무선 센서 네트워크에서의 자기센서기반 이동경로 추적과 데이터 처리 모듈

김 홍 규[†] · 문 승 진[††]

요 약

센서네트워크 환경에서 이동객체의 위치추적방법은 추적방법론에 따라 데이터의 수신과 위치추적의 정확성에 문제가 있었으며, 이러한 위치 데이터는 오로지 위치 추적만을 위한 데이터이었기에 주변의 환경 데이터를 활용하여 상황에 맞는 정보판단 또는 모니터링에 한계가 있었다. 이러한 한계를 극복하기 위한 방법으로 상황인식(context aware)을 종종 활용하는데, 본 논문은 자기센서를 기반으로 측정된 자성체(magnetic line tracer)의 자성데이터 위치 추적 및 제어에 관한 데이터처리에 상황인식을 활용한 방법을 제안하였다. 제안된 방법은 센서네트워크 노드의 자기센서의 자성데이터 측정과 주변 환경정보 데이터를 이용하였고, 이러한 데이터를 상황인식 데이터베이스의 엔트리로 구성함으로써 추후 행동정보 추론 및 제어할 수 있다. 아울러 제안된 방법의 타당성을 검증하기 위하여 센서네트워크 환경을 만족하는 테스트베드를 구축하였으며, 이를 이용한 라인트레이서의 위치 추적과 경로탐색의 결과를 도출 하였으며, 도출된 데이터를 근거로 하여 주변 환경정보 데이터베이스 엔트리 및 상황인식에 따른 라인트레이서의 방향제어에 미치는 영향을 분석하였다.

키워드 : 무선 센서 네트워크, 이동경로 추적, 위치추적, 상황인식, 자성체

Route Tracking of Moving Magnetic Sensor Objects and Data Processing Module in a Wireless Sensor Network

Kim Hong Kyu[†] · Moon Seung Jin[††]

ABSTRACT

In sensor network processing environments, current location tracking methods have problems in accuracy on receiving the transmitted data and pinpointing the exact locations depending on the applied methods, and also have limitations on decision making and monitoring the situations because of the lack of considering context-awareness. In order to overcome such limitations, we proposed a method which utilized context-awareness in a data processing module which tracks a location of the magnetic object(Magnetic Line Tracer) and controlled introspection data based on magnetic sensor. Also, in order to prove its effectiveness we have built a wireless sensor network test-bed and conducted various location tracking experiments of line tracer using the data and resulted in processing of context-aware data. Using the new data, we have analyzed the effectiveness of the proposed method for locating the information database entries and for controlling the route of line tracer depending on context-awareness.

Key Words : Wireless Sensor Network, Object Route Tracking, Location Tracking, Context Aware, Magnetic Object

1. 서 론

MEMS(Micro Electro Mechanical System), 나노기술(nano technology) 등과 같은 초소형 마이크로 센서의 하드웨어 기술이 발전함에 따라 다양한 기능의 센서를 이용한 무선 센서 네트워크의 구축이 가능하게 되었다. 가장 일반적인 위치 추적을 위한 기술에는 삼각 측량법, 장면 분석법, 근접방법의 세 가지 방법이 있으며, 현재 개발되고 있는 위치 추적 시스템은 이 세 가지 방법을 사용하고 있다. 기존의 위치추적방법은 '새로운 기반 구조를 구축해야 하는가', '이미 구축되어 있는 기반 구조를 사용할 수 있는가'에 따라 두 부류로 나뉠 수 있다[1].

이러한 위치 추적 방법을 이용한 시스템이 개발되어 있음에도 불구하고 아직도 위치 추적에 대한 연구가 진행중인 이유는 최근의 위치 추적 시스템의 개발에 대한 초점이 특정 응용에 사용될 수 있는 적합한 위치 추적 시스템을 개발하고, 사용자의 위치 추적을 위해 위치 추적의 정확도와 시스템의 가격, 설치의 용이성, 실시간성, 주변 환경인식성 등을 고려하여 특정한 응용에 최적의 시스템을 활용하기 때문

† 준 회 원 : 수원대학교 컴퓨터학과 박사과정
†† 종신회원 : 수원대학교 컴퓨터학과 부교수
논문접수 : 2006년 6월 21일, 심사완료 : 2006년 11월 8일

이다. 이러한 특징들을 초소형 마이크로 센서 하드웨어 기술들이 포함된 무선 센서네트워크를 활용하여 현실 세계에서 발생하는 여러 이벤트를 감지하여 이를 네트워크를 통해 수집, 처리하여 위치기반 서비스(LBS; Location Based Service)의 다양한 콘텐츠로 개발한다면 사물(things)의 네트워크를 통한 통신, 지능화, 자율화가 되어 "언제", "어디서나", "어느 것"과도 통신이 가능한 환경을 기반으로 새로운 서비스를 제공하여 인간의 삶을 윤택하게 할 수 있을 것으로 보여진다.

본 논문에서는 사물의 네트워크를 통한 통신, 지능화, 자율화된 무선 센서 네트워크 기술에 기존의 특정 위치에 근접하여 자성체(magnetic; 이하, 라인트레이서)의 위치를 알아내는 근접접근 방법을 이용한 위치기반 서비스의 기술접목을 위한 모듈을 제안한다. 제안된 방법은 위치 추적을 위해 근접방법을 이용하였기 때문에 적은 프로세싱 파워를 가지고 실시간으로 라인트레이서의 위치 추적이 가능하다. 또한 라인트레이서의 위치만이 아닌 주변의 환경정보 수집을 기반으로 하여 특정한 위치에서 라인트레이서의 상황인식 정보적용과 제어가 가능하다.

본 논문에서 제안된 위치 추적 방법은 이동체의 위치 추적 방법과 주변 환경정보 모니터링에 최적화 되었고, MEMS의 기술에 부합된 센서들의 조합으로 이동체의 위치, 방위, 속력 등의 정보 외에 대형 범용 데이터베이스로 이동경로 추측 및 온도, 조도, 가속도, 음향 등의 정보를 알 수 있으며 이러한 정보를 토대로 몇 가지 행위 정보를 추론할 수 있다.

본 논문은 다음과 같이 구성된다. 2장에서 무선 센서 네트워크와 위치기반 기술, 그리고 실시간 컨트롤 시뮬레이터 모델에 대해 알아보고, 3장에서는 제안하는 시스템의 시나리오와 각 모듈별 구조에 관하여 설명하고, 4장에서는 제안된 시스템의 모듈별 구현을 기술한다. 5장에서는 구현된 시스템을 시뮬레이션 도구를 이용하여 실행 및 분석을, 6장에서는 본 논문의 결론과 앞으로 해결해야 할 과제들을 제시한다.

2. 관련 연구

2.1 무선 센서 네트워크(WSN; Wireless Sensor Network)

'어느 곳에서나 존재한다'라는 의미의 무선 센서네트워크는 시간과 장소에 구애받지 않고 생활 속에서 자연스럽고 편리하게 컴퓨터를 사용할 수 있는 환경을 의미한다. 즉, 컴퓨터가 도처에 편재하여 감지와 추적을 통해 장소나 시간에 따라 그 내용이 변화(context aware)하는 특화된 정보 서비스를 받을 수 있으며, 이러한 센서 네트워크는 통상적으로 특정 지역에 소형의 센서노드를 설치하여 주변 정보 또는 특정 목적의 정보를 획득하고, 베이스 스테이션(BS; Base Station)이 정보를 수집하여 이를 활용하기 위한 서비스 환경을 말한다[2]. 즉, 각 센서 노드가 특정 목적을 위해 필요한 주변정보를 센싱하고, 센싱된 정보를 센서노드간의 무선통신을 이용하여 특정 지점으로 자동화된 방식으로 전달함으로써 사용자가 센서필드 주변의 정보를 원격으로 수집하여 활용할 수 있다는 것이다. 센서 네트워크의 전통적인 개념은 무선의 센서필드 개념을 중심으로 불특정 공간에 배포된 센서로부터 수집된 정보를 일괄적으로 활용하는 기술이다.

2.2 위치기반 기술(LBS; Location Based Service)

가장 일반적인 위치 측위 방법으로는 물체간의 거리 차나 각도 또는 방위각을 측정하여 위치를 측정하는 삼각측량방법(triangulation)과 특정관점(vantage point)에서 보이는 풍경을 이용한 장면분석방법(scene analysis), 그리고 특정 위치에 근접하여 대상체를 알아내는 근접방법(proximity)이 있다[3]. 이러한 위치 측위 방법의 기술로는 위성통신(GPS; Global Positioning System), 이동통신(단말 기반, 하이브리드), 무선통신(적외선; diffuse-infrared, 초음파; ultrasonic wave, RF; Radio Frequency, UWB; Ultra Wideband, RFID), 영상인식(개인위치 추적기; person tracker) 방법이 있으나 신호 또는 전파의 수신가능지역, 신호의 감쇠, 태양광등의 주위 환경이나 잡음 및 반사, 굴절 등에 의해 정확도가 떨어지는 문제점이 있다[4].

또한 무선 센서네트워크 기술의 하나인 Ad-Hoc을 이용한 위치측위는 고정되지 않은 노드들의 위치를 스스로 찾아내며 다수의 노드들로 구성된 비교적 규모가 큰 센서 네트워크를 목적으로 사용된다. 하지만, 이러한 기술은 이동 Ad-Hoc 네트워크의 특성에 따라 그대로 적용하기가 어렵고, 특히 독립적이고 분산된 환경을 고려해야 하며 고정된 래퍼런스 노드의 지원 없이 정확하고 안정된 서비스를 기대하기가 용이치 않다.

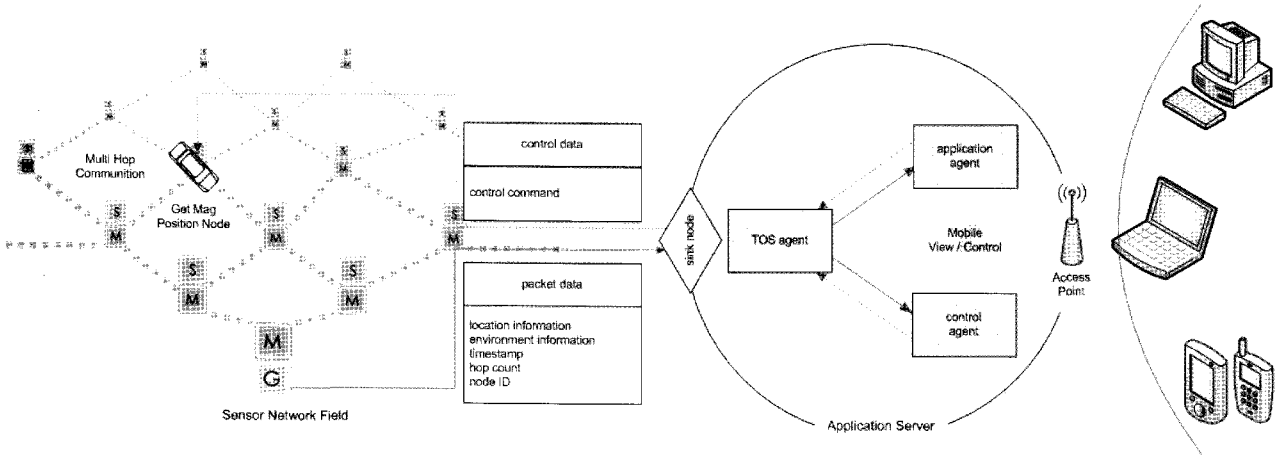
2.3 Real-Time Lego Control

80년대 초반부터 덴마크 레고사와 미국 M.I.T(메사추세츠 공과대학)가 공동으로 개발한 레고 브릭(brick)에 동력과 컴퓨터 센서를 결합하여 기초기계의 원리에서부터 구조역학, 에너지, 메카트로닉스, 컴퓨터 프로그램으로 제어되는 로보틱스와 같은 첨단 응용과학 분야를 자유롭게 설계할 수 있으며, 본 논문에서는 ez-Robomaster 컨트롤러를 이용하여 라인트레이서를 제작하여 자성체의 이동을 돕거나 제어하기 위하여 사용한다[5].

3. 자기센서 기반 모듈 설계

본 논문에서는 기존의 위치기반 기술과 무선 센서네트워크 기술을 바탕으로 시스템을 설계한다. 센서 네트워크 필드의 데이터 수집을 위한 센서노드는 무선통신을 위한 모트(MPR2400; MICAz)와 라인트레이서의 위치 및 주변 환경정보 수집을 위해 각종 센서가 탑재된 센서보드(MTS310)의 묶음을 말한다.

센서노드는 센서네트워크 필드에서의 라인트레이서 위치 추적을 위해 200㎳ 주기로 데이터를 수집한다. 수집된 데이터와 주변의 자연적인 자기장 수치를 비교하여 라인트레이서의 근접 방위를 계산한다. 이에 라인트레이서와 가장 근



(그림 1) 전체 시스템 흐름도

접한 위치의 센서노드는 위치데이터와 상황인식에 사용되는 데이터를 싱크노드로 전송한다. 싱크노드(sink node; MTS310, MIB510)는 어플리케이션 서버와 연결되어 있으며 센서노드들의 데이터를 어플리케이션 서버의 TOS agent에서 수신한다. 수신된 데이터는 application agent에서 패킷의 일관성 확인, 분리작업, 명령, 화면정보, 제어정보 등을 데이터베이스 서버에 저장한다. 이러한 데이터는 상황인식에 필요한 자료로 사용되고 상황인식에 따른 control agent의 control command를 이용하여 블루투스통신으로 라인트레이서를 제어할 수 있도록 구현하였다.

3.1 시스템 시나리오

기존 위치기반 기술과 무선 센서 네트워크 기술을 바탕으로 시스템을 설계하기 위하여 시스템의 시나리오를 다음과 같이 구성하였다. 시스템 시나리오의 전체적인 흐름은 (그림 1)과 같은 방법으로 실행된다.

① 센서네트워크 필드에 있는 각 센서노드들은 무선으로 구성되어 있으며 라인트레이서의 위치추적을 위해 200 ㎳ 주기로 데이터를 수집한다. 수집한 데이터는 센서노드에서 라인트레이서의 자기장 측정값과 평균값을 비교하여 라인트레이서의 접근을 추론한다.

② 라인트레이서의 근접위치를 감지한 센서노드는 패킷데이터를 생성한다.
- location information
- environment information
- timestamp
- hop Count
- node ID

③ 위 ②에서 생성된 패킷 데이터를 다른 센서노드로 전송한다. 패킷을 수신한 센서노드는 타임스탬프를 확인하여 데이터의 일관성을 유지하며 다시 주변의 센서노드들로 패킷을 전송한다.

④ 패킷이 센서네트워크 필드의 지정된 게이트웨이 센서노드에 도착하면, 다시 싱크노드로 패킷을 전송한다

⑤ 응용프로그램 서버의 싱크노드 TOS agent는 수신된 패킷을 application agent로 전송한다.

⑥ application agent에서 채널파싱을 통한 ②에서의 데이터로 분류 하며, 분류된 데이터는 데이터베이스에 저장한다.

⑦-A 파싱된 채널데이터의 node ID, location information, timestamp와 데이터베이스에 저장되어 있는 과거 location information, node id, timestamp를 가지고 이동경로 유추를 한다.

⑦-B 현재 라인트레이서가 위치한 node ID를 node index와 비교하여 track map에 점으로 표기 한다.

⑧ application agent의 데이터베이스 데이터와 현재 위치한 라인트레이서의 주변 환경정보를 배경으로 현재 라인트레이서의 상황인식을 판별한다.

⑨ 상황인식의 결과는 control agent로 전송하여 라인트레이서의 제어정보로 사용할 수 있으며, 블루투스로 전송하여 라인트레이서의 제어를 실행한다.

⑩ 응용프로그램 서버는 라인트레이서의 위치정보 또는 주변 환경정보의 상황인식상태 등의 정보를 사용자에게 서비스 할 수 있도록 도와준다.

⑪ 응용프로그램 서버로부터의 서비스신호 등을 무선 환경에서 PDA를 통하여 직접 제어하거나 위치 정보를 보여준다.[1)]

본 논문에서 구현한 시스템 모듈은 총 5개의 모듈로 센서노드에서 동작하는 sense agent와 env agent로 구성되어 있고, 응용프로그램 서버에서 동작하는 TOS agent와 application agent 그리고 control agent로 구성되어 있다. 센서노드에서 동작하는 sense agent는 200㎳ 주기로 라인트레이서의 근접탐지를 우선적으로 수행하고, env agent는 sense agent가 감지한 라인트레이서의 근접한 센서노드로부터 주변 환경정보를 수신하여 각 채널별로 데이터 수집하여 시나리오 ②와

1) 본 논문에서는 PDA Client를 구현하지 않았다.

같은 정보들로 패킷을 생성하여 응용프로그램 서버의 TOS agent로 송신한다. 응용프로그램 서버의 TOS agent는 싱크 노드에서 동작하게 되며 env agent의 데이터를 항상 감지하며 수신된 데이터를 application agent로 보낸다. application agent는 env agent로부터 수신한 데이터를 각 채널별로 라인트레이서의 위치정보, 주변 환경정보, Node ID, timestamp 등으로 분리하며 이러한 정보들을 기반으로 라인트레이서의 현재 위치 또는 이동경로정보 및 주변 환경정보를 바탕으로 라인트레이서의 상황에 따른 제어가 가능한 모듈인 control agent 로 구성되어 있다. 이는 각 시스템을 세분화된 모듈로 적용함으로써 시스템들 사이의 데이터 처리율을 높여 시스템 부하를 감소시키기 위해서이다.

3.2 센서노드 모듈

센서노드 모듈은 2개의 agent로 구성되어 있으며 (그림 2)와 같다. 센서노드의 초기 동작은 전원이 켜지는 시점에서 sense agent가 수행되며 라인트레이서의 위치 정보를 추적한다. 이러한 sense agent는 라인트레이서의 위치와 가장 근접한 센서노드에서 데이터의 변화폭이 변화 할 경우 env agent를 호출하여 주변 환경정보를 각 8개의 채널의 대입하여 센싱하게 되고 데이터 패킷의 마지막은 위치정보를 포함하여 데이터 패킷을 생성하게 된다. 따라서 env agent에서 라인트레이서의 위치 정보와 연관된 주변 환경정보를 하나의 패킷으로 생성하게 되며 생성된 데이터 패킷은 TOS agent로 송신한다.

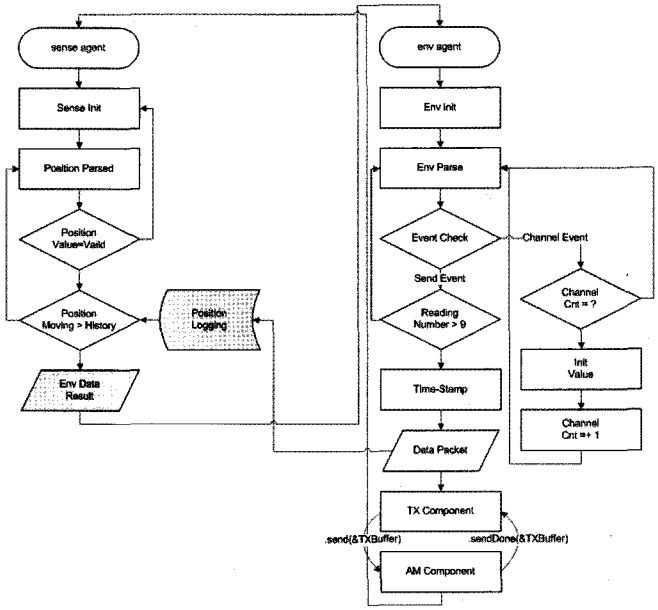
3.2.1 sense agent

(그림 2)에서의 sense agent는 센서노드의 초기 동작을 수행하게 되며 라인트레이서의 위치감지 센서인 자기센서의 초기화를 시작으로 동작하게 된다. 센서 초기화가 종료 되면 Position Parsed에서 라인트레이서의 근접정보 감지를 위한 데이터 수집을 수행하며 수집된 데이터를 위치 값과 비교하여 재 감지를 하거나 다음 세션으로 넘기는 작업을 수행한다. 따라서 Sense Init와 Position Value=Valid는 200㎳를 주기로 수행한다. 라인트레이서의 위치 정보가 이전의 위치정보와 같으면 Position Logging에서 History정보를 비교하여 값이 크다면 새로운 위치로 확인하여 env agent를 호출한다. Position Logging은 sense agent에서 해당 센서노드가 감지한 과거 2회의 정보를 말하며 이는 라인트레이서의 이동성 확인을 위하여 사용된다.

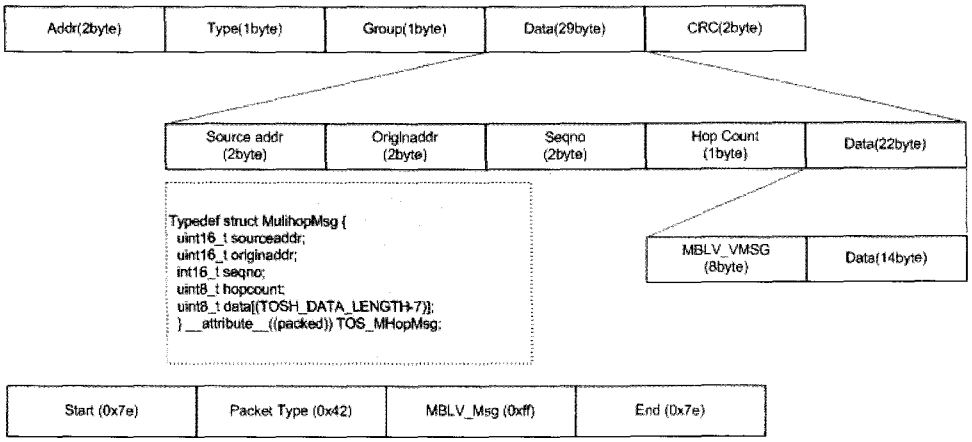
- Sense Init : 센서노드의 시스템을 초기화함으로써 센서노드의 데이터 수집 컴포넌트를 호출 및 수집된 데이터의 전송 완료, 데이터수집 버퍼를 삭제한다.
- Position Packet : Position Parsed, Position Value=Valid, Position Moving>History를 말하며, 라인트레이서의 근접정보 데이터를 수집하여 이전에 sense agent에서 수집한 근접정보 데이터와 비교하여 재 감지를 하거나 다음세션으로 넘긴다. 또한 이전 2 step 정도의 과거 데이터를 비교 하여 라인트레이서의 이동성을 측정하게 되며 라인트레이서의 이동이 생겼을 경우에만 env agent를 호출하게 된다.

3.2.2 env agent

env agent는 sense agent의 호출로 환경 센서(조도, 온도, 가속도, 소리)를 초기화로 시작하고 센서의 초기화 이후 환경 센서를 이용한 라인트레이서의 주변 환경정보를 센싱한다. Env Parse는 한 번에 하나의 환경을 센싱하며 Event Check에서 채널 값에 대응시키고 Channel Cnt를 증가시켜 하나에 데이터 패킷에 4가지의 환경 데이터를 2개씩 저장한다.



(그림 2) 센서노드 Agent 모듈



(그림 3) 생성된 데이터 패킷 구조

생성된 환경정보에 timestamp를 추가하여 환경정보 데이터 패킷을 생성하고 sense agent가 다음 자성체의 위치 추적을 위해 Position Logging에 저장하고 동시에 TX, RX Component에서 송수신 한다. 이렇게 생성된 데이터 패킷은 (그림 3)과 같다.

- Env Init : env agent가 환경데이터 및 타임스탬프를 위한 버퍼사용을 초기화 하며, 모니터링 데이터 수집을 알리기 위해 사용된다.
- Env Parse : 수신된 자기장 데이터에 각 환경정보 데이터의 신호인 Channel, 환경정보 데이터, 타임스탬프를 하나의 패킷으로 만들며, 환경정보 데이터는 최대 9개 이하의 데이터로 이루어진다. 타임스탬프는 환경정보 데이터의 수집이 완료된 후에 끝에 삽입함으로써 데이터 패킷이 완성되며, 이 데이터 패킷은 sense agent가 다음 자성체의 위치 추적을 위해 Position Logging에 저장 및 env agent의 TX, RS Component에서 송·수신 한다.
- Time-Stamp : 본 논문에서 타임스탬프는 데이터의 멀티 홉 전송 및 자성체의 위치 데이터 수신 시간을 나타내기 위해 사용되었다. 따라서 데이터의 수신은 곧 자성체의 위치 데이터 수신을 뜻하며 이는 자성체의 수신 시간이 포함된 타임스탬프이다.

3.3 서버 응용프로그램 모듈

서버 응용프로그램 모듈은 3개의 agent로 구성되어 있으며 (그림 4)와 같다. TOS agent는 env agent의 데이터를 수신하여 application agent를 호출하기 위해 사용된다. application agent는 사용자 응용 프로그램과 라인트레이서의 위치와 주변 환경정보 또는 이동경로 추적 데이터의 채널 파싱을 하며 이러한 정보는 상황인식 데이터에 사용되며 라인트레이서를 제어 할 수 있는 제어 데이터를 생성하여 control agent를 호출한다. control agent는 application agent의 명령을 기계어로 변환하게 되며 변환된 데이터는 블루투스를 통하여 라인트레이서에 전송한다.

3.3.1 TOS agent

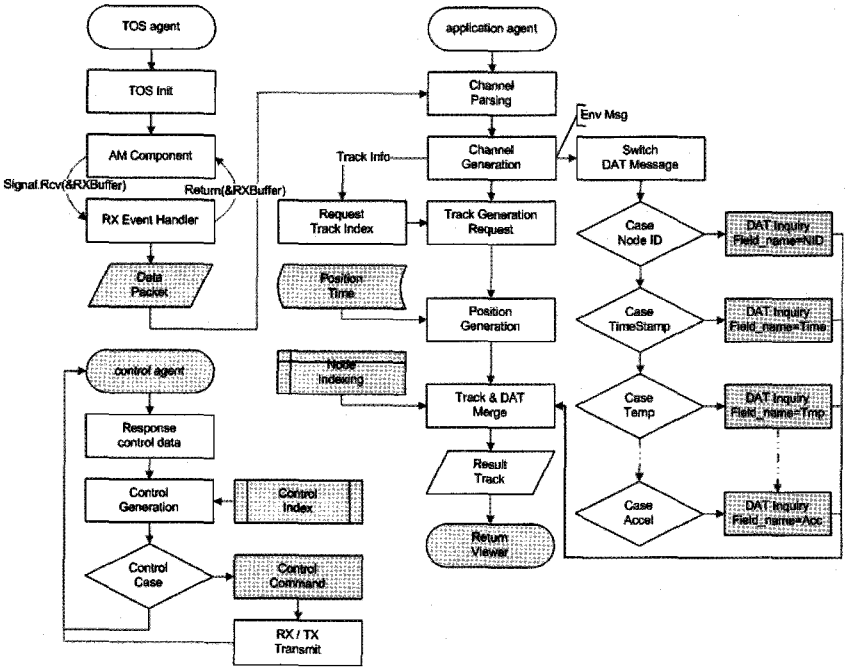
TOS agent는 센서노드로부터 데이터 패킷을 수신하여 application agent로 데이터 패킷를 전송하는 수신 부 역할을 한다.

- TOS Init : TOS Agent의 데이터 송신 함수 및 이전 데이터 수신을 위해 사용되었던 버퍼를 초기화하며 AM Component로부터 데이터 신호 수신 및 RX Event Handler로 수신 완료 신호를 리턴 하는 구조로 되어 있다.

3.3.2 application agent

application agent는 TOS agent로부터 데이터 패킷을 받아 데이터를 파싱하거나, 파싱된 데이터를 상황인식데이터 처리를 위한 데이터베이스에 저장 및 검색 한다. 데이터베이스에 저장된 데이터로 라인트레이서의 Track Index와 데이터 패킷의 값을 비교하여 위치추적 및 동작을 제어할 수 있고 이러한 위치데이터 및 동작제어 데이터를 사용자에게 시각적으로 표현한다.

- Channel Parsing : TOS agent로부터 수신된 데이터는 index를 가지게 되며, 수많은 데이터를 하나의 패킷으로 구분 및 분할하게 된다. 이러한 패킷은 다시 여러 채널로 구성되어 하나의 데이터 구조를 가지게 되므로 위치 정보, 환경정보 데이터로 구분된다.
- Channel Generation : 채널별로 구분된 데이터는 Event Msg를 생성하게 되며, 채널 1번과 2번 데이터는 마그네틱 센서의 자기장 데이터(X축, Y축)를 나타내고 나머지 채널들은 환경정보 데이터를 가리키게 된다. 즉, 채널 1번, 2번 데이터는 Track Info데이터로 Track Index와 비교하여 현재의 라인트레이서의 위치 정보를 말해 주며, 나머지 채널데이터들은 Env Msg를 생성하여 Switch DAT Message를 호출한다.
- Switch DAT Message : 채널 데이터를 기준으로 센서



(그림 4) 서버 응용프로그램 agent 모듈

노드의 Node ID, timestamp, Temp, Accel 등의 여러 환경정보 데이터로 분류되어 각 필드에 저장하게 되며, 저장된 데이터는 추후 타임스탬프와 Node ID데이터를 기준으로 하여 라인트레이서의 이동경로 추적에도 사용된다.

- Track Generation Request : Track Info의 데이터와 미리 정의한 Track Index를 서로 비교하여 수신된 Track Info 데이터가 갖는 센서노드의 위치를 라인트레이서의 위치로 치환하며 Position Generation과 함께 Position Time을 불러 자성체의 이동을 최적화한다.
- Track/DAT Merge : 라인트레이서의 위치 정보와 저장된 환경 정보, 그리고 최종 Node Indexing정보를 조합하여 하나의 Track데이터로 만들며, 라인트레이서의 위치 정보와 주변 환경정보를 바탕으로 라인트레이서의 동작제어 패킷을 생성하거나 건너뛰고, 최종적으로는 응용 프로그램 Viewer로 각종 데이터를 확인할 수 있는 프로세스이다.

3.3.3 control agent

control agent는 application agent에서 상황인식정보를 확인하여 상황인식정보 데이터가 존재할 경우 수행되는 agent로써 라인트레이서로 블루투스를 통하여 제어정보를 송신한다.

- Response Application Data : Server Agent로부터 제어 데이터의 존재여부를 판별하게 되며 데이터가 존재한다면 Control Generation을 수행한다.
- Control Generation : 미리 정의된 Control Index에 따라 수신된 제어 명령을 판단함으로써 라인트레이서의 제어를 Control Command에서 제어 정보를 패킷으로 생성하여 RX/TX Transmit를 호출하여 전송하고 전송이 완료된 후에는 RX/TX Transmit의 신호와 함께 application agent의 데이터 송신 대기 상태로 이동한다.

4. 자기센서 기반 모듈 구현

4.1 센서노드 모듈

센서네트워크 모듈은 일전 시간동안 주기적으로 데이터를 수집하며, 수집된 데이터는 정해진 규칙에 맞을 경우 전송함수를 호출함으로써 끝나게 된다. 따라서 본 논문에서의 데이터 수집은 센서노드 응용프로그램 상에서 약 200㎳ 주기로 위치 측정 데이터인 자기장과 함께 주변 환경정보 수집으로 온도, 조도, 가속도, 소음 등을 측정한다. 즉, 센서노드 모듈은 <표 1>과 같이 데이터의 수집, 센서노드 동기화, 데이터의 전송과 같이 세 부분으로 나누어진다.

센서노드들은 <표 1>과 같이 센서를 통한 데이터의 수집, 센서노드의 동기화, 데이터의 전송으로 기본 동작한다. 데이터의 수집은 각각의 센서 컴포넌트들로부터 총 8개의 raw데이터와 채널번호를 삽입한다. 이러한 데이터에 기본 멀티 홉 전송에서 사용되는 timestmp를 이용하여 라인트레이서의 경로추적 및 데이터 수집 시점을 확인할 수 있다.

수집된 데이터와 현재 데이터를 수집하고 있는 센서노드의 정보는 하나의 패킷으로 생성된다. 이러한 데이터 패킷의 구조는 앞서 설명한 (그림 3)과 같으며 TinyOS의 "broadcast" 라이브러리 패킷 구조에 추가적으로 7byte 크기의 헤더를 사용함으로써 데이터를 저장할 수 있는 공간이 7byte 줄었다.

〈표 1〉 센서노드 모듈구분

구 분	내 용

데이터 수집	asyanc event result_t MagX.dataReady(uint16_t data){
	struct MBLVMsg *pack;
	if(pack = (struct MBLVMsg *) call Send.getBuffer(&msg,&len)){
	atomic{
	pack->datapacketReadingNumber++] = data;
	pcak->channel = 1;
	readingNumber++;
	if(packetReadingNumber == BUFFER_SIZE){
	post dataTask();
	}
	…중략…
	}

센서노드 동기화	static void initialize(){
	iFwdBufHead = iFwdBufTail = 0;
	BcastSeqno = 0;
	Time_Stamp = 0; }
	static bool newBcast(uint64_t proposed){
	if((proposed - Time_Stamp) > 0){
	Time_Stamp = proposed;
	…중략…
	}

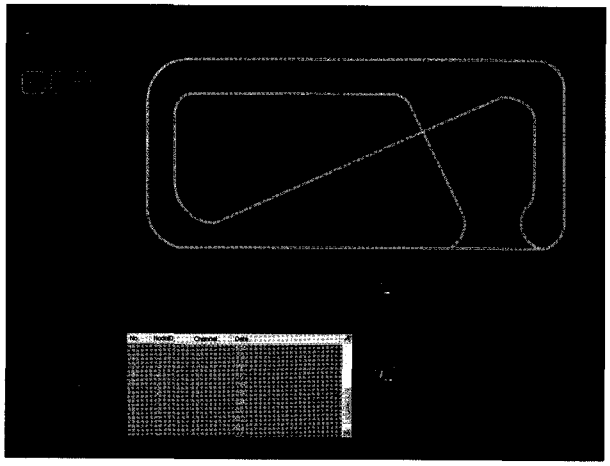
데이터 전송	task void dataTask(){
	struct MBLVMsg *pack;
	if(pack = (struct MBLVMsg *)call Send.getBuffer(&msg,&Len)){
	atomic{
	packetReadingNumber = 0;
	pack->lastSampleNumber = readingNumber; }
	pack->soutceMoteID = TOS_LOCAL_ADDRESS;
	if((call Send.send(&msg.sizeof(struct MBLVMsg))) == SUCCESS){
	…중략…
	}

생성된 msg 패킷의 경우 기존 6byte의 "readingnumber"의 크기를 2byte에서 4byte로 늘려서 총 8byte로 만들고 payload는 총 29byte에서 multihopMsg header크기 7byte와 생성된 msg header크기 8byte를 뺀 14byte가 되며 생성된 msg에 14byte 즉, 7개의 raw 센서데이터 값을 삽입할 수 있다. 이러한 데이터 패킷은 이웃한 다른 센서노드로 데이터를 송신하며 데이터의 전송은 TinyOS의 멀티 홉 통신에 따라 진행된다.

4.2 서버 응용프로그램 모듈

서버 응용프로그램 모듈은 시리얼 통신과 블루투스 통신을 지원해야 한다. 센서노드의 데이터를 서버 응용프로그램에서 확인할 수 있도록 싱크노드와 시리얼 케이블로 연결되어 라인트레이서의 위치 및 상황인식에 필요한 주변 환경정보를 수신한다. 서버 응용프로그램에서 상황인식에 대응되는 라인트레이서 제어 명령은 블루투스를 이용하여 전송한다. 따라서 서버응용프로그램 모듈은 통신의 연결 설정, 데이터 처리, 연결제어, 화면처리 부분으로 나눌 수 있다.

이에 각 통신 쓰레드는 패킷 데이터를 수신하거나 송신하게 되며 패킷 데이터는 byte코드로 되어 있으며 판독할 수 있도록 패킷을 하나씩 나눠서 처리해야 한다. 즉, 수신되는 데이터는 순서대로 들어오지만 들어오게 된 데이터를 나누어서 센서노드번호, 채널번호, 데이터 등으로 분리해야 한다. 이러한 데이터 분리는 패킷에 따라서 시작점 "0x7e", "0x42", "0xff"를 찾아서 데이터를 분할하지만 시리얼 통신은 시리얼 통신만의 새로운 형식과 프로토콜을 사용하기 때문에 고유의 framing을 하게 된다. framing을 하다 보니 byte stuffing을 하게 되고 또 유연한 통신을 위해서 ACK신호를 주고받는다. 즉, "0x7e"는 패킷의 시작과 끝을 나타내고 "0x42"는 시리얼 통신의 flag이며 "7d5e"는 "7e"를 스태핑 처리한 것이 된다. 이렇게 처리된 데이터는 화면에 보이며 (그림 5)는 다중 센서노드 상태에서 라인트레이서 두 대를 이용한 위치 및 상태 제어를 보여주는 화면이다.



(그림 5) 서버응용프로그램 / 클라이언트 응용프로그램

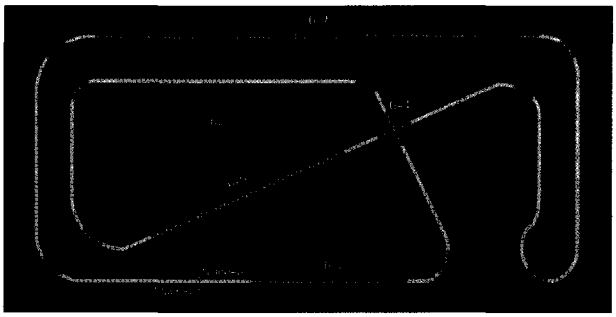
4.3 모듈 데이터 활용

본 논문에서 raw데이터는 센서노드에서 수집하며, 수집된 데이터는 서버응용프로그램에서 여러 가지 상태 정보로 활용할 수 있는 데이터로 변환된다. 이에 활용할 수 있는 정보의 변환으로는 센서노드의 배터리 전압모니터링, 무선 전송 전력, 환경 온도, 환경 조도, 2축 가속도계(2-Axis Accelerometer; 가속도 센서), 2축 자기계(2-Axis Magnetometer; 자기센서) 등을 이용하여 환경 모니터링이 가능하다.

<표 2>의 데이터 측정은 센서노드의 배터리 전압과 데이터 전송에 사용되는 무선 전송전력을 계산하여 센서노드의 배터리 방전을 응용프로그램 서버로 확인시키기 위해 사용된다. 센서노드는 별도의 전원공급 없이 외부에서 배터리로 정보의 수집과 계산 그리고 데이터의 송신을 하게 되며 센서노드의 배터리 확인은 반드시 필요하다. 또한 많은 무선 전송 전력의 사용으로 배터리의 수명이 짧아지는 것을 방지하기 위해 자기센서로부터 새로운 데이터의 수집이 있을 경

〈표 2〉 raw 데이터 계산 식

데이터	계산 식
배터리 전압	$V_{batt} = V_{ref} \times ADC\_FS/ADC\_Count$
무선 전송 전력	$V_{RSSI} = V_{batt} \times ADC\_Counts/1024$ $RSSI(dBm) = -50.0 \times V_{RSSI} - 45.5$
환경 온도/조도	$1/T(K) = a + b \times \ln(R_{thr}) + C \times [\ln(R_{thr})]^3$ $R_{thr} = R1(ADC\_FS - ADC)/ADC$



(그림 6) 트랙구간별 라인트레이서의 상황인식위치

우에만 데이터를 전송하며 이러한 방식은 TinyOS 데이터 전송 스케쥴링을 따른다. 자기센서는 라인트레이서의 위치를 감지하여 현재의 주변 온도와 조도 등을 싱크노드로 전송한다. 전송된 데이터는 서버 응용프로그램에서 상황인식을 판단하기 위한 데이터로 사용한다.

가속도 센서는 라인트레이서의 진행방향에 따른 높낮이를 측정하여 언덕과 같은 장애물인식 및 속도 제어에 사용한다. 가속도센서의 2축 센서를 이용하여 기존 진행방향에서 변화한 높이의 폭을 인식하게 되며 높은 지형에서 낮은 지형으로 라인트레이서의 진행을 측정하여 속도 제어할 수 있는 센서이다.

5. 시뮬레이션 실행 및 분석

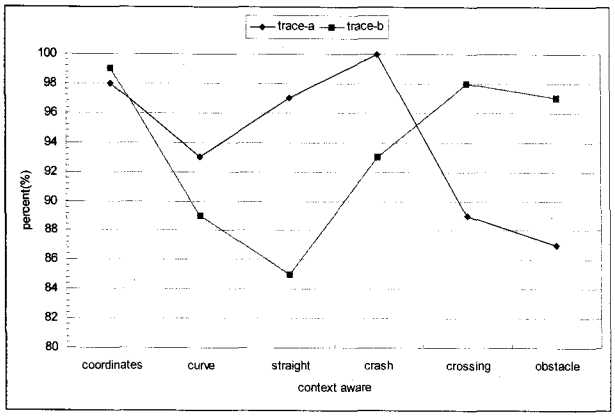
자기센서를 이용한 위치추적방법의 유용성을 검증하기 위해 센서노드에 탑재된 TinyOS와 응용프로그램의 동작 및 상호작용 시뮬레이션이 가능한 OS시뮬레이터인 TOSSIM을 사용하여 실험하였다. TOSSIM은 TinyOS시뮬레이터로서 제한되지 않은 센서노드의 수와 센서들을 사용할 수 있게 도와주며 TOSSIM과 함께 사용하는 TinyViz는 TinyOS에서 처리된 데이터들을 비주얼한 화면으로 표현해 주는 도구이다. TOSSIM에 (그림 6)의 라인트랙을 가로 3m 세로 1.5m의 영역에 센서노드를 30cm간격으로 배치하여 두 대의 라인트레이서 tracer-a와 tracer-b의 이동경로추적과 이동경로에 따른 상황인식에 관하여 실험하였다.

TOSSIM의 실험에서 (그림 6)과 같이 trace-a는 반시계방향 trace-b는 시계방향으로 진행하여 위치추적과 파란색으로 구분된 지역에서의 상황인식에 관하여 분석하였다.

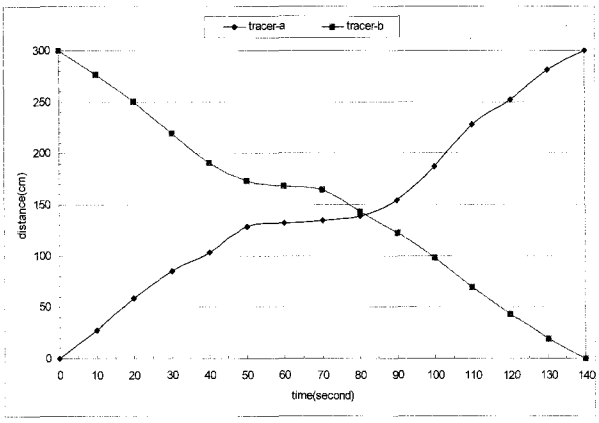
라인트레이서 trace-a와 trace-b의 좌표인식 범위는 (그림

〈표 3〉 라인트레이서 상황인식 표

트랙 대조번호	상황인식	설명
트랙전역	좌표인식	라인트레이서의 상황인식을 위한 위치감지 정확도
6-1	곡선인식	곡선에서의 상황인식 감지와 라인트레이서 속도 제어
6-2	직진성인식	직선상황에서 라인트레이서의 가·감속 속도제어
6-3	충돌인식	충돌인식상황에서의 라인트레이서 상황인식 및 회피능력
6-4	교차점인식	교차점인식 및 교차점에서의 상황에 따른 방향제어
6-5	장애물인식	정의된 장애물 인식에 따른 상황판단 및 라인트레이서 제어



(그림 7) 상황에 따른 상황인식의 정확성



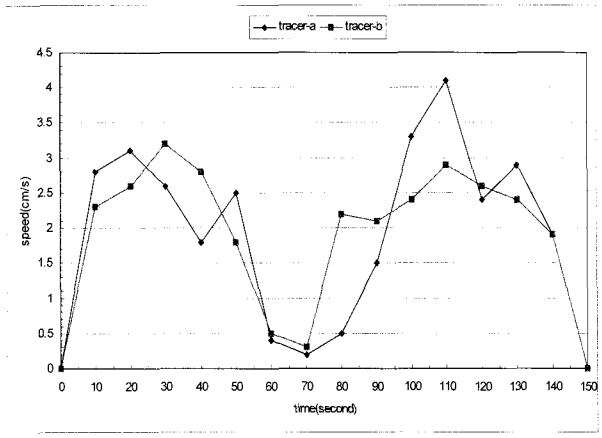
(그림 8) 단위시간 대비 주행거리

6)의 트랙전역이며 곡선인식 범위는 9군데이다. 라인트레이서의 직진구역 상황인식을 위한 2.7m길이의 1곳과 1/3또는 2/3길이의 직진구간이 6군데이다. 충돌인식, 교차점인식, 장애물인식 구간은 중간에 섞여 있어 상황인식 능력을 확인할 수 있다.

(그림 7)은 TOSSIM데이터의 평균을 이용하여 <표 3>의 상황인식에 따른 정확성을 보여주는 그래프이다. (그림 7)과 같이 라인트레이서의 위치측정에 중요한 좌표점 인식은 98%이상의 높은 인식률을 확인 할 수 있다. trace-a는 곡선과 직선가속도 그리고 충돌인식에서 낮은 인식률을 확인 할 수 있으며 직선가속도 인식을 기점으로 점차 인식확률이 증가하였다. 이에 반해 trace-b는 곡선과 직선가속도 그리고 충돌인식은 높은 인식률을 확인할 수 있으나 충돌인식 이후부터 저조한 인식률을 보여준다.

(그림 7)의 trace-a와 trace-b는 동일한 트랙구간에서 서로 다른 상황인식률을 보여 주는데 이는 두 대의 라인트레이서가 다른 진행방향으로 진행하여 인식하는 상황의 순서가 다르기 때문이다. 따라서 (그림 7)과 같이 trace-a와 trace-b의 위치인식좌표는 높은 인식률을 보이는 반면에 시작위치와 진행구간의 상황에 따른 상황인식의 정확성이 불규칙한 것을 알 수 있다. 두 대의 라인트레이서 trace-a와 trace-b의 상황인식 그래프가 서로 대치되는 점으로 미루어 보아 간단한 상황인식 후 복잡한 상황인식인 경우 상황인식률이 저조하지만 라인트레이서의 전체적인 상황인식률은 tracer-a는 94%이며 tracer-b는 93.5%로 서로 비슷한 상황인식률을 확인 할 수 있다. 하지만 tracer-a는 가장 중요한 상황인식인 충돌인식과 교차점 및 장애물인식에서 tracer-b보다 뛰어난 것을 확인하였다.

라인트레이서 tracer-a와 tracer-b의 상황인식에 따른 제어처리를 확인하기 위해 3m구간의 이동가속도 변화폭을 측정하였고 충돌상황인식은 tracer-a와 tracer-b의 사이에 센서노드가 4개 존재한다고 가정하였을 때 충돌상황으로 인식하도록 설정하였다. 3m의 거리를 이동하는데 소요되는 시간은 (그림 8)과 같이 약 140초이며 2/4의 거리에 도달 하였을 경우 속도가 느려지는 것을 확인 할 수 있다. 이는 tracer-a



(그림 9) 단위시간 대비 이동속도

와 tracer-b를 같은 곳에서 서로 다른 방향으로 동일한 시간에 출발하여 라인트레이서는 평균 10초에 21.43cm를 이동하였고, 측정된 데이터를 기준으로 평균 속력은 2.14cm/sec의 결과를 확인할 수 있다.

(그림 9)의 그래프에서 60초에서 80초의 저속 주행은 직선거리에서의 충돌상황인식으로 인한 라인트레이서의 속도제어로 인한 것이다. 또한 10초, 50초, 110초에서 급격한 속도변화가 나타나는데 이는 라인트레이서가 충돌인식과 같은 상황인식에 따라 평균 속력을 유지할 수 없는 구간에 도달하거나 통과할 경우에 변화량이 증가 하거나 감소하였다. 이는 (그림 8)에서도 (그림 9)와 같은 변화량을 유지 하는데 (그림 8)에서 50sec에서 85sec에 이동 거리가 짧아 (그림 9)에서 단위시간 대비 이동속도 또한 감소하는 결과를 보이고 있다.

6. 결론 및 향후 연구과제

본 논문에서는 센서네트워크기반의 자기센서를 이용하여 자성체(Magnetic ; Line Tracer)의 위치추적과 데이터 처리 그리고 제어하는 방법을 제안하였다. 제안된 방법은 라인트

레이서의 위치, 방위정보와 더불어 속력정보를 추출하고 주변 환경 정보를 확인하여 라인트레이서의 충돌 방지 행동을 추론하는데 이용된다. 라인트레이서의 주변 환경 정보를 이용하여 현재의 상태 또는 직선이나 굴곡 교차로의 근접을 알 수 있으며, 이는 정보이용자 측면에서 라인트레이서가 위치한 주변 환경정보를 원격지에서 확인할 수 있도록 정보 제공에 이용된다. 이러한 제안 방법은 객체의 상황을 좀 더 정확하게 인식하고 이를 바탕으로 객체의 제어를 추론하기 위한 정보 및 사용자 서비스 정보를 제공하며 결과적으로 상황에 따른 가장 적합한 서비스를 제공할 수 있도록 한다.

본 논문에서 제안한 위치추적 방법은 현재 사용되고 있는 위치추적 시스템을 자기센서를 이용하여 개선하였다. 현재의 위치추적 방법은 한시적으로 제한된 공간 안에서의 위치 추적 방법이며 이는, 여러 제약 조건과 사용자가 요구하는 정보를 확보할 수 없다. 그러나 본 논문에서 제안한 위치 추적 방법은 제한된 공간과 범용의 공간에서 위치정보 및 주변 환경정보를 이용하여 사용자가 요구하는 정보를 충족시켜 주며, 이러한 위치추적 방법을 확장하여 실시간 스케쥴링 기법과 실시간 데이터베이스를 이용한다면 빠르고 정확한 행동 정보의 추론 엔진을 구현할 수 있을 것으로 예상된다.

따라서, 향후 연구과제로서 본 연구에서 제안한 위치추적 방법을 실시간 스케쥴링 기법과 실시간 데이터베이스를 통해 제안된 위치추적 방법의 우수성을 검증하고, 새롭게 도출될 수 있는 문제점과 개선점을 찾아 극복하는 과정이 이루어져야 한다. 이러한 과정에서 도출된 결과는 앞으로의 센서네트워크 세상에서의 사용자 정보 추론 및 경로 추적을 위한 여러 분야에서 이용될 것으로 보인다.

참 고 문 헌

[1] 정우진, 윤효석, 우운택, "ubiHome을 위한 ubiTrack 기반 위치 추적 방법"

[2] 남상엽, 송병훈, "Mote-Kit를 이용한 무선 센서 네트워크 활용", 상학당, 2005

[3] P. Enge, and P. Misra, "Special issue on GPS: The Global Positioning System," Proc, of the IEEE International Conf. on, Jan., pp.3-172, 1999

[4] Digital Cellular Telecommunications System(Phase 2+); Location Services(LCS); (Functional description) - stage 2,(GSM 03.71 version 8.0.0 Release 1999).

[5] 김창화 외2인, "LEGO로 즐기는 로봇공학", 이지테크, 2004

[6] Jochen Schiller, "Mobile Communications second edition", 한티미디어, 2003

[7] Alec Woo, "More routing protocols", PPT자료, 2002

[8] David Wagner, "Wireless Mobile Platform OAEP", UC Berkeley자료

[9] 대한교통학회, "지능형교통시스템 기본계획 수립을 위한 차세대 도로시스템 연구", 1996

[10] ITS Korea, http://www.itskorea.or.kr

[11] 정보통신부, "지능형교통체계(ITS) 활성화 방안", 1999

[12] 김재형, "첨단 차량/도로 시스템(AVHS)을 위한 통신 시스템의 신뢰도 분석", 1997

[13] Jiro Kumayama, "Advanced Cruise-Assist Highway System Technology", 2002.6

[14] 신동필, "교통신호처리 시스템개발을 위한 선행 연구", 한국전자통신연구원, 1990

[15] J. Hightower and G. Borriello, "Location Sensing Techniques"

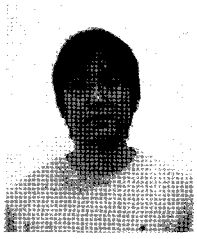
[16] J. Werb and C. Lanzl, "Designing a positioning system for finding things and people indoor," IEEE Spectrum, Vol.35, Issue 9, Sep., pp.71-78, 1998

[17] L.M.Ni, U. Liu, U. C. Lau and A. P. Patil, "LANDMARC: Indoor Location Sensing Using Active RFID," Pooc. of the IEEE International Conf. on PerCom, Dec., pp.407-416, 2004

[18] P. Enge, and P. Misra, "Special issue on GPS: The Global Positioning System," Proc, of the IEEE International Conf. on, Jan., pp.3-172. 1999

[19] N. Priyantha, A. Miu, H. Balakrishnan, S. Teller, "The Cricket Compass For Context-Aware Mobile Applications," ACM MobiCom pp.1-14, 2001

[20] Y.Oh, W.Woo, "User-centric Integration of 5W1H Contexts for A Unified Context-aware Application Model," UbiPCMM05.



김 홍 규

e-mail : beeniis@suwon.ac.kr

2004년 평택대학교 컴퓨터학과 학사

2006년 수원대학교 컴퓨터학과 석사

2006년~현재 수원대학교 컴퓨터학과 박사과정

관심분야 : 센서네트워크 위치추적, 센서네트워크 헬스케어(BSN), 실시간 센서네트워크 운영체제, 실시간 데이터 베이스

문 승 진

e-mail : sjmoon@suwon.ac.kr

1986년 미국 텍사스 주립대학교 컴퓨터학과 학사

1991년 미국 플로리다 주립대학교 컴퓨터학과 석사

1997년 미국 플로리다 주립대학교 컴퓨터학과 박사

1997년~현재 수원대학교 IT대학 컴퓨터학과 부교수

관심분야 : 실시간 임베디드 시스템, 실시간 센서네트워크 시스템, 실시간 데이터 베이스