyDB는 전통적인 쿼리 프로세서의 특징과 이러한 ACQP의 특징을 가지고 있다[12].

그림 6은 TinyDB GUI 구조를 보여준다[15]. TinyDB GUI 구조는 TinyDBMain 함수에서 시작한다. 쿼리가 시작되면, TinyDBMain에서 설정된

환경을 기반으로 시리얼 포트를 열고, tinyDB : SensorQueryer 에서 쿼리를 읽고 모트와 통신을 하게 된다. QueryResult는 네트워크에서 byte단위로 쿼리 결과를 받은 다음, 그 결과를 수집해서, 많은 유틸리티에 제공하게 된다[12]. 본 논문에서는 이동하는 위치의 데이터를 나타내기 위해서 QueryResult에서 얻어진 데이터를 통해서

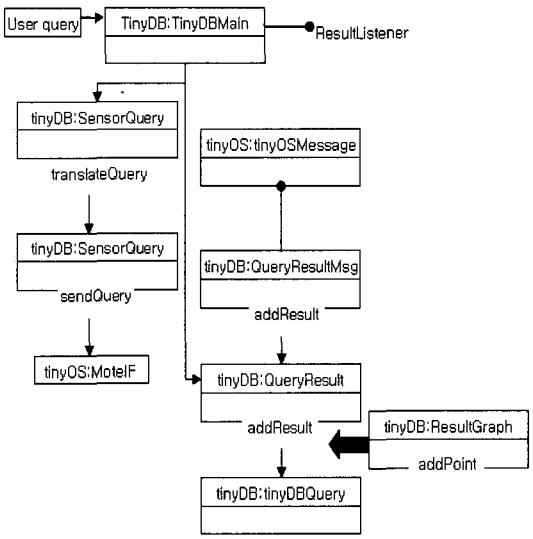

그림 6. TinyDB GUI 구조

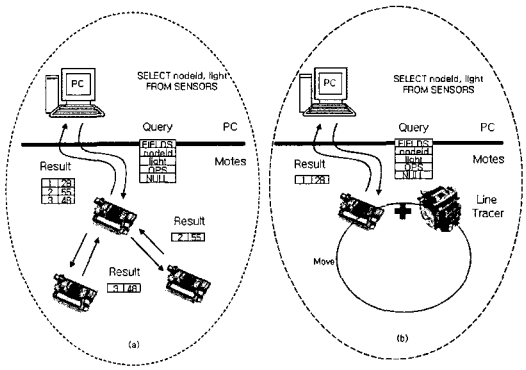

그림 7. TinyDB 데이터 처리 모델 (a) 기존 TinyDB 데이터 처리 모델 (b) TinyDB 이동 센서 데이터 처리 모델

ResultGraph부분을 추가하여 받아진 데이터 값을 주기적으로 나누어 표현하였다.

그림 7(a)는 기존의 TinyDB 데이터 처리 모델을 보여준다. PC에서 생성된 쿼리는 네트워크에 최적화된 쿼리를 보내주게 되고, 전파된다. 그 다음에 전달된 쿼리는 라우팅 트리를 기반으로 이웃 노드의 데이터를 습득하고 PC에 결과를 전달하게 된다. 본 논문에서는 노드의 움직이는 위치에서 데이터를 습득하기 위해서 TinyDB 네트워크 기본구조에서 이웃 노드를 제외하고 라인트레이서에 노드를 탑재하였다. 따라서 쿼리는 PC에서 생성된 후, 최적화된 쿼리가 탑재된 센서 노드에 전달되고, 습득된 데이터는 바로 PC에 전달되게 된다. 그리고 라인 트레이서가 이동하면서 센서 노드들이 위치했던 경로를 이동하면서 습득된 데이터를 지속적으로 PC에 전달하게 된다.

3.4. 웹 서버 구현 및 임베디드 보드에서의 데이터 처리

그림 8은 본 논문에서 사용된 웹 서버에서의 데이터 흐름을 보여준다. 본 논문에서는 아파치 웹 서버와 톰캣을 사용하였다[16]. 싱크노드에 요청된 쿼리의 결과로 얻어진 데이터는 TinyDB 애플리케이션에 의해 PostgreSQL에 저장되고 사용 된다. 또한 저장된 데이터를 사용하기 위해서 Java Server Page를 구현해서 PostgreSQL에서의 데이터를 임베디드 보드에서 사용 가능하게 처리하였다. 그림 9는 X-Hyper255B 임베디드 보드에서의 "SELECT * FROM TABLE"의 쿼리에 대한 데이터 처리 결과를 보여준다. 즉 보드에서 쿼리를 실행하게 되면 그림 8의 흐름도 처럼 Java Server Page에 접근해서 PostgreSQL에 저장된 데이터를 출력하게 된다.

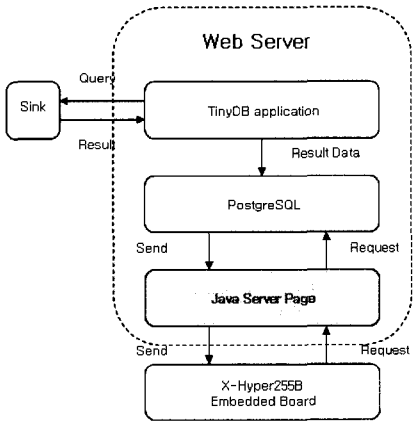

그림 8. 웹 서버에서의 데이터 흐름도

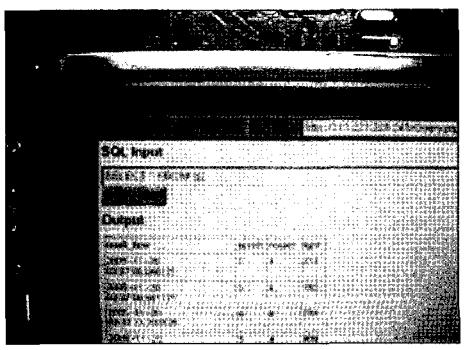

그림 9. X-Hyper255B에서의 쿼리 데이터 처리

Ⅳ. 데이터 측정 및 분석

이동 센서의 데이터를 처리하기 위해서 3.4절에서는 PostgreSQL을 이용하여 데이터를 웹 서버에 저장하고 X-Hyper255보드에 나타내었다. 4장에서는 이러한 데이터를 정지 센서 노드와 비교하여 처리하고 분석하였다. 먼저 하나의 센서 노드가 탑재된 라인 트레이서가 지나가는 위치에 일정한 간격으로 3개의 센서 노드를 고정 시켜놓고, 쿼리를 입력한 후 데이터 변화를 측정하였다. 변화를 측정하기 위해서, 하드웨어는 TinyDB가 포팅 된 다섯 개의 Micaz 모트와 센싱을 위한 MTS310 센서 보드, 그리고 웹 서버와 시리얼 통신을 하는 MIB510 프로그래밍 보드가 사용되었고, 3장에서 설계된 라인트레이서와 결과를 표현하기 위한 X-Hyper255B 임베디드 보드가 사용되었다. 소프트웨어는 웹 서버에서 Window XP 기반으로 Cygwin-1.5.12버전을 사용하였고, 데이터 저장을 위한 PostgreSQL이 사용되었다. 또한 X-Hyper255B 보드에서는 Linux Kernel 2.4.20을 사용하였다.

그림 10은 이동 경로 상에 위치한 정지 노드 3개의 데이터 변화를 보여준다. 그림 10에서 그림 왼쪽의 Epoch는 센싱 되는 주기의 수를 나타내고, nodeid는 노드 번호를 그리고 light는 빛의 세기를 0(어두움)에서 1000(밝음)으로 했을 때의 수치를 나타낸다. 왼쪽 표에 따른 오른쪽 그래프에서Time(s)은 0초부터 100초까지만 계산된 것이다. Time(s)에 따른 Light의 변화는 각 위치에서의 Light 세기를 나타내기 때문에, 임의로 변화를 주지 않는 한 일정하게 유지되는 것을 볼 수 있다. 본 논문에서는 고

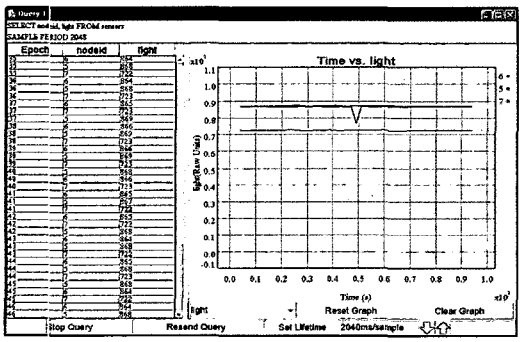

그림 10. 정지노드의 데이터 변화

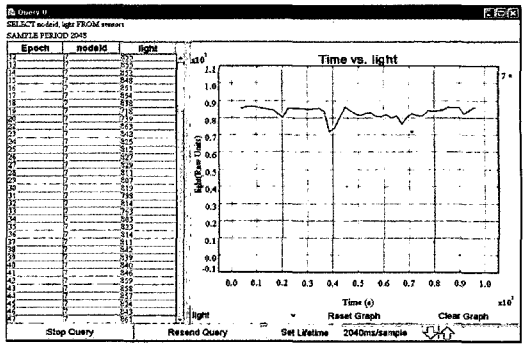

그림 11. 이동노드의 데이터 변화

의적으로 노드 7의 위치를 어둡게 해서 노드 5, 6과 차별화를 두었다. 그 결과 노드 5, 6의 데이터는 평균 880을 나타내고, 노드 7의 데이터는 평균 720을 나타내게 된다.

그림 11은 이동 노드의 데이터 변화를 보여준다. 그림 10과는 다르게 그림 11은 라인트레이서와 함께 센서 노드가 움직이기 때문에 주기 별로 각 위치에서의 데이터가 각각 다르게 나타난다.

그림 12는 이동 노드와 정지 노드의 데이터 변화를 200초 동안 동시에 측정한 결과를 그래프로 나타낸 것이다. 그리 12에서 노드 4, 5, 6은 정지 노드, 노드 7은 이동 노드를 나타낸다. 그래프에서 1Cycle은 라인 트레이서가 한 바퀴 움직일 때 경과한 시간까지를 나타낸 것으로 총 3바퀴를 움직인 것이다. 그리고 그래프가 겹쳐지는 지점에서 노드 7이 이동하면서 노드 5를 지나는 위치를 예상할 수 있다. 본 논문에서 사용된 환경을 기준으로 보면 이동 노드 7은 정지 노드와 함께 측정할 경우 측정

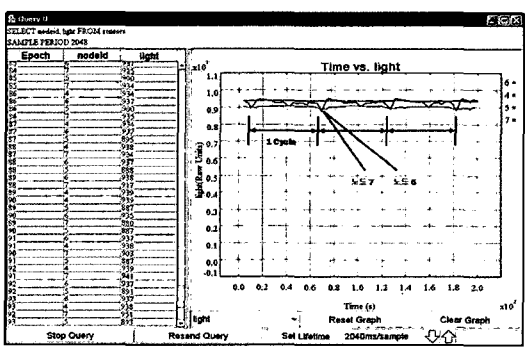

그림 12. 이동노드와 정지노드의 데이터변화

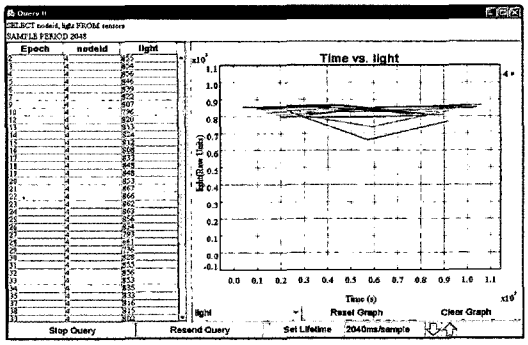

그림 13. 이동노드의 데이터 처리 후 변화

된 데이터가 겹쳐지는 구간의 예상을 통해서 데이터 변화를 예측할 수 있지만, 그림 11처럼 이동 노드 하나로만 측정할 경우 정지해 있는 노드처럼 데이터 변화를 측정할 수 없다. 따라서 이동 노드의 경우 그 반복 측정이 가능한 구간에서의 데이터 처리가 필요하게 된다. 그림 13은 이동 노드 한 개로 1Cycle을 주행한 후 측정한 데이터를 그래프로 보여준다. 하나의 이동 노드로 측정하였기 때문에 nodeid는 4로 고정이 되어있지만 이동 노드의 주기를 2048ms로 설정하고, 1Cycle에 걸리는 시간을 약 30초로 측정하여 주기 별로 데이터를 나타내었다. 따라서 그래프의 수는 총 12개로 표시된 것이고, 그림 10에서 사용된 3개의 정지 노드의 그래프와 그림 13에서 사용된 12개의 그래프는 비교될 수 있다. 즉 이동 노드 한 개로 12개의 정지 노드로 나타낼 수 있는 것이다. 하지만 라인트레이서의 일정하지 않은 이동과 모터 속도 등의 차이 문제로 정확한 위치에서 센싱 되는 데이터의 처리가 그림 10, 그림 11과 같이 정확하게 처리되기는 어렵다는 문제를 가지고 있다.

V. 결 론

본 논문에서는 쿼리 프로세싱 시스템인 TinyDB를 활용해서 정지해 있는 센서 노드를 정해진 이동 경로를 주행할 수 있는 라인트레이서에 탑재해서 이동 경로 주변의 데이터를 수집하여 TinyDB를 활용해서 처리 하였고, 수집된 데이터를 저장하여 웹 서버에서 구현하였다. 또한 사용자의 편의를 위해 임베디드 보드에 웹 브라우저를 만들어 터치 식으로 데이터를 처리 하도록 처리하였다.

이동 노드의 데이터 처리 후 변화를 측정하기 위해서 구현한 이동 센서 데이터 처리 모듈은 이동함으로써 정지해 있는 센서 노드들의 데이터를 일정한 간격으로 수집할 수 있었고, 특정한 위치에서의 데이터만 측정 할 수 있는 정지노드의 데이터와는 달리 이동 경로의 데이터를 수집해서 그래프로 전체적인 이동 경로의 데이터를 보다 정확하게 알 수 있지만, 이동 경로가 정해지지 않은 센서의 데이터는 일정하지 않은 데이터처리 등의 문제를 발생 시킬 수 있고, 이동 중에 정확한 위치에서의 데이터 처리 등의 여러 문제점을 야기 할 수 있다. 따라서 수집된 데이터를 저장하고 처리하는데 있어 이동 경로에서 정확한 데이터 측정과 처리등과 정지노드에서의 데이터와 비교하는데 보다 정밀하고, 효율적인 접근 방법이 반영되어야 하는 과제를 가지고 있고, 또한 웹에서의 데이터 구현으로 인한 임베디드 보드에서의 데이터처리로 보다 편리하게 데이터에 접근할 수 있지만, 인터넷 보안에 관한 문제를 해결해야 하는 과제를 가지고 있다.

참 고 문 헌

[1] Ian F. Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci Georgia Institute of Technology, "A Survey on Sensor networks", *IEEE Communication Magazine*, pp.102-114, August 2002.

[2] I.F Akyildiz, W.Su, Y. Sankrasubramaniam, E. Cayirci, "Wireless sensor networks: a survey", *Computer Networks* 38, pp.393-422, 2002.

[3] Tinyos web site. Available in: http://www.tinyos.net

[4] Philip Levis, Nelson Lee, Matt Welsh, and David Culler, "TOSSIM:Accurate and Scalable Simulation of Entire TinyOS Applications", *SenSys'03,* November 5-7, 2003.

[5] Samuel Madden, Joe Hellerstein, and Wei Hong, "TinyDB: In-Network Query Processing in TinyOS", *IRB-TR-02-014*, October, 2002.

[6] I.F.Akyildiz,W.Su, Y.Sankarasubrarsubramaniam, E. Cayircy, "Wireless sensor networks: a survey, Computer Networks", *The International Journal of Computer and Telecommunications Networking,* v.38n.4, pp.393-422, 15 March 2002.

[7] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister. "System Architecture Directions for Networked

Sensors", *In the Ninth International Conference on Architecture Support for Programming Languages and Operating Systems*, 2000.

[8] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler. "The nesC Language: A Holistic Approach to Networked Embedded Systems", *In Proceedings of Programming Language Design and Implementation(PLDI)*, June 2003.

[9] David Gay, Phil Levis, David Culler, "Software design patterns for TinyOS", ACM SIGPLAN Notices , *Proceedings of the 2005 ACM SIGPLAN/SIGBED conference on Languages, compilers, and tools for embedded systems* LCTES'05,Volume 40 Issue 7. June 2005.

[10] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, "The Design of an Acquisitional Query Processor for Sensor Networks," *ACM SIGMOD*, June 9-12 2003.

[11] TinDB web site. Available in: http://berkeley.intel-research.net/tinydb

[12] ChangWha Kim, SungDae SaGong, MungWoo Lee, "Robotice that enjoy by LEGO" EASYTECH

[13] EASYTECH site http://www.ezlab.com

[14] Lina Al-Jadir "Sensor Network Databases- An Introduction", *Advanced Research Topics inSummer* 2004-2005.

[15] Jing Jing Zhu, "SEMENTIC ROUTING IN WIRELESS SENSOR NETWORKS", *A thesis submitted in partial fulfillment of the requirements for the degree of Computer Science Rochester Institute of Technology* 2005.

[16] Web Services Project Apache. Available in: http://ws.apache.org

이 상 훈 (Sang-hoon Lee) 정회원

2004년 2월 수원대학교 컴퓨터학과 졸업

2006년 2월 수원대학교 컴퓨터학과 석사

2006년 3월~현재 수원대학교 컴퓨터학과 박사과정

<관심분야> 무선 센서 네트워크, 임베디드 리눅스

문 승 진 (Seung-jin Moon) 정회원

1986년 미국 텍사스대학교 컴퓨터학과 졸업

1991년 미국 플로리다 주립대학교 컴퓨터학과 석사

1997년 미국 플로리다 주립대학교 컴퓨터학과 박사

1997년~현재 수원대학교 컴퓨터학과 부교수

<관심분야> 실시간 멀티미디어 리눅스, 실시간 모바일 데이터베이스, 실시간 임베디드 시스템, 실시간 센서 네트워크 시스템