고신뢰성 USN 응용 서비스 지원을 위한 오작동 진단 상황인지 미들웨어 구현

Implementation of Failure-Diagnostic Context-awareness Middleware for Support Highly Reliable USN Application Service

이 용 웅* 김 세 한** 손 교 훈*** 이 인 환**** 신 창 선*****
Lee Yong-Woong Kim Se-Han Son Kyo-Hun Lee In-Hwan Shin Chang-Sun

요 약

본 논문에서는 실내에 센서 네트워크가 적용된 USN 응용 시스템에서 발생하는 센서나 설비 장치의 오작동을 진단하여, 해당 시스템이 제공하는 서비스의 신뢰성을 높여주는 오작동 진단 상황인지 미들웨어를 제안한다. 본 논문에서 새롭게 제안하는 미들웨어는 데이터관리 모듈, 상황정보제공 모듈, 상황분석 모듈, 서비스제공 모듈, 정보저장소 모듈로 구성되며, 모듈 간의 상호작용으로 얻은 데이터는 오작동 진단 알고리즘을 통해 비교 분석함으로써, 센서나 설비 장치의 오작동 여부를 판단한다. 구현된 미들웨어는 시뮬레이션을 통해 수행성을 검증하였다. 그 결과 다수의 센서가 설치된 대형 시스템에서 본 미들웨어가 높은 성능을 발휘한다는 것을 확인할 수 있었다.

ABSTRACT

In this paper, we proposed the Failure-Diagnostic Context-awareness Middleware (FDCM) for improving the reliability in the USN application service. The middleware diagnoses the failure occurred in sensors or facilities in the indoor USN application system. The new middleware suggested in this paper consists of DataManagement module, ContextProvider module, ContextInterpreter module, ServiceProvider module and DataStorage module. By analysing the data obtained by the interaction between modules through the diagnostic algorithm, the FDCM determines the malfunction of sensors and equipment devices. Then we verified the performance of middleware by using simulation. As a result, the FDCM showed the high performance in the large systems that many of the sensors and devices are installed.

☞ keyword : USN 미들웨어, 상황인지, 오작동 진단, USN, USN Middleware, Context-awareness, Failure Diagnosis, USN

1. 서 론

USN(Ubiquitous Sensor Network)은 모든 사물에 RFID(Radio-Frequency Identification)나 센서를 부착하여 사용자에게 최적의 서비스를 가능하게 하여 언제, 어디서나 사용자가 요구하는 맞춤형 서비스를 제공하는 기술이다. 이러한 USN 기술은 국방, 제조, 건설, 교통, 의료, 환경, 교육, 물류, 유통, 농/축산업 등 다양한 분야에 적용이 가능하여 향후 정보통신 분야에 국가 경쟁력을 좌우할 가장 유망한 차세대 성장 동력이자 사회 전반의 일대 혁신을 가져올 수 있는 중요한 미래 기술로 관심을 받고 있다. 이미 우리나라를 포함

* 준 회 원 : 순천대학교 정보통신공학과 석사
kgcri@hanmail.net
** 정 회 원 : 한국전자통신연구원 선임연구원
shkim72@etri.re.kr
*** 정 회 원 : 한국전자통신연구원 선임연구원
sonkh@etri.re.kr
**** 정 회 원 : 한국전자통신연구원 USN기반기술연구팀 (팀장) 책임연구원 ihlee@etri.re.kr
***** 정 회 원 : 순천대학교 정보통신공학과 교수
csshin@sunchon.ac.kr(교신저자)
[2010/12/27 투고 - 2011/01/12 심사(2011/03/10 2차) - 2011/03/21 심사완료]

한 미국, 유럽, 일본 등 일부 선진국에서는 USN 기반 기술을 상당 부분 확보하고 있으며 응용기술에서 우위를 차지하기 위한 치열한 경쟁이 전개되고 있다[1-3].

일반적으로 USN 기술을 활용한 시스템들은 시스템 내에 구성된 센서 네트워크에서 수집한 다량의 데이터를 바탕으로 서비스를 제공한다. 이러한 USN 응용 시스템들의 특성 때문에 설치된 센서들이나 설비장치들의 오작동이 발생할 경우 해당 시스템의 성능에 큰 제약을 준다. 하지만 일반적으로 많은 수의 노드로 망을 구성하기 때문에, 각각의 센서가 정상 동작하는지 파악하는 일은 쉬운 일이 아니며, 이를 위해 많은 관리 인력 및 비용이 소모되는 실정이다.

본 논문에서는 이러한 문제점을 해결하기 위해 USN 응용시스템의 고신뢰성 서비스 지원을 위한 오작동 진단 상황인지 미들웨어(Failure-Diagnostic Context-awareness Middleware, FDCM)를 제안한다.

본 논문의 구성은 2장에서 본 논문이 제안하는 기술의 핵심이 되는 USN 미들웨어 및 상황인지에 관련된 IT관련 연구를 소개하고, 3장에서는 FDCM의 구조 및 제공 서비스를 설명하며, 4장에서는 FDCM의 수행성을 검증하기 구동결과에 대하여 설명 하고, 마지막으로 5장 결론 및 향후 연구내용을 제시한다.

2. 관련 연구

2.1 USN 미들웨어

다양한 산업 분야에서 USN 응용 서비스를 구축하기 위해 USN 미들웨어도 그 적용분야에 따라 기능 및 동작특성을 지원하는 방향으로 연구되고 있다[4-5].

ETRI는 다수의 이기종 센서 네트워크를 손쉽게 통합할 수 있도록 센서 네트워크 추상화를 지원하고 센서 네트워크의 상태를 실시간 모니터링 할 수 있는 COSMOS(COmmon System for Middleware Of Sensor networks)를 개발하였다 . COSMOS는 USN 전용 미들웨어로 센서 네트워크로부터 수집되는 센싱 데이터를 가공하여 정보로서의 의미를 부여한다. 본 미들웨어는 기 구축된 비즈니스 정보를 활용하여 수집된 센싱 데이터와 결합하여 새로운 상황 정보를 생성하는 역할을 수행한다. 또한 API를 제공함으로써 USN 응용 서비스의 개발을 효율적으로 지원하고 다수의 서비스 사용자들에 대한 관리 기능을 제공한다[2-6].

LSIR Laboratory의 Karl Aberer의 GSN(Global Sensor Networks) 프로젝트는 서로 다른 기관 또는 그룹에서 구축한 센서 네트워크들을 상호 연결을 통해 센싱 데이터의 통합 및 분산 질의처리를 지원하는 것을 목표로 하였다. 이를 위해 GSN은 센서 네트워크와 생성된 데이터 스트림을 동적으로 통합 및 관리한다[7].

2.2 상황인지 미들웨어

Gaia는 상황인지 서비스 개발 아키텍처로서 응용 서비스에 필요한 다양한 상황정보를 수집하고 추론할 수 있는 인프라를 제공한다[8]. Gaia의 상황정보제공자는 센서 또는 상황정보 데이터 소스로부터 상황정보를 수집하여 제공하며 상황정보합성기는 수집된 상황정보로부터 상위 개념의 상황정보로 추론할 수 있는 기능을 제공한다. 상황정보 제공자룩업서비스는 서비스 지원을 위해 요구되는 상황정보제공자의 속성정보를 제공하는 기능을 담당하며, 상황정보히스토리는 제공된 상황정보에 대한 명세를 제공한다.

MoCE(MobileContextExplorer)는 빠르고 쉬운 상황인지 기반 서비스 개발을 위해 모바일 환경에서 네트워크를 이용하여 이질적인 장치간의 상황정보를 공유하는 서비스 프레임워크이다[9]. MoCE 아키텍처는 추상화된 정보모델을 설계 하였으며 효율적인 상황정보의 공유를 위하여 상황정보 디스커버리 프로토콜과 상황정보 전송프로토콜을 사용한다.

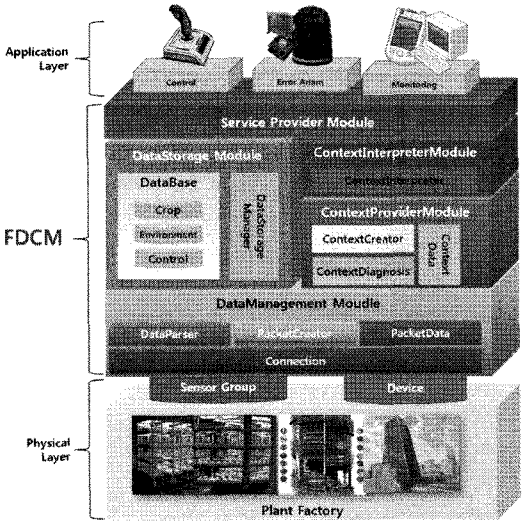
2.3 센서 오작동 검출

일반적으로 이러한 센서 노드의 오작동은 마이크로프로세서 및 전력 고갈 등과 같은 하드 오작동(Hard Failure)과 센서의 켈리브레이션 에러, 랜덤 노이즈 에러 등과 같은 소프트오작동(Soft Failure)으로 구분될 수 있다[10,11].

센서 노드의 하드 오작동과 관련된 연구로 전력고갈 및 하드웨어 오작동 등의 검출을 위해 BIST(Bult-In Self-Test)기법을 적용한 연구가 진행된바 있다. BIST 기법은 어느 정도의 하드웨어 중복성(Redundancy)이 보장되는 상호 연결된 다중프로세서(Multiprocessor)시스템의 오작동진단에 적용될 수 있어 광범위한 영역에 분산 설치되는 센서 노드들에 효과적인 적용이 가능하다. 또한 하드웨어 중복성이 없는 경우에 적용 가능한 오작동진단 기법으로 Consensus알고리즘이 있다[12,13]. Consensus 알고리즘은 센서 네트워크상에서 오작동이 없는 것으로 판정된 노드들에 의해 고정 노드를 검출하는 알고리즘으로 주변 노드로부터의 정보로 생성되는 Suspect matrix와 Fault vector를 이용하여 주변 노드들과의 지속적으로 정보 교류와 갱신을 통해 오작동을 검출한다.

센서의 소프트 오작동의 검출을 위한 다양한 기법들로는 J.Chen 등은 주기적으로 얻어지는 주변 노드들의 센서 값과 자신의 측정값 간의 차이 및 이의 시간에 따른 변화를 기반으로 주변노드로부터 전송된 센서 데이터의 에러를 검출할 수 있는 LFSD(Localized Faulty Sensor Detection) 알고리즘이 있다[14].

이와 같이 현재까지 연구된 USN 미들웨어 및 상황인지 미들웨어는 센서네트워크의 범용성 및 사용자에게 필요한 정보를 추론할 수 있는 기술들을 중심으로 개발되었다. 그러나 USN 응용 시스템의 운영과정에서 필요한 센서나 설비장치의 오작동 등의 진단을 위한 연구는 포함하지 못했다. 또한 센서 오작동 검출 기법들은 센서간의 통신을 통해 에러를 검출하기 때문에 전력소모



(그림 1) USN 응용 시스템 구성도

및 시스템 구성에 제약을 갖게 되며 설비장치들의 동작을 고려하지 않았다. 본 논문에서는 USN 미들웨어에 상황인지개념과 오작동 검출 기법을 통해 센서의 오작동 여부를 효과적으로 판단하는 미들웨어를 제안한다.

3. 오작동 진단 상황인지 미들웨어

3.1 FDCM 구조

본 논문에서 제안하는 오작동 진단 상황인지 미들웨어(FDCM)는 USN 응용시스템의 신뢰성을 높이고 시스템 운영 중에 시설 관리의 효율화를 제공한다. USN 응용 서비스를 지원하기 위한 시스템 구조는 (그림 1)과 같으며 물리계층, 중간계층 및 응용계층으로 구성된다. 물리계층에는 센서들로 구성된 센서그룹과 응용 설비기기들이 존재한다. 중간계층인 FDCM은 데이터관리 모듈, 상황정보제공 모듈, 상황분석 모듈, 서비스제공 모듈, 정보저장소 모듈로 구성된다. 응용계층에는 센서나 설비들을 통한 응용 모니터링 및 제어, 오작동 경보 등의 서비스가 제공된다. 본 논문에서는 중간계층인 FDCM의 구현을 연구의 목적으

(표 1) 패킷 데이터 구성

패킷 데이터		
변수	설명	비고
id	해당 센서의 ID	
block_id	해당 센서가 설치된 블록의 ID	
sector_id	해당 블록이 포함된 섹터의 ID	
data_CODE	센서가 수집하는 데이터의 종류	
currentData	현재 센서가 수집한 데이터	
beforeData	currentData 수집하기 바로 이전 센싱 데이터	
blcokAve	같은 블록내 센서들의 currentData의 평균값	현재 센서 제외
beforeBlockAve	같은 블록내 센서들의 beforeData의 평균값	
sectorAve	같은 블록내 센서들의 blockData의 평균값	현재 센서 제외

로 하며 물리적인 장치 및 서비스는 응용의 특성에 따라 구성을 달리할 수 있다.

3.1.1 데이터관리 모듈

데이터관리 모듈은 센서에서 수집한 데이터를 처리하여 데이터베이스에 저장하고 센서나 설비 기기들의 오작동 여부를 조사하는 상황정보제공자 모듈로 전달하는 기능을 한다. 단순히 센서에서 수집한 데이터만으로는 정상동작 상황을 파악하기 어렵기 때문에 수집된 센싱 데이터에 판단에 필요한 정보들을 추가하여 패킷을 재구성한다. 데이터 관리 모듈은 DataParser 클래스, Connection 클래스, PacketData 클래스, PacketCreator 클래스로 구성된다. DataParser 클래스는 여러 이기종 센서에서 수집된 원시데이터를 가공하여 실제 오작동 여부 판단의 근거가 되는 데이터를 추출한다. 수집된 센싱 데이터에서 싱크 바이트 및 헤더부분을 제거하는 기능과 1차 가공된 데이터를 실제 활용할 수 있는 데이터로 변환하는 기능을 지원한다. Connection 클래스는 싱크노드에서 수집한 센싱 데이터들을 시리얼통신으로 상황정보제공자 모듈로 전달한다. 통신을 위해 필요한 기본정보를 설정하고 시리얼 통신의 연결과 해제, 데이터 전송 중단과 같은 기능을 수행한다. PacketData 클래스는 상황정보제공자 모듈에서 데이터를 분석할 수 있게 필요정보를 추가하여 재구성 패킷을 정의하는 클래스이다. (표 1)은 본 클래스에서 구성하는 패킷의 구조를 보인다. 이전 수집된 정보들로부터 현재 수집된 정보를 통해 센서의 오동작 여부를 판단할 수 있는 정보들을 포함한다. 패킷 정보 생성을 지원하는 클래스인 PacketCreator는 PacketData 클래스가 관리하는 변수들의 데이터 입력 및 데이터가 패킷을 반환하는 기능을 수행한다. 본 논문에서는 USN 응용 시스템의 센서 및 설비기기가 설치된 구역별로 블록과 섹터라는 영역을 정의하였다. 블록은 3개 이상의 센서로 이루어진 센서 그룹으로 설비 기기가 영향을 미칠 수 있는 범위이며, 섹터는 3개 이상의 블록으로 이루어진다. 여기에서 정의한 블록과 섹터는 일반적은 USN 응용 시스템을 구성하는 센서나 설비기기들의 오작동 진단을 위해 주변 데이터를 비교할 수 있는 최소한의 개수로 정의했으며, 응용에 따라 그 수 및 범위는 증가할 수 있다.

3.1.2 상황정보제공 모듈

상황정보제공 모듈은 데이터관리 모듈에서 전송한 재구성된 데이터 패킷을 상황정보로 변환시켜주는 역할을 수행하며, ContextDiagnosis 클래스, ContextData 클래스와 ContextCreator 클래

(표 2) 상황정보 구성

상황정보	
변수	설명
packet	패킷정보
isDecision	상황이 결정 되어 있는가?
isErr	오류에 관한 상황인가?
isSensor	센서에서 발생한 상황인가?
isDevice	설비 장치에서 발생한 상황인가?
isActivation	설비 장치가 구동중인가?
DEVICE_CODE	관련 있는 설비 장치의 ID
deviceName	관련 있는 설비 장치의 이름
optimalName	해당 섹터에서 관리하는 분야 이름
optimalData	해당 분야의 최적 기준정보
DataType	관련 있는 데이터 종류

스로 구성된다. ContextDiagnosis 클래스는 데이터관리 모듈에서 재구성한 패킷을 분석하여 센서 및 설비 장치들의 오작동을 확인하며, 설비기기의 구동이 필요한 상황을 판별하고 이를 상황정보로 재구성한다. ContextData 클래스는 상황분석 모듈에서 요청할 서비스를 분석할 수 있도록 상황정보를 정의하는 클래스로 (표 2)와 같이 상황정보를 결정하기 위한 정보들을 포함한다. 상황정보는 ContextCreator 클래스로부터 데이터를 입력받는다. 본 클래스는 센서 및 설비기기 오류를 판단할 수 있도록 상황정보를 설정하는 기능과 해당섹터에서 관리하는 최적 기준 데이터 및 응용 서비스 수행과 관련 있는 설비기기의 ID를 설정하는 기능을 수행한다.

3.1.3 상황분석 모듈

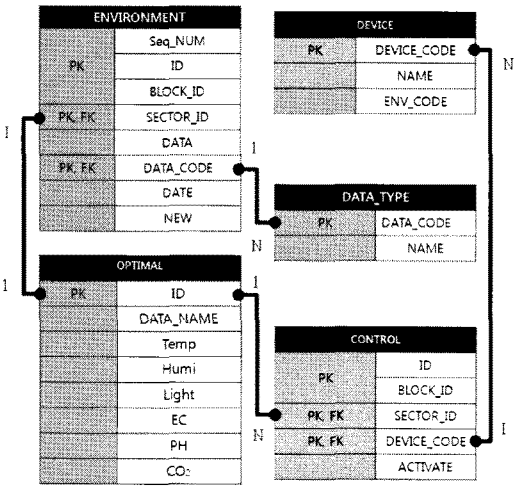
상황분석 모듈은 상황정보제공 모듈에서 전송한 상황정보를 분석하여 어떠한 응용 서비스를 제공할 것인지를 최종 결정한다. 본 모듈을 구성하는 ContextInterpreter 클래스는 상황정보제공 모듈에서 전송한 상황정보를 분석하여 서비스제공 모듈에 요청 하는 클래스로 상황정보를 분석하여 현재 센서나 설비기기들이 제대로 작동하고 있는지 판단하는 기능을 수행하고, 센서나 설비기기들의 오작동 시 서비스제공 모듈에 오작동 알림 서비스를 호출한다.

3.1.4 서비스제공 모듈

서비스제공 모듈은 본 미들웨어에서 제공하는 서비스들이 구현 되어 있는 모듈이다. 상황분석 모듈에서 상황 분석 결과로 요청하면 해당 서비스를 제공해준다. 오작동 알림 서비스가 구현된 ServiceProvider 클래스는 요청된 데이터를 분석하여 어떤 오류 메세지를 실행할 것인지 결정하고 이상이 있는 센서 및 설비기기의 정보를 사용자에게 제공한다.

3.1.5 정보저장소 모듈

정보저장소 모듈에는 응용 시스템에 필요한 기준 정보 및 센서에서 수집한 환경정보를 저장한다. 또한 데이터저장소로부터 데이터를 추출하여 사용할 데이터를 정의하고, 데이터 처리에 관련된 메소드를 정의 한다. 본 미들웨어는 데이터베이스를 통하여 데이터 수집 모듈로부터 데이터를 전달받기 때문에 패킷의 재구성이나 상황정보를 진단할 때 요구되는 정보들도 데이터베이스를 통한 전달 방식으로 이루어진다. 정보저장소 모듈은 DataStorageManager 클래스와 데이터베이스로 구성되어 있다. DataStorageManager 클래스는 데이터베이스에 데이터를 삽입, 수정 및 삭제 와 같은 전반적인 사항을 관리하며 다른 모듈에서 요청하는 데이터를 전달한다. 정보저장소 모듈에서 관리하는 데이터베이스의 스키마는 그림 2와 같다. 데이터베이스는 ENVIRONMENT, DATA_TYPE, DEVICE, CONTROL, OPTIMAL총 5개의 테이블로 구성되어있다. ENVIRONMENT 테이블은 센서에서 수집된 정보를 저장하고 있으며, 센서에서 수집한 데이터 종류 목록을 저장하고 있는 DATA_TYPE 테이블과 N 대 1 관계를 가짐으로 새로운 정보를 가진 센서가 추가되더라도 유연하게 확장할 수 있도록 하였다. 설비장치도 마찬가지로 CONTROL 테이블과 DEVICE



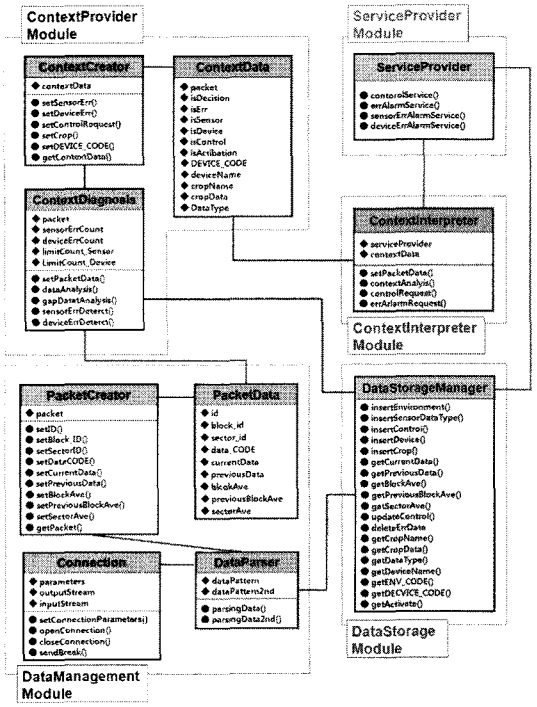
(그림 2) 데이터베이스 스키마

테이블이 N 대 1 관계를 갖도록 설계함으로써 시스템내부에 새로운 장치가 설치되더라도 유연하게 확장이 가능하다. OPTIMAL 테이블은 해당 시스템이 요구하는 최적상태의 환경 기준정보를 저장하는 테이블로써 상황정보 제공자에서 제어 및 진단을 할 때 해당 테이블을 참고하도록 하였다.

위에서 설명한 모듈과 각 모듈에 구현된 클래스들의 전체적인 연관 관계를 나타내는 FDCM의 클래스 다이어그램은 (그림 3)과 같다.

3.2 오작동 진단 서비스

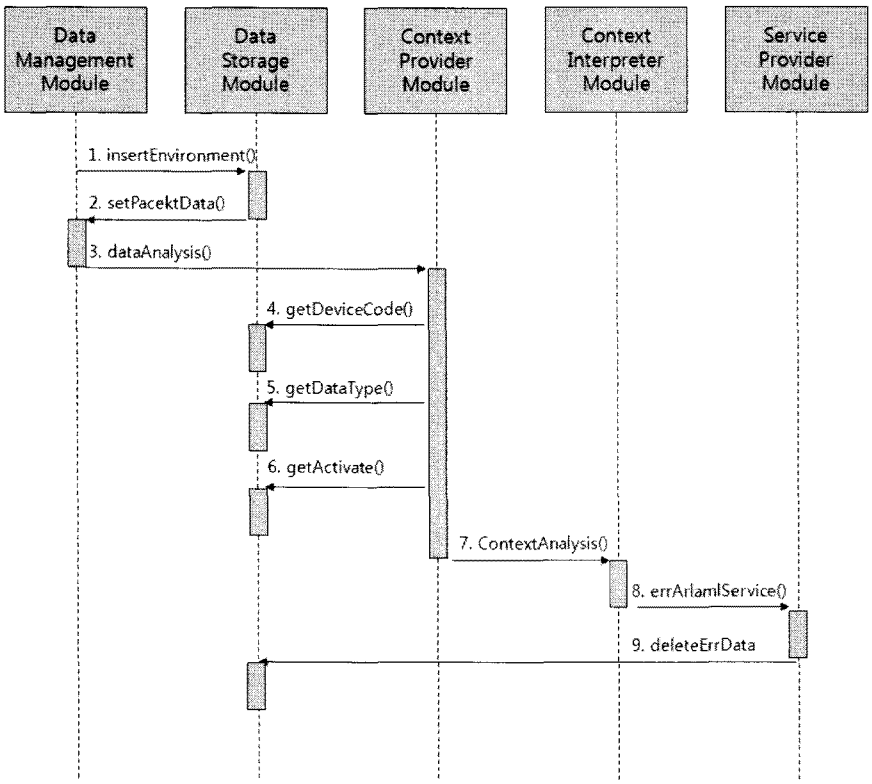
FDCM의 최종 서비스인 오작동 진단 서비스는 시스템에 설치된 센서들과 설비 장치들이 정상적으로 작동하고 있는가를 진단함으로써 효율인 시설관리와 신뢰성 있는 정보를 제공하도록 해주는 서비스이다. 구동과정은 센서에서 데이터를 전달 받으면 데이터관리 모듈에서 관련정보를 정보저장소 모듈을 통해 데이터베이스에 저장한다. 그 후 센싱 데이터의 재구성을 위해 필요한 정보를 정보저장소 모듈에 요청하여 재구성된 패킷을 만든다. 재구성된 패킷은 상황정보제공 모듈에서 분석과정을 거치면서 오작동 여부를 진단한다. 해당 센서가 오작동인 것으로 판단이



(그림 3) FDCM 구성 모듈의 클래스 다이어그램

되면 정보저장소 모듈에 데이터를 요청하여 상황정보를 만든다. 해당상황정보는 상황분석 모듈에 전달되어 상황 분석을 통해 어떤 서비스를 제공받을 것인지를 요청한다. 서비스제공 모듈은 사용자에게 센서나 기기에 관한 오류 정보를 제공한다. 이때 센서가 오류로 판단된 경우 데이터베이스에서 해당 ID의 센서 값을 삭제함으로써 다른 데이터에 영향을 주지 못하게 한다. (그림 4)는 오작동 진단서비스의 구동과정을 나타내는 시퀀스다이어그램이다.

여기서 중요한 부분은 상황정보제공 모듈에서 패킷을 분석하여 상황정보를 구성하는 과정이다. USN응용시스템의 운영 중 발생할 수 있는 상황은 크게 두 가지로 구분된다. 내부 환경이 식물 생장에 적합한 최적 수준을 유지하고 있어 설비 장치들의 추가적인 제어가 되고 있지 않은 상황과 내부 환경에 이상이 발생하여 이를 해결하기 위해 설비 장치가 구동되고 있는 상황이 있다.



(그림 4) 오작동 진단 서비스

각 상황에 따른 오작동 진단에 대한 설명은 과 같다.

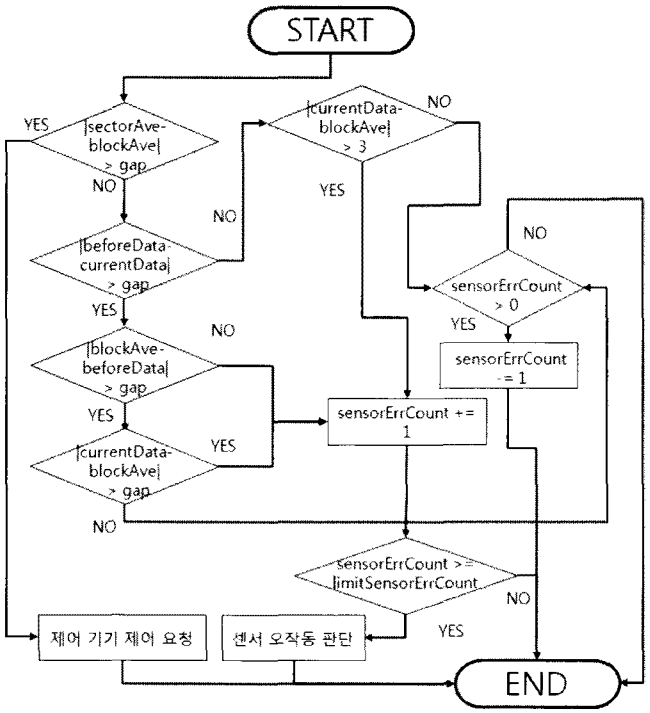
(1) 설비 장치가 제어 중이 아닌 상황

설비 장치가 제어 중이 아닌 상황일 때 서비스를 판단하는 과정은 (그림 5)와 같으며 이를 정형화한 의사코드는 (그림 6)과 같다.

장치가 작동하지 않고 있을 때 blockAve이 sectorAve차이가 gap보다 크지 않고 beforeData와 currentData가 차이가 gap 이상 나지 않을 때는 currentData와 blockAve의 비교를 통해 (그림 7)의 a)와 b) 같은 상황을 유추할 수 있다. (그림 7)의 a)와 같은 경우는 센서가 지속적으로 평균값을 유지해 왔다는 것이므로 정상 동작으로 판단하고 sensorErrCount가 0 이상일 경우 1 감소시킨다. (그림 7)의 b)와 같은 경우는 지속적으로 평균값을 벗어났다고 판단된다. 이는 센서가 오작동 되고 있을 확률이 크므로 sensorErrCount를 1 증가시켜준다.

beforeData와 currentData가 차이가 gap 이상 나고 blockAve가 beforeData의 차이가 gap보다 작다면 (그림 7)의 c)와 같이 현재 데이터가 정상적으로 동작하다가 평균값을 벗어난 상황이다. 이러한 경우 오동작이나 외부적인 요인이 발생했다고 추측할 수 있으므로 sensorErrCount를 1 증가시켜준다.

beforeData와 currentData가 차이가 gap보다 크고 blockAve가 beforeData의 차이가 gap보다 크다면 (그림 7)의 d)과 같이 센싱 데이터가 평균값을 벗어났다가 다시 평균으로 돌아온 상황을 추정할 수 있다. sensorErrCount를 1감소 시켜준다.

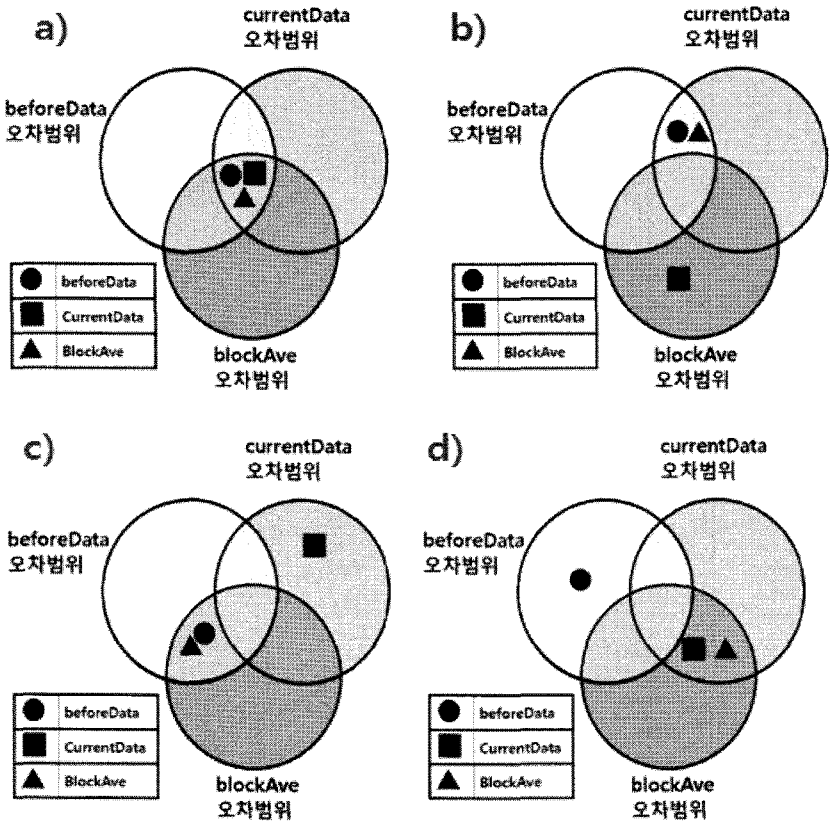


currentData : 현재 센싱 값 **beforeData** : 이전 센싱값 **blcokAve** : 블록 평균값 **beforeBlockAve** :이전 블록 평균값 **gap** : 차이를 비교하는 기준 임계값	**sectorAve** : 섹터 평균값 **sensorErrCount** : 센서 오류 카운트 **limitsensorErrCount** : 센서 오류 판정 카운트 임계값

(그림 5) 설비 장치가 제어 중이 아닐 때 오작동 진단 서비스 알고리즘

```
/* 기기 비작동 중에 센서의 오작동 감지 */
만일, 섹터 평균과 블록 평균의 차이가 작으면
    만일, 이전데이터와 현재데이터의 차이가 크면
        만일, 블록의 평균값과 이전 값의 차이가 크면
            만일, 현재 데이터와 블록평균값의 차이가 크면
                센서 오작동 카운터를 증가한다. (오류 -> 오류);
            그렇지 않으면,
                센서 오작동 카운터를 감소한다. (오류 -> 정상);
            만일 끝.
        그렇지 않으면,
            센서 오작동 카운터를 증가한다. (정상 -> 오류);
        만일 끝.
    그렇지 않으면,
        만일, 현재 데이터와 주변평균값의 차이가 크면
            센서 오작동 카운터를 증가한다. (오류->오류);
        그렇지 않으면,
            센서 오작동 카운터를 감소한다. (정상->정상);
        만일 끝.
    만일 끝.
그렇지 않으면,
    설비장치 제어 요청을 한다;
만일 끝.
```

(그림 6) 기기 비작동 중에 센서의 오작동 감지에 관한 의사코드



(그림 7) 설비장치가 작동하고 있지 않을 때 유추할 수 있는 상황

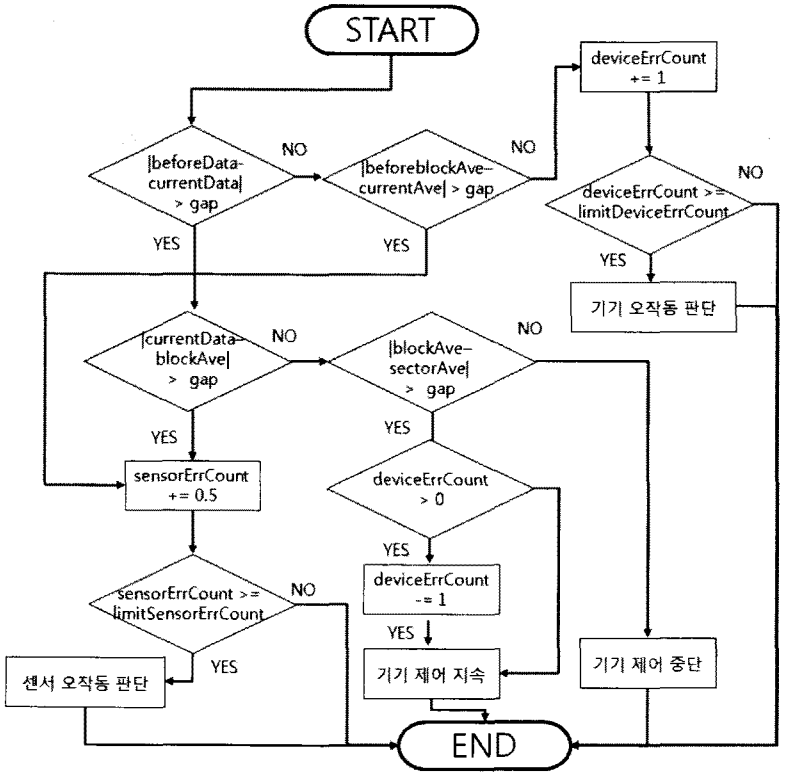
(2) 설비 장치가 제어 중인 상황

설비 장치가 제어 중인 상황일 때 서비스를 판단하는 과정은 (그림 8)과 같으며 이를 정형화한 의사코드는 (그림 9)과 같다.

설비 장치가 작동 중에 있을 때 beforeData와 currentData의 차이가 gap보다 크고 currentData와 blockAve차이가 gap보다 크면 (그림 7)의 c)와 (그림 10)의 a)와 같은 상황을 예상할 수 있다. (그림 10)의 b)는 currentData가 blockAve를 벗어난 상황이긴 하지만 설비 장치가 작동하고 있으므로 blockAve 값 및 currentData는 계속 변하게 될 것이다. 이것은 센서가 오작동인지 아니면 제어 중에 발생한 순간적인 변화인지 명확하지 않기 때문에 sensorErrCount를 0.5 증가시켜 차후 데이터 변화를 지켜본다.

설비 장치가 작동 중에 있을 때 beforeData와 currentData의 차이가 gap보다 크지 않고 currentData와 beforeBlockAve차이가 크면 센싱값은 변하지 않는데 주변 환경은 계속 변화하고 있다는 의미이므로 센서가 오작동일 수 있으므로 sensorErrCount를 0.35 높인다. 반대로 currentData와 beforeBlockAve차이가 gap보다 크지 않으면 설비 장치가 제대로 작동하지 않는다는 의미가 된다. deviceErrCount를 1 증가시킨다.

이러한 진단 과정을 통해 나올 수 있는 상황 및 조치 내용을 정리하면 (표 3)과 같다. 이러한 진단 및 조치는 식물공장에 설치되니 센서에서 데이터를 전송받을 때마다 반복되며 해당 센서의 sensorErrCount나 deviceErrCount가 임계 값인 5를 넘어가면 각각 센서 오류 및 기기오류로 판정한다.

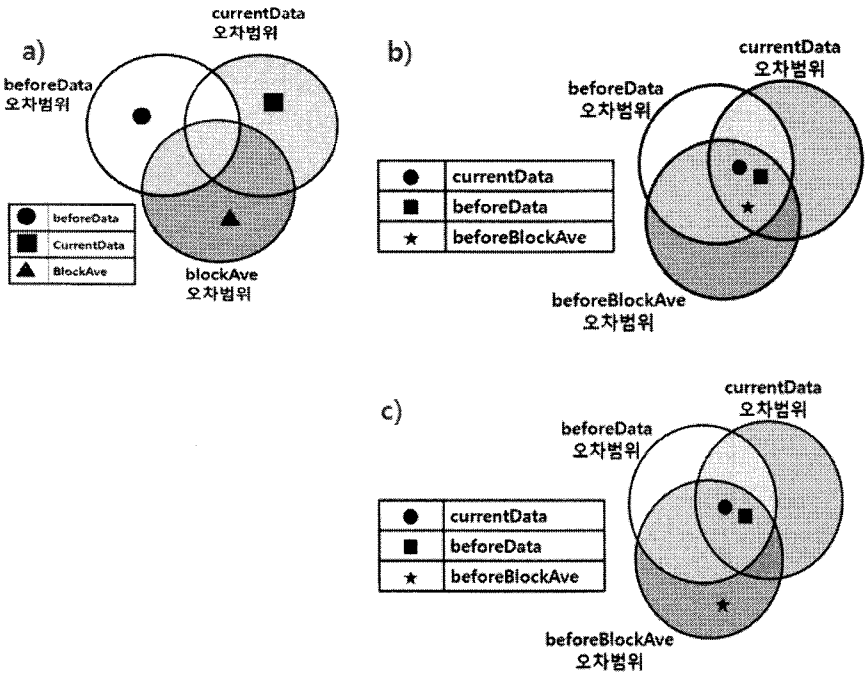


currentData: 현제 센싱 값 **beforeData**: 이전 센싱값 **blcokAve**: 블록 평균값 **beforeBlockAve**:이전 블록 평균값 **sectorAve**: 섹터 평균값 **gap**: 차이를 비교하는 기준 임계값	**sensorErrCount** :센서 오류 카운트 **limitsensorErrCount** : 센서 오류 판정 카운트 임계값 **deviceErrCount** : 제어기기 오류 카운트 **limitDeviceErrCount** : 제어기기 오류 판정 카운트 임계값

(그림 8) 설비 장치가 제어 중 일 때 오작동 진단 서비스 알고리즘

```
/* 기기 작동 중에 센서의 오작동 감지 */
만일, 이전데이터와 현재데이터의 차이가 크면
    만일, 블록의 평균값과 현재 값의 차이가 크면
        만일, 현재 값과 블록 평균값의 차이가 크면
            센서 오작동 카운터를 증가한다.(오류 확률 존재);
        그렇지 않으면,
            만일, 블록 평균과 섹터평균의차이가 크면
                센서 오작동 카운터를 증가한다.(오류 확률 존재);
            그렇지 않으면,
                설비장치 가동중단 요청을 한다;
            만일 끝.
        만일 끝.
    만일 끝.
그렇지 않으면,
    만일, 이전 블록 평균과 현재 값의 차이가 크면
        센서 오작동 카운터를 증가한다.(오류 확률 존재);
    그렇지 않으면,
        설비장치 오작동 카운터를 증가한다.(동작 하지 않음);
    만일 끝.
만일 끝.
```

(그림 9) 서비스 요청에 관한 의사코드



(그림 10) 제어장치 구동 중에 유추할 수 있는 상황

(표 3) 진단 결과를 통해 유추할 수 있는 상황 및 조치 사항

설비장치 비가동				
상황	그림 6-a)	그림 6-b)	그림 6-c)	그림 6-d)
	정상 → 정상	오류 → 오류	정상 → 오류	오류 → 정상
sensorErrCount	1감소	1 증가	1 증가	1감소
deviceErrCount	-	-	-	-
설비장치 가동 중				
상황	그림 8 - a)	그림 8 - b)	그림 8 - c)	
	장치 가동으로인한 환경 변화	해당 센서 오류 주변 데이터 정상	장치 오류	
sensorErrCount	0.5 증가	1 증가	-	
deviceErrCount	-	-	1 증가	

(표 4) 구현 환경

구현환경		
개발언어	데이터베이스	통신방법
JAVA	MS-ACCESS	시리얼, Socket

4. 수행결과

4.1 구현

본 장에서는 제안한 FDCM의 수행성을 검증하기 (표 4)와 같은 환경에서 구현하였다. 개발언어와 데이터베이스는 각각 JAVA 와 MS-ACCESS를 이용하였고 센싱 데이터를 수신하는 수신모듈과 통신방법은 시리얼통신 및 Socket 통신을 기반으로 하였다.

4.2 구동 결과

본 절에서는 FDCM을 실제 시스템에 적용하였을 때 어떤 성능을 발휘하는지 확인하기 위한 검증과정에 대하여 다룬다. 검증을 위하여 실제 센서데이터와 비슷한 형태의 가상의 데이터를 입력하고 FDCM에서 오작동된 센싱 데이터를 찾아내는지를 시험하였다. 검증을 위해 (표 5)와 같이 가상의 식물공장에서 상추를 재배한다고 가정하

(표 5) 식물공장 테스트 환경 구성

센서		내부 환경			비고
구분	개수	재배 작물	기준 환경	기준값	
센서	3~9 (블록 당)	상추	조도	25,000Lux	gap = 3
블록	9		온도	18℃	
섹터	1		습도	-20kPa	

(표 6) 가상 센싱 데이터 구분

서비스	구 분	개 수	센싱 데이터	전송 주기	전송한 데이터 수	비 고
오작동 진단	정상 센서	18~72	17~9 ℃	1 초	50	기기 작동 시 초당 0.1 ℃ 증가
	오작동 센서	9	10~30 ℃			
			10~27 ℃			
			12~27 ℃			
			12~24 ℃			
			14~24 ℃			
			14~21 ℃			
			16~21 ℃			

고 테스트를 진행하였다. 가상의 식물공장은 9개의 블록으로 이루어져 있고 섹터는 1개로 구성되어 있다. 시험 기준에 사용한 센싱 데이터의 종류는 온도로 하도록 하였고, 가상의 센서 노드에 각각 쓰레드를 구성하여 실제 센서와 같이 동시 다발적인 센싱 데이터를 전송하도록 하였다.

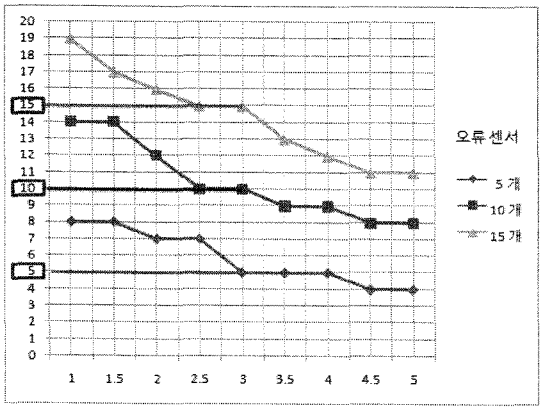
또한 센싱 데이터와 blockAve, sectorAve을 비교할 때 허용 범위의 기준이 되는 gap변수의 값은 온도 센서를 기준으로 3으로 정하였다. 근거 기준은 9개의 센서로 구성된 블록이 9개로 이루어진 시스템에서 gap변수를 1~5사이의 값으로 각각 테스트 해본 결과 그림 9와 같이 gap 변수가 3인 경우 가장 정확한 성능을 발휘하였고 작물의 재배 기준 환경에서 3℃ 이상 차이가 나면 작물에 악영향을 미칠 수 있다고 판단하였기 때문이다.

검증 시나리오는 식물공장의 총 81개의 센서 중 72개의 정상적인 센서는 18~19℃ 사이의 정상적인 데이터를 전송하고, 오류 센서로 가정한 9개의 센서는 (표 6)과 같이 7가지 범위의 값 사이의 임의의 값을 전송하도록 하였다. 또한 블록당 센서 개수에 따른 오작동 센서 검출 결과를 확인하기 위해 10~30 ℃ 사이의 임의의 값을 전송하는 오작동 센서 9개와 블록 내에 설치된 센서의 수를 3~9개까지 조절해가면서 테스트를 해보았다. 또한 제어장치가 작동 중인 상황은 온풍기를 가동해 초당 0.1℃씩 증가하여 최종적으로 5℃가 증가한다고 가정하였다.

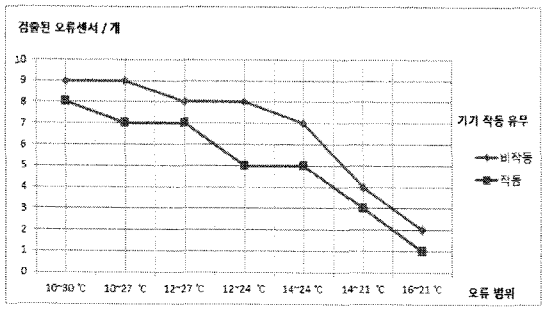
각각의 센서는 1초 간격으로 50번의 데이터를 전송하도록 테스트를 수행하였다.

테스트 결과 (그림 10)과 같이 오류데이터의 범위가 정상 데이터들과의 차이가 클수록 검출하는 오작동 센서의 정확도가 높았으며, (그림 11)과 같이 블록 내에 설치된 센서의 수가 많을수록 검출되는 오작동 센서가 많았다. 그리고 기기가 작동 중인 상황에서는 검출되는 오작동 센서가 다소 적었다. 하지만 미처 검출하지 못한 오작동 센서도 기기의 작동이 끝난 후에는 결국 검출이 될 것이므로 성능에 큰 지장을 주지 않을 것으로 예상된다.

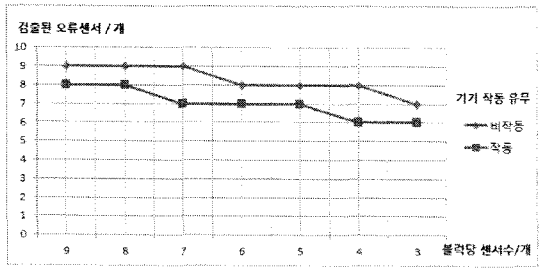
이 같은 시뮬레이션결과 센서가 한 블록에 설



(그림 11) gap 변수 결정을 위한 테스트 결과



(그림 12) 오류센싱 데이터의 범위의 차이에 따른 오작동 센서 검출결과



(그림 13) 블록당 센서 개수에 따른 오작동 센서 검출 결과

치된 숫자가 많을 수 록 오작동 센서가 정확히 검출됨을 확인하였다. 일반적으로 큰 규모의 USN응용 시스템에서는 작은 규모에 비해 센서들이 많이 설치가 되어있으므로 FDCM가 대형 시스템일수록 효과적인 성능을 발휘한다는 것을 유추할 수 있었다.

5. 결 론

본 논문에서는 USN기술을 적용한 자동화 시스템의 신뢰성 있고 효과적인 운영을 위해, 설비 장소에서 발생할 수 있는 센서나 여러 가지 기기들의 오작동이나 환경적인 문제들을 능동적으로 진단할 수 있는 FDCM을 제안하였다. 이를 위해 데이터관리 모듈, 상황정보제공 모듈, 상황분석 모듈, 서비스제공 모듈, 정보저장소 모듈 등 5개의 모듈을 구성하고 각 모듈에서 필요한 기능을 수행하는 클래스를 구현 하여, 각 모듈 간의 상호작용을 통해 센서나 설비장치의 오작동 여부를 확인하도록 하였다. 핵심기능인 센서나 설비장치의 오작동 여부를 판단하기 위해 현재 센싱 데이터, 이전 센싱 데이터, 주변데이터의 평균, 이전 주변데이터의 평균 등을 논문에서 제안하는 오류 진단 알고리즘을 통해 비교하여 오류가 나올 수 있는 상황들을 정의하고, 이를 시뮬레이션을 통해 검증하였다. 그 결과 규모가 큰 대형 시스템의 경우 효과적인 성능을 발휘 하였으며, 센서네트워크가 구성된 USN 응용시스템에 적용하여 다양한 응용 분야에 활용할 수 있음을 검증하였다.

향후 연구로는 현재 증명된 온도값 뿐만이 아닌 여러 환경정보와 센서 및 디바이스들의 특성을 감안한 기준 임계값을 여러 테스트를 통해 검증 및 개선하여 본 알고리즘을 보완 하고, 본 미들웨어가 진단할 수 있는 상황 정보를 추가함으로써 소규모 USN응용 시스템에서 발생할 수 있는 다양한 상황들에 대해서도 유연하게 대처할 수 있도록 하고, 설비장치들의 효과적인 운영을 위한 정밀한 제어알고리즘에 관한 연구가 필요하다.

Acknowledgement

본 논문은 지식경제부 산업원천기술개발사업으로 지원된 연구임.

참 고 문 헌

[1] 강정훈, "유비쿼터스 센서네트워크 기술", 방송공학회지, 2005.

[2] 김석우, "센서 네트워크 연구개발 및 사용화 사례", 주간기술동향 통권 2007.12.5

[3] 김민수, 이용준, 박종현, "USN 미들웨어 기술개발 동향", 한국전자통신연구원, 전자통신동향분석 제22권 제3호, 2007. 6

[4] 김민수, 이은규, "유비쿼터스 환경에서의 센서 데이터베이스 기술," IITA, 주간기술동향, 제1187호, 2005. 3.

[5] Salem Hadim and Nader Mohamed, "Middleware Challenges and Approaches for Wireless Sensor Networks," IEEE Distributed Systems Online, Vol.7, No.3, 2006.

[6] Marie Kim, Jun Wook Lee, Yong Joon Lee, Jae-Cheol Ryou,"COSMOS: A Middleware for Integrated Data Processing over Heterogeneous Sensor Networks," ETRI Journal, Vol.30, No.5, pp.696-706, 2008.

[7] Karl Aberer, Manfred Hauswirth, Ali Salehi, "The Global Sensor Networks middleware for effcient and flexible deployment and interconnection of sensor networks".

[8] Anand Ranganathan, RoyH. Campbell, "A Middleware for Context-Aware Agents in Ubiquitous Computing Environments," In ACM/IFIP/USENIX International Middleware Conference, RiodeJaneiro, Brazil, 2003.

[9] Hyung-Min Yoon, Woo-Shik KangG, Oh-Young Kwon, Seong-Hun Jeong, Bum-Seok Kang, Tack-Don Han, "Design of a Mobile Application Framework with Context Sensitivities", IEICE TRANSACTIONS on Information and Systems, Vol. E89-D No.2 pp.508-515, 2006.

[10] J.Chen, "Distributed Fault Detection of Wireless Sensor Networks", Iowa, 2007

[11] 육의수, 윤성웅, 김성호, "센서 네트워크 시스템에 적용 가능한 오작동 검출 알고리즘 개발", 한국퍼지 및 지능시스템학회지 제17권 제6호, pp.760-765, 2007

[12] S.ranganathan, A.D. George, R.W. Todd, Matthew C. Chidester, "Gossip-Style Failure Detection and Distributed Consensus for Scalable Heterogeneous Clusters", HCS Research Laboratory, 2000.

[13] M. Young, The Technical Writer's Handbook, Mill Valley, Seoul, 1989.

[14] 이문노, 문정호, 정명진, "광 디스크 드라이버의 트랙킹 서보 시스템을 위한 다목적 강인제어기의 설계", 제어, 자동화, 시스템공학 논문지, vol.4 No.5 pp.592-599, 1998

[15] W. Heinzelman, A. Murphy, H. Carvalho, and M.Perillo, "Middleware to Support Sensor NetworkApplications," IEEE Network Magazine Special Issue, Jan. 2004.

[16] 이정은, 박현정, 박두경, 윤태복, 박교현, 이지형, "유비쿼터스 컴퓨팅 환경을 위한 상황 모델 정의 및 상황 인식 프레임워크 구현", 퍼지 및 지능 시스템학회 논문지 제16권 제4호, pp.389~527, 2006.8.

[17] 임재현, "지능형 서비스를 위한 상황인식 해석 구조 설계 및 구현",인터넷정보학회논문지, 제7권 제5호, pp.123~134, 2006.10

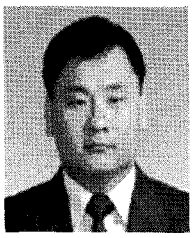
[18] Chang-Sun Shin, Myoung-Suk Kang, Chang-Won Jeong and Su-Chong Joo, "TMO-based Object Group Framework for Supporting Distributed Object Management and Real-Time Services", Lecture Notes in Computer Science, Vol.2834, pp.525-535. 2003.9.

[19] 신창선, 서종성, "분산객체그룹프레임워크 기반의 프로액티브 응용서비스엔진 개발", 한국인터넷정보학회논문지, 제11권 제1호, pp.153~165, 2010.2.

◐ 저 자 소 개 ◑

이 용 웅

2009년 순천대학교 정보통신공학과 학사
2008년~현재 순천대학교 정보통신공학과 석사
관심분야 : 분산컴퓨팅, USN
Email : kgcri@hanmail.net



김 세 한

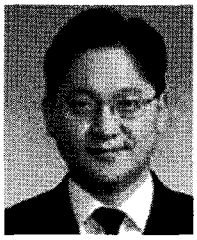
1998년 한국항공대학교 컴퓨터공학과 졸업(학사)
2000년 한국항공대학교 정보통신공학과 졸업(석사)
2007년 충남대학교 정보통신공학과 박사 수료
2000년~현재 한국전자통신연구원 선임연구원
관심분야 : RFID/USN
E-mail : shkim72@etri.re.kr

손 교 훈

1997년 창원대학교 제어계측공학과 졸업(학사)
1999년 창원대학교 대학원 전기전자제어공학과 졸업(석사)
2008년~현재 한국전자통신연구원 선임연구원
관심분야 : RFID/USN, Wibro, DMB, UWB etc.
E-mail : sonkh@etri.re.kr

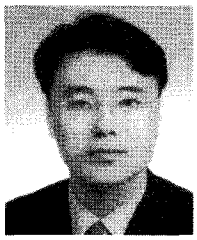
◐ 저 자 소 개 ◑

이 인 환



1988년 한양대학교 전기공학과 졸업(학사)
1990년 한양대학교 전기공학과 졸업(석사)
2011년 한양대학교 전자컴퓨터통신공학과 졸업(박사)
1990년 1월~1993년 3월 (주)동아전기 연구소 연구원
1993년~현재 한국전자통신연구원(ETRI) USN기반기술연구팀(팀장) 책임연구원
관심분야 : 무선통신, RFID, WSN Application
E-mail : ihlee@etri.re.kr

신 창 선



1996년 우석대학교 전산학과 학사
1999년 한양대학교 컴퓨터교육과 석사
1999년 원광대학교 컴퓨터공학과 박사
1999년 원광대학교 헬스케어기술개발센터 Post-Doc
2005년~현재 순천대학교 정보통신공학과 교수
순천대학교 공과대학 부학장
전라남도 IT 융합모델 기획위원
관심분야 : 분산컴퓨팅, 분산알고리즘
Email : csshin@sunchon.ac.kr