# 유비쿼터스 센서 네트워크에서 응용 프로그램 개발을 위한 모델 기반 통합 개발 도구

# (Model-based Integrated Development Tool for the Development of Applications in Ubiquitous Sensor Network)

정 기 원 † 김 주 일 †† 이 우 진 †††
(Kiwon Chong) (Juil Kim) (Woojin Lee)

**요 약** 본 논문에서는 센서 네트워크에서 응용 프로그램 개발을 지원하는 모델 기반의 통합 개발 도구를 제시한다. 제안하는 도구는 이클립스 플랫폼의 플러그인으로 개발되었으며, 사용자 그래픽 인터페이스, 모델 작성기, 설정정보 생성기, 모델 검증기, 소스코드 생성기 및 템플릿 저장소로 구성된다. 이 도구는 UML의 클래스다이어그램과 같은 표기법을 이용하여 구축하고자 하는 센서 네트워크를 모델링하고, 센서 네트워크 모델로부터 응용 프로그램 코드를 자동으로 생성할 수 있도록 한다. 개발자들은 도구를 이용하여 구현하고자 하는 센서 네트워크에 대한 모델을 작성하고, 모델에 속해 있는 각 센서들의 역할을 정의하여, 역할에 따른 속성값을 설정해 주기만 하면 각 센서들의 역할 수행을 위한 응용 프로그램이 자동으로 생성된다. 또한 작성한 모델에 대한 설계를 검증하여 오류를 조기에 발견하여 수정할 수 있도록 함으로써 고품질의 USN 응용 프로그램을 생성할 수 있다. 제안하는 도구를 이용하면, 개발자들은 하위레벨의 정보를 자세히 알지 못하더라도 다수의 센서 네트워크 응용 프로그램을 쉽고, 빠르게 구현할 수 있다.

**키워드** : 모델 기반 개발, 유비쿼터스 센서 네트워크, 통합 개발 도구

***Abstract*** A model-based integrated development tool for the development of USN application programs is proposed in this paper. The proposed tool has been implemented as a plug-in for Eclipse platform. The tool consists of Graphical User Interface, Modeler, Configuration Information Generator, Validity Checker, Source Code Generator and Templates Storage. Developers can implement USN applications from models of sensor networks using the tool. The developer can implement USN applications by automatic generation of execution code of each node in the sensor network after he/she designs a model of the sensor network. The configuration information of each node is automatically generated from the validated USN model. Then, the execution code is automatically generated using the configuration information and the predefined templates. Through the tool of this paper, developers can easily implement valid USN applications even if they do not know the details of low-level information. Also, a large number of application programs can be generated at once because application programs are generated from sensor network model instead of models of applications. Accordingly, the development effort of USN applications will be decreased and developers can consistently construct USN applications from USN models using the proposed tool.

**Key words** : Model-based Development, Ubiquitous Sensor Network, Integrated Development Tool

† 종신회원 : 숭실대학교 컴퓨터학부 교수
chong@ssu.ac.kr
†† 학생회원 : 숭실대학교대학원 컴퓨터학과
sespop@empal.com
††† 학생회원 : 한국정보통신대학교 공학부 Post Doc.
bluewj@empal.com
논문접수 : 2007년 9월 6일
심사완료 : 2007년 11월 26일


정보과학회논문지: 컴퓨팅의 실제 및 레터 제13권 제7호(2007.12)

## 1. 서 론

유비쿼터스 센서 네트워크(Ubiquitous Sensor Network: USN)는 유비쿼터스 컴퓨팅 구현을 위한 기반 네트워크로 초경량, 저전력의 많은 센서들로 구성된 무선 네트워크이다. 하나의 네트워크로 연결되어 있는 수많은 센서들이 필드(Field)의 지리적, 환경적 변화를 감지하여 베이스 스테이션으로 그 정보를 전달한 후 센서 네트워크 서버를 통해 사용자에게 전달되는 방식으로 정보 수집이 이루어진다. 유비쿼터스 센서 네트워크를 통해서 사물이 인간과 같은 다른 사물을 인식하고 주변 환경을 감지하게 하여, 사용자는 네트워크를 통해서 언제, 어느 곳에서든 정보를 확인하고 활용할 수 있다[1,2]. 센서 네트워크는 생산, 유통, 물류 같은 경제 활동, 의료 서비스, 복지 서비스, 환경 감시와 같은 다양한 분야의 서비스를 위한 응용 프로그램을 개발하는데 사용되고 있으며, 유비쿼터스 사회가 실현됨에 따라 사회 전반에 걸친 서비스를 위한 응용 프로그램을 개발하는데 사용될 것이다.

그러나 이러한 센서 네트워크 응용 프로그램을 작성하는 것은 어려운 작업이다. 센서 네트워크를 구성하는 노드들의 자원은 한정되어 있으며, 노드간의 무선통신은 신뢰하기 어려우며, 노드들은 저전력 오퍼레이션을 수행해야 하므로, 이러한 것들을 고려하여 응용 프로그램을 작성하는 것은 매우 어려운 작업이다. 이러한 이유로 필요한 때에 원하는 센서 네트워크 응용 프로그램을 작성하는 것이 쉽지 않다. 따라서 필요할 때에 쉽게 센서 네트워크 응용 프로그램을 작성할 수 있도록 하는 기술이 필요하다. 즉, 개발자들이 센서 네트워크에 대한 하위레벨의 정보를 자세히 알지 못하더라도 센서 네트워크 응용 프로그램을 쉽게 작성하도록 하는 방법이 필요하다. 또한 센서 네트워크는 수많은 센서 노드로 구성되므로, 센서 네트워크를 수행하려면 노드의 수만큼 응용 프로그램을 구현해야 한다. 따라서 많은 수의 응용 프로그램을 효율적으로 개발하기 위한 방법이 필요하다.

본 논문에서는 하위레벨의 정보를 자세히 알지 못하더라도 센서 네트워크 응용 프로그램을 쉽게 구축할 수 있도록 하는 프로그래밍 모델[3]에 대한 연구를 확장하여 하위레벨의 정보를 자세히 알지 못하더라도 센서 네트워크 응용 프로그램을 쉽게 구축할 수 있도록 지원하며, 다수의 응용 프로그램을 효율적으로 개발할 수 있도록 지원하는 도구를 제시한다. 이 도구는 UML의 클래스다이어그램과 같은 표기법을 이용하여 구축하고자 하는 센서 네트워크를 모델링하고, 센서 네트워크 모델로부터 응용 프로그램 코드를 자동으로 생성할 수 있도록 한다. 센서 네트워크 응용 프로그램은 특정 플랫폼에 종속적이며, 앞에서 설명한 것과 같이 여러 제약사항이 존재하므로 일반적인 개발도구를 이용해서 개발하는 데에는 한계가 있다. 따라서 특정 플랫폼에 적합한 센서 네트워크 응용 프로그램을 효율적으로 개발하도록 지원해주는 도구가 필요하다.

제안하는 도구를 이용하면, 개발자들은 다수의 센서 네트워크 응용 프로그램을 쉽고, 빠르게 구현할 수 있다. 개발자들은 도구를 이용하여 구현하고자 하는 센서 네트워크에 대한 모델을 작성하고, 모델에 속해 있는 각 센서들의 역할을 정의하여, 역할에 따른 속성값을 설정해 주기만 하면 각 센서들의 역할 수행을 위한 응용 프로그램이 자동으로 생성된다. 또한 작성한 모델에 대한 설계를 검증하여 오류를 조기에 발견하여 수정할 수 있도록 함으로써 고품질의 USN 응용 프로그램을 생성할 수 있다. 본 도구는 특히, 한국전자통신연구원(ETRI)에서 개발된 나노 큐플러스(Nano-Qplus)[4]를 기반으로 하는 센서 네트워크 응용 프로그램의 구축을 지원한다.

## 2. 센서 네트워크 응용 프로그램 개발을 위한 모델 기반의 통합 개발 도구

이 장에서는 본 논문에서 제안하는 모델 기반의 센서 네트워크 응용 프로그래밍의 개념, 통합 개발 도구의 구조 및 프로그래밍 절차를 설명한다.

### 2.1 USN 응용 프로그래밍의 개념

그림 1은 본 논문에서 제안하는 모델 기반의 센서 네트워크 응용 프로그래밍의 개념을 나타낸 것이다. 본 논문에서 제안하는 센서 네트워크 응용 프로그래밍의 특징으로는 첫째, 센서 네트워크 모델로부터 센서 네트워크를 구성하는 다수의 노드를 위한 응용 프로그램을 한 번에 생성할 수 있다는 것이다. 기존의 프로그래밍 기법들은 센서 네트워크를 구성하는 여러 개의 노드들 중에서 하나의 노드를 위한 응용 프로그램을 생성하기 위한 모델을 작성하기 때문에, 노드의 수만큼 모델을 작성해야 한다. 이에 반해, 제안하는 방법은 센서 네트워크에 대한 모델을 작성하고, 그로부터 센서 네트워크에 속한 모든 노드를 위한 응용 프로그램을 생성하는 것이므로, 하나의 모델만 작성하면 된다. 두 번째 특징은 모델 작성시에 응용 프로그램의 생성을 위해 미리 정의되어 있는 속성에 대한 값을 결정하기만 하면, 추가적인 설계 작업을 수행하지 않고서도 응용 프로그램을 자동으로 생성할 수 있다는 것이다. 일반적인 프로그래밍 방법에서는 모델링 단계에서 응용 프로그램을 위해 필요한 속성과 오퍼레이션을 결정하고, 실제 구현 시에 각 속성에 대한 값을 결정하도록 소스코드를 작성하게 된다. 그러나 제안하는 방법에서는 응용 프로그램을 위해 필요한 속성은 미리 결정되어 있고, 모델링 단계에서는 그 속성에 대한 값을 설정함으로써 소스코드를 작성하는 단계

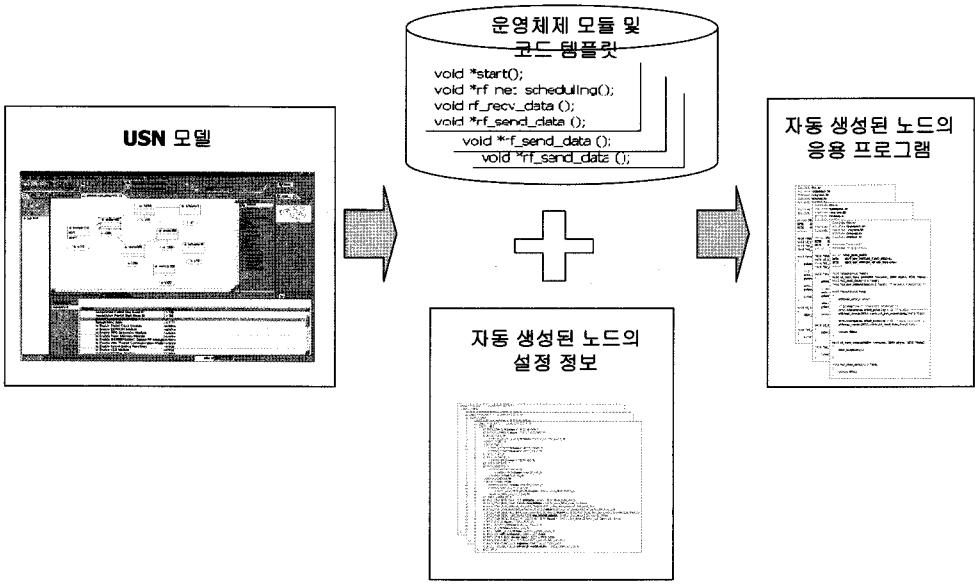



그림 1 USN 프로그래밍의 개념

를 미리 수행하는 것과 같은 역할을 한다. 이러한 방법이 가능한 이유는 센서 네트워크 응용 프로그램의 크기는 매우 작으며, 노드들이 네트워크 상에서 수행하는 기능들이 한정되어 있기 때문에 필요한 기능들을 미리 만들어 놓을 수 있기 때문이다. 따라서 모델링 단계에서 속성값을 선택하는 것은 미리 구현되어 있는 기능들을 선택하는 것과 같다고 할 수 있다.

제안하는 방법으로 센서 네트워크 응용 프로그램을 개발하기 위해서는, 우선 구축하고자 하는 센서 네트워크 모델을 설계하고, 모델을 구성하는 각 센서 노드들의 역할에 따라 응용 프로그램을 생성하기 위해 필요한 속성값들을 설정한다. 제안하는 도구는 설계한 모델로부터 각 센서 노드의 응용 프로그램을 생성하는데 필요한 설정정보를 자동 생성하고, 설정정보에 따라 미리 정의되어 있는 코드 템플릿과 타겟 운영체제의 모듈을 이용하여 응용 프로그램의 소스코드를 생성한다. 이와 같은 방법으로 센서 네트워크의 수행을 위해 필요한 수많은 응용 프로그램은 센서 네트워크 모델로부터 자동으로 생성할 수 있다.

### 2.2 통합 개발 도구의 구조

본 논문에서 제안하는 통합 개발 도구는 이클립스 플랫폼의 플러그인으로 구현된다. 개발자들은 통합 개발 도구를 이용하여 센서 네트워크를 모델링하고, 모델로부터 센서 네트워크를 구성하는 노드들이 필요로 하는 응용 프로그램을 자동으로 생성할 수 있다. 제안하는 도구는 센서 네트워크 운영체제가 미리 구현해 놓은 모듈을 조합하여 응용 프로그램을 생성하는 것으로 도구를 이용하지 않고 응용 프로그램을 생성할 때와 같은 방식으로 프로그램을 생성한다. 따라서 도구를 이용하지 않고 프로그램을 작성할 때와 거의 비슷한 크기로 소스코드를 생성하게 되므로, 자동 생성을 하더라도 일반적인 자동화 도구와 같이 코드의 크기가 증가하지 않아 한정된 자원을 이용하는 센서 네트워크를 수행하기에 적합한 응용 프로그램을 생성할 수 있다.

통합 개발 도구는 그래픽 사용자 인터페이스, 모델 작성기, 모델 검증기, 설정정보 생성기, 소스코드 생성기 및 템플릿 저장소로 구성되어 있다. 그림 2는 제안하는 통합 개발 도구의 구조를 보여주는 것이다. 도구를 구성하는 각각의 모듈에 대한 자세한 설명은 다음과 같다.

- 그래픽 사용자 인터페이스(GUI): 그래픽 사용자 인터페이스는 USN 응용 프로그램을 구축하기 위하여 사

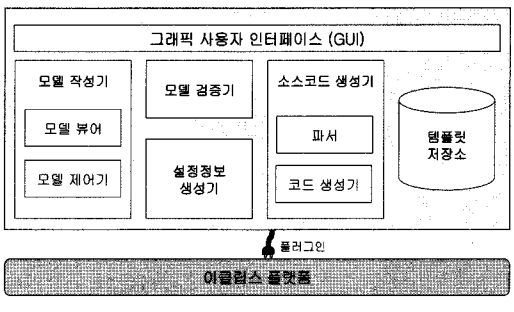



그림 2 통합 개발 도구의 구조

용자에게 제공되는 인터페이스이다.

- 모델 작성기: 개발자들은 모델 작성기를 통하여 구축하고자 하는 센서 네트워크를 모델링할 수 있다. 모델 작성기는 모델 뷰어 및 모델 제어기로 구성된다. 모델 뷰어는 센서 네트워크 모델을 그래픽으로 표현해주기 위한 모듈이며, 모델 제어기는 모델의 정보와 뷰를 관리하고, 모델에 대한 정보를 XML로 생성하기 위한 모듈이다. 모델 작성기를 통하여 개발자들은 구축하고자 하는 USN에 대한 모델 다이어그램을 작성하고, USN 모델의 각 센서 노드에 대한 속성 설정을 통하여 각 센서 노드의 역할을 정의하고 필요한 모듈을 선택하도록 한다. 이때 설정한 정보에 따라 각 노드의 프로그램을 생성할 때 필요한 운영체제의 모듈이 결정된다.
- 모델 검증기: 모델 검증기는 센서 네트워크 모델의 설계가 올바르게 되어 있는가를 검증한다. 모델 검증의 결과로 모델에 대한 오류를 발견하면, 모델 작성기를 통해 모델에 대한 설계를 수정함으로써 고품질의 응용 프로그램을 생성하는데 기여할 수 있다.
- 설정정보 생성기: 설정정보는 모델에 있는 각 노드들에 대한 속성 설정 값을 바탕으로 각 노드마다 생성된다. 모델 작성기에서 작성하고, 모델 검증기를 통하여 검증한 센서 네트워크 모델 정보를 바탕으로 센서 네트워크를 구성하는 각 노드의 역할에 따른 응용 프로그램을 구축하는데 필요한 정보를 생성한다.
- 소스코드 생성기: 소스코드 생성기는 설정정보 생성기를 통해 생성된 설정정보를 이용하여, 각 노드의 실행을 위해 필요한 응용 프로그램 코드를 생성한다. 소스코드 생성기는 설정정보를 분석하는 파서와 소스코드를 생성하는 코드 생성기로 구성된다. 소스코드 생성기가 생성하는 응용 프로그램 코드는 C 코드로 생성된다.
- 템플릿 저장소: 템플릿 저장소는 응용 프로그램을 자동으로 생성하기 위해 미리 정의해 놓은 템플릿과 운영체제 모듈을 저장해 놓은 곳이다.

### 2.3 통합 개발 도구를 이용한 USN 응용 프로그래밍의 절차

그림 3은 제안하는 통합 개발 도구를 이용하여 센서 네트워크 응용 프로그램을 생성하기 위한 절차를 보여준다.

각 단계에 대하여 자세하게 설명하면 다음과 같다.

- Step 1 - 구축하고자 하는 센서 네트워크를 모델링한다.
- Step 2 - 속성값의 설정을 통하여 센서 네트워크 모델을 구성하는 각 노드의 역할을 정의하고, 정의한 역할을 수행하는데 필요한 값들을 설정한다.
- Step 3 - 설계한 센서 네트워크 모델을 검증한다.

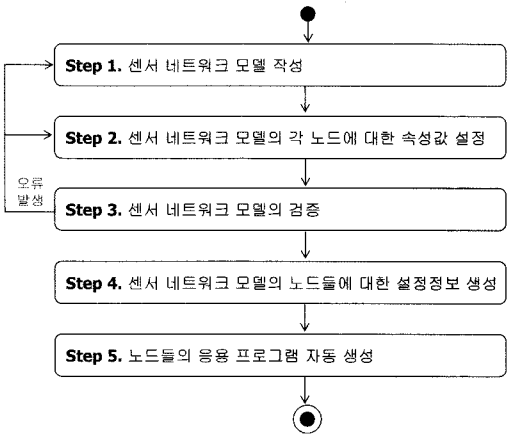



그림 3 센서 네트워크 응용 프로그래밍 절차

Step 1과 Step 2에서 설계한 정보를 바탕으로 센서 네트워크 모델이 수행 가능하도록 설계되었는지 검증한다. 만약 모델에 오류가 있다면, Step 1이나 Step 2로 돌아가서 오류를 수정하고 다시 검증한다.

- Step 4 - 검증된 센서 네트워크 모델로부터 응용 프로그램을 생성하는데 필요한 설정정보를 생성한다. 응용 프로그램은 센서 네트워크를 구성하는 각 노드의 수만큼 생성되어야 하므로, 모델을 구성하는 노드들에 대하여 설정정보를 각각 생성한다.
- Step 5 - Step 4에서 생성한 설정정보를 바탕으로 센서 네트워크를 구성하는 각 노드를 위한 응용 프로그램을 생성한다.

### 2.4 도구의 구현

본 논문에서 제시하는 통합 개발 도구는 이클립스의 플러그인[5]으로 개발하였으며, 센서 네트워크의 모델링을 위해서 GMF(Eclipse Graphical Modeling Framework)[6] 플러그인을 사용하여 개발하였다. 구현한 통합 개발 도구의 실행화면은 그림 4와 같다. 센서 네트워크 모델은 모델링 윈도우에서 작성하고, 센서 네트워크를 구성하는 각 노드에 대한 속성값은 속성 설정 윈도우에서 설정할 수 있다.

## 3. 제안하는 도구를 이용한 센서 네트워크 응용 프로그래밍

이 장에서는 나노 큐플러스[4] 운영체제를 기반으로 한 가스 및 조도 모니터링 시스템의 예를 통하여 논문에서 제안하는 통합 개발 도구를 이용한 센서 네트워크 응용 프로그래밍 방법을 설명한다. 가스 및 조도 모니터링 시스템은 가스 및 조도 데이타를 감지하는 센서 노드, 감지한 데이타를 센서 노드로부터 전달받아 싱크 노드로 전송하는 라우터 노드, 그리고 라우터 노드로부터

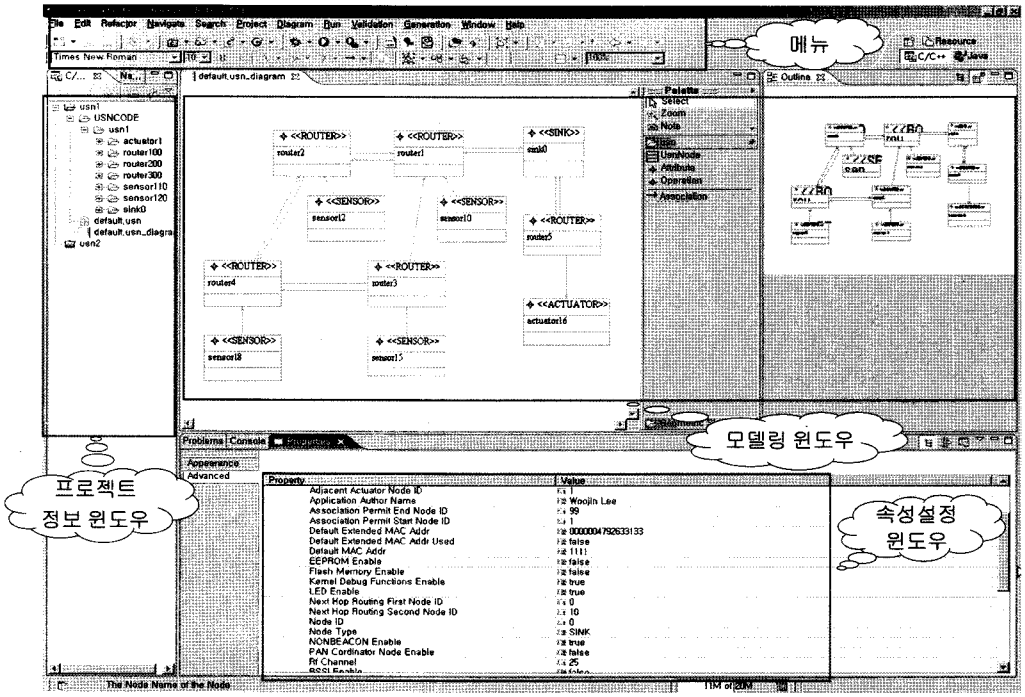



그림 4 통합 개발 도구 화면

표 1 센서 네트워크 모델 작성을 위한 표기법

| 이름 | 표기법 | 설명 |
|---|---|---|
| 노드 | <<노드 타입>><br>노드명<br>속성<br>오퍼레이션 | 센서 네트워크를 구성하는 각 노드를 표현하는 것으로, UML의 클래스에 해당 |
| 노드 타입 | <<SENSOR>> | 센서 역할을 수행하는 노드를 나타내는 스테레오 타입 |
| | <<ROUTER>> | 라우터 역할을 수행하는 노드를 나타내는 스테레오 타입 |
| | <<SINK>> | 싱크 역할을 수행하는 노드를 나타내는 스테레오 타입 |
| | <<ACTUATOR>> | 엑츄에이터 역할을 수행하는 노드를 나타내는 스테레오 타입 |
| 관계 | ⟶ | 클래스 간의 관계 |

데이타를 전달받아 임계값에 따른 행동명령을 결정하는 싱크 노드, 싱크 노드로부터 행동명령을 전달받아 그에 따른 행동을 취하는 엑츄에이터 노드로 구성된다.

### 3.1 센서 네트워크 모델 작성

제안하는 통합 개발 도구를 이용하여 구축하고자 하는 센서 네트워크 모델은 표 1과 같은 노드, 노드 타입, 관계의 3가지 요소를 이용하여 작성할 수 있다. 이 요소들은 UML의 클래스다이어그램의 요소로부터 채택하였다. 이와 같은 표기법을 사용하여 그림 5(a)와 같이 가스 및 조도 모니터링 시스템을 위한 센서 네트워크를 모델링할 수 있다.

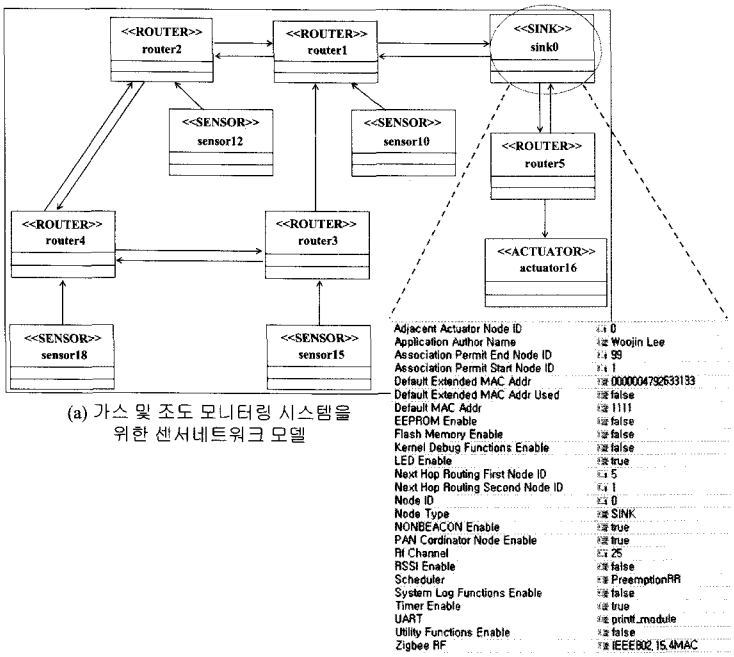
### 3.2 속성값 설정

모델로부터 응용 프로그램을 생성하기 위해서는 센서 네트워크를 구성하는 각 노드의 역할을 정의하고, 역할에 따라 필요한 속성값을 설정해야 한다.

표 2는 나노 큐플러스를 기반으로 한 응용 프로그램을 생성하기 위해 설정해 주어야 하는 노드의 속성을 종류별로 분류해 놓은 것이다. RF 통신 분류에 속해있는 associationPermitStartNodeID, association Permit-EndNodeID, nextHopRoutingFirstNode ID, nextHop-

표 2 노드의 속성

| 속성 분류 | 설명 |
|---|---|
| RF 통신 | 센서 네트워크에서 RF 통신을 수행하는 데 필요한 정보를 설정하기 위한 속성 분류로, nodeID, adjacentActuatorNodeID, PAN_Cordinator_Node_Enable, NON_BEACON_Enable, defaultMACAddr, Default_Extended_MAC_Addr_Used, defaultExtendedMACAddr, association PermitStartNodeID, associationPermitEndNodeID, nextHopRouting FirstNodeID, nextHopRoutingSecondNodeID, rfChannel, scanChannel, Zigbee RF 등이 포함된다. |
| 스케줄러 | 스케줄러의 종류를 설정하기 위한 속성 분류로 Scheduler가 포함된다. |
| 디바이스 드라이버 | 각 센서 노드의 디바이스 드라이버를 생성하기 위해 필요한 속성 분류로, EEPROM_Enable, Flash_Memory_Enable, Timer_Enable, UART, LED_Enable, RSSI_Enable 등이 포함된다. |
| 센서 유형 | 센서의 유형을 결정하기 위한 속성 분류로, Sensor_Battery_Enable, Sensor_Temperature_Enable, Sensor_Light_Enable, Sensor_Gas_Enable, Sensor_Point_infra_red_Enable, Sensor_Humidity_Enable, Sensor_Ultra_sonic_Enable 등이 포함된다. |




(a) 가스 및 조도 모니터링 시스템을 위한 센서네트워크 모델

(b) 싱크 노드의 속성값 설정

그림 5 가스 및 조도 모니터링 시스템을 위한 센서 네트워크 모델링

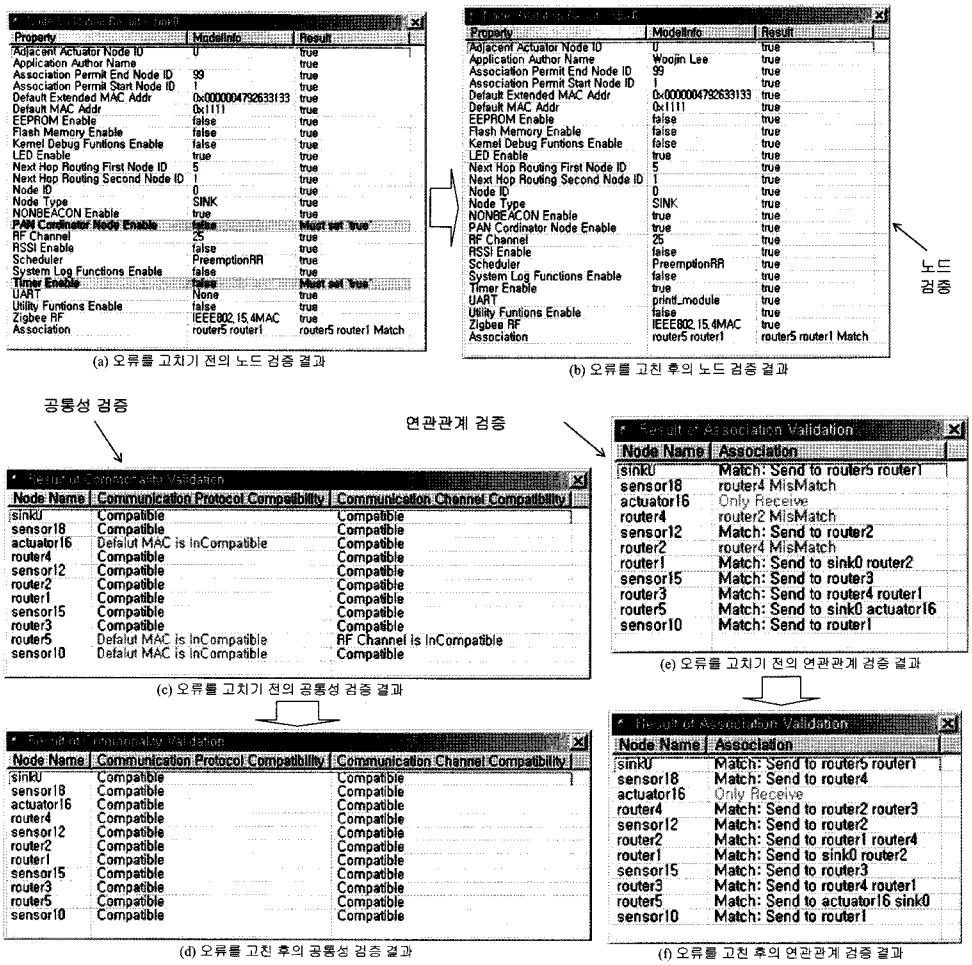
RoutingSecondNodeID 및 rfChannel 속성은 싱크와 라우터 역할을 수행하는 노드의 응용 프로그램을 생성하기 위해 필요한 속성이며, scanChannel 속성은 센서와 엑츄에이터의 역할을 수행하는 노드의 응용 프로그램을 생성하기 위해 사용된다. 센서 유형은 센서 역할을 수행하는 노드의 응용 프로그램을 생성하기 위해 사용되는 속성들을 포함한다.

그림 5(b)는 가스 및 조도 모니터링 시스템을 구성하는 싱크 노드의 속성과 그 설정값을 보여준다. 속성값 설정을 통하여 센서 네트워크 모델을 구성하는 각 노드의 역할에 따라 스케줄러 타입, 네트워크 토폴로지, 노드의 타입 등을 결정한다.

### 3.3 센서 네트워크 모델 검증

모델이 잘못되어 있다면, 프로그램을 구현하는 과정에서 오류가 생길 수 있고, 개발 후기로 갈수록 오류를 발견하여 수정하는데 비용이 많이 든다. 따라서 고품질의 응용 프로그램을 효율적으로 생성하기 위해서는 코드 생성 이전에 모델을 검증할 필요가 있다. 이를 위하여 제안하는 통합 개발 도구는 설계한 센서 네트워크 모델에 대한 검증기능을 제공한다.

제안하는 통합 개발 도구를 이용하면, 설계한 센서 네트워크 모델은 크게 3가지 측면에서 검증할 수 있다. 첫째는, 모델에 나타나 있는 모든 노드들이 공통적인 값을 가져야 하는 속성에 대하여 일치하도록 설계되었는가를 검증하는 것으로, 노드 간에 통신 프로토콜이나 통신 채널과 같은 것들이 서로 호환되는가를 검증할 수 있다.




(a) 오류를 고치기 전의 노드 검증 결과

| Property | ModelInfo | Result |
|---|---|---|
| Adjacent Actuator Node ID | 0 | true |
| Application Author Name | | true |
| Association Permit End Node ID | 99 | true |
| Association Permit Start Node ID | 1 | true |
| Default Extended MAC Addr | 0x0000004792633133 | true |
| Default MAC Addr | 0x1111 | true |
| EEPROM Enable | false | true |
| Flash Memory Enable | false | true |
| Kernel Debug Funtions Enable | false | true |
| LED Enable | true | true |
| Next Hop Routing First Node ID | 5 | true |
| Next Hop Routing Second Node ID | 1 | true |
| Node ID | 0 | true |
| Node Type | SINK | true |
| NONBEACON Enable | true | true |
| PAN Cordinator Node Enable | false | Must set 'true' |
| RF Channel | 25 | true |
| RSSI Enable | false | true |
| Scheduler | PreemptionRR | true |
| System Log Functions Enable | false | true |
| Timer Enable | false | Must set 'true' |
| UART | None | true |
| Utility Funtions Enable | false | true |
| Zigbee RF | IEEE802.15.4MAC | true |
| Association | router5 router1 | router5 router1 Match |

(b) 오류를 고친 후의 노드 검증 결과

| Property | ModelInfo | Result |
|---|---|---|
| Adjacent Actuator Node ID | 0 | true |
| Application Author Name | Woojin Lee | true |
| Association Permit End Node ID | 99 | true |
| Association Permit Start Node ID | 1 | true |
| Default Extended MAC Addr | 0x0000004792633133 | true |
| Default MAC Addr | 0x1111 | true |
| EEPROM Enable | false | true |
| Flash Memory Enable | false | true |
| Kernel Debug Funtions Enable | false | true |
| LED Enable | true | true |
| Next Hop Routing First Node ID | 5 | true |
| Next Hop Routing Second Node ID | 1 | true |
| Node ID | 0 | true |
| Node Type | SINK | true |
| NONBEACON Enable | true | true |
| PAN Cordinator Node Enable | true | true |
| RF Channel | 25 | true |
| RSSI Enable | false | true |
| Scheduler | PreemptionRR | true |
| System Log Functions Enable | false | true |
| Timer Enable | true | true |
| UART | printf_module | true |
| Utility Funtions Enable | false | true |
| Zigbee RF | IEEE802.15.4MAC | true |
| Association | router5 router1 | router5 router1 Match |

(c) 오류를 고치기 전의 공통성 검증 결과

| Node Name | Communication Protocol Compatibility | Communication Channel Compatibility |
|---|---|---|
| sink0 | Compatible | Compatible |
| sensor18 | Compatible | Compatible |
| actuator16 | Defalut MAC is InCompatible | Compatible |
| router4 | Compatible | Compatible |
| sensor12 | Compatible | Compatible |
| router2 | Compatible | Compatible |
| router1 | Compatible | Compatible |
| sensor15 | Compatible | Compatible |
| router3 | Compatible | Compatible |
| router5 | Defalut MAC is InCompatible | RF Channel is InCompatible |
| sensor10 | Defalut MAC is InCompatible | Compatible |

(d) 오류를 고친 후의 공통성 검증 결과

| Node Name | Communication Protocol Compatibility | Communication Channel Compatibility |
|---|---|---|
| sink0 | Compatible | Compatible |
| sensor18 | Compatible | Compatible |
| actuator16 | Compatible | Compatible |
| router4 | Compatible | Compatible |
| sensor12 | Compatible | Compatible |
| router2 | Compatible | Compatible |
| router1 | Compatible | Compatible |
| sensor15 | Compatible | Compatible |
| router3 | Compatible | Compatible |
| router5 | Compatible | Compatible |
| sensor10 | Compatible | Compatible |

(e) 오류를 고치기 전의 연관관계 검증 결과

| Node Name | Association |
|---|---|
| sink0 | Match: Send to router5 router1 |
| sensor18 | router4 MisMatch |
| actuator16 | Only Receive |
| router4 | router2 MisMatch |
| sensor12 | Match: Send to router2 |
| router2 | router4 MisMatch |
| router1 | Match: Send to sink0 router2 |
| sensor15 | Match: Send to router3 |
| router3 | Match: Send to router4 router1 |
| router5 | Match: Send to sink0 actuator16 |
| sensor10 | Match: Send to router1 |

(f) 오류를 고친 후의 연관관계 검증 결과

| Node Name | Association |
|---|---|
| sink0 | Match: Send to router5 router1 |
| sensor18 | Match: Send to router4 |
| actuator16 | Only Receive |
| router4 | Match: Send to router2 router3 |
| sensor12 | Match: Send to router2 |
| router2 | Match: Send to router1 router4 |
| router1 | Match: Send to sink0 router2 |
| sensor15 | Match: Send to router3 |
| router3 | Match: Send to router4 router1 |
| router5 | Match: Send to actuator16 sink0 |
| sensor10 | Match: Send to router1 |

그림 6 가스 및 조도 모니터링 시스템을 위한 센서 네트워크 모델 검증

둘째는, 각 노드간의 연관 관계가 올바르게 설계되었는가를 검증하는 것으로, 적절한 경로를 통하여 감지된 데이타가 서버로 전송되는가를 검증할 수 있다. 마지막으로, 각 노드에 대하여 역할에 맞도록 속성 설정이 올바르게 되었는가를 검증할 수 있는데, 노드의 역할이나 타겟 플랫폼에 따른 제약사항을 반영하여 올바르게 설계되었는가를 검증할 수 있다. 모델의 공통성 및 연관 관계에 대한 검증은, 센서 네트워크를 설계한 후에 전체 모델에 대한 검증 메뉴를 수행하면, 모델에 대한 정보를 분석하여 공통적인 속성에 대한 값과 네트워크의 연관 관계가 올바르게 설계되었는가를 검사한다. 각 노드에 대한 검증은, 센서 네트워크 모델에서 검증하고자 하는 노드에 해당하는 클래스를 선택한 후 노드 검증 메뉴를 수행하면, 해당 노드의 속성에 대한 설정값을 분석하여 역할에 맞도록 올바르게 설정 되었는가를 검사한다.

그림 6은 가스 및 조도 모니터링 시스템의 구현을 위해 설계한 센서 네트워크 모델을 검증한 결과를 보여준다. 앞에서 설명한 3가지 측면에 대하여 오류가 생긴 경우의 검증 결과와 오류를 수정한 후의 검증 결과를 볼 수 있다.

### 3.4 설정정보 생성

제안하는 통합 개발 도구를 이용하여 설계한 센서 네트워크 모델에 대한 정보는 XML로 저장이 된다. 그러나 이 정보는 센서 네트워크 모델 전체에 대한 정보이므로, 센서 네트워크를 구성하는 각 노드를 위한 응용 프로그램을 생성하기 위해서는 모델 정보로부터 각 노드에 대한 정보를 추출해야 한다. 통합 개발 도구의 설정정보 생성기는 다음과 같은 절차를 통하여 센서 네트워크 모델 정보로부터 각 노드에 대한 정보를 추출하고, 추출한 정보를 XML로 저장한다.

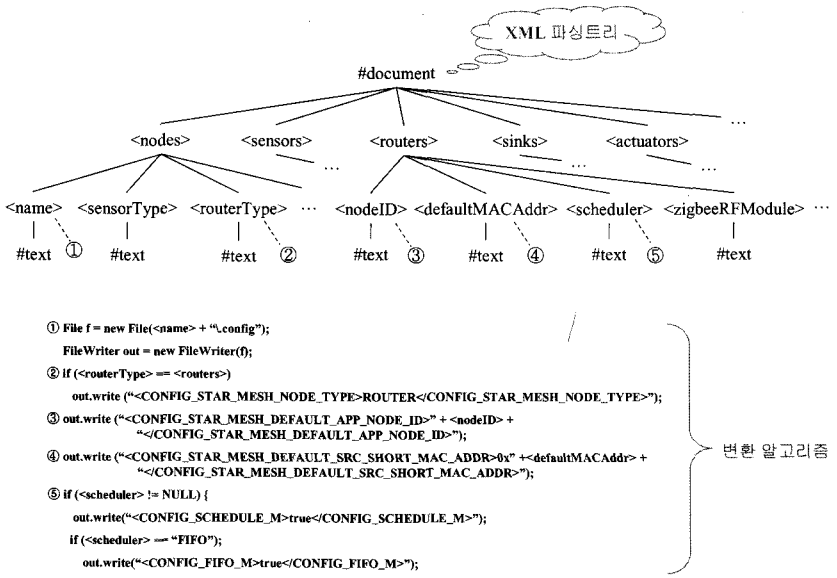



그림 7 설정정보 생성 알고리즘

```
<?xml version="1.0" encoding="UTF-8"?>
<CONFIG_INFO>
    <CONFIG_EEPROM_M>false</CONFIG_EEPROM_M>
    <CONFIG_FLASHMEM_M>false</CONFIG_FLASHMEM_M>
    <CONFIG_TIMER_M>true</CONFIG_TIMER_M>
    <CONFIG_UART_M>
        <CONFIG_PRINTF_M>true</CONFIG_PRINTF_M>
        <CONFIG_SCANF_M>false</CONFIG_SCANF_M>
    </CONFIG_UART_M>
    <CONFIG_ACTUATOR_M>
        <CONFIG_LED_M>true</CONFIG_LED_M>
    </CONFIG_ACTUATOR_M>
    <CONFIG_SCHEDULE_M>
        <CONFIG_PREEMPTION_RR_M>
        </CONFIG_PREEMPTION_RR_M>
    </CONFIG_SCHEDULE_M>
    <CONFIG_ZIGBEE_RF_M>
        <CONFIG_IEEE_802_15_4_MAC_M>
            <CONFIG_STAR_MESH_ROUTE_M>true</CONFIG_STAR_MESH_ROUTE_M>
        </CONFIG_IEEE_802_15_4_MAC_M>
    </CONFIG_ZIGBEE_RF_M>
    <CONFIG_STAR_MESH_NODE_TYPE>SINK</CONFIG_STAR_MESH_NODE_TYPE>
    <CONFIG_STAR_MESH_DEFAULT_RF_CHANNEL>25</CONFIG_STAR_MESH_DEFAULT_RF_CHANNEL>
    <CONFIG_STAR_MESH_DEFAULT_APP_NODE_ID>0</CONFIG_STAR_MESH_DEFAULT_APP_NODE_ID>
    <CONFIG_STAR_MESH_ADJACENT_ACTUATOR_NODE_ID>0</CONFIG_STAR_MESH_ADJACENT_ACTUATOR_NODE_ID>
    <CONFIG_STAR_MESH_THIS_NODE_PAN_COORDINATION_ENABLE>true</CONFIG_STAR_MESH_THIS_NODE_PAN_COORDINATION_ENABLE>
    <CONFIG_STAR_MESH_COORDINATOR_TYPE>NON_BEACON_ENABLE</CONFIG_STAR_MESH_COORDINATOR_TYPE>
    <CONFIG_STAR_MESH_DEFAULT_SRC_SHORT_MAC_ADDR>0x1111</CONFIG_STAR_MESH_DEFAULT_SRC_SHORT_MAC_ADDR>
    <CONFIG_STAR_MESH_USE_DEFAULT_EXTENDED_MAC_ADDR>0x0000004792633133</CONFIG_STAR_MESH_USE_DEFAULT_EXTENDED_MAC_ADDR>
    <CONFIG_STAR_MESH_NEXT_HOP_ROUTING_FIRST_NODE>5</CONFIG_STAR_MESH_NEXT_HOP_ROUTING_FIRST_NODE>
    <CONFIG_STAR_MESH_NEXT_HOP_ROUTING_SECOND_NODE>1</CONFIG_STAR_MESH_NEXT_HOP_ROUTING_SECOND_NODE>
    <CONFIG_START_MESH_ASSOCIATION_PERMIT_NODEID_START>1</CONFIG_START_MESH_ASSOCIATION_PERMIT_NODEID_START>
    <CONFIG_START_MESH_ASSOCIATION_PERMIT_NODEID_END>99</CONFIG_START_MESH_ASSOCIATION_PERMIT_NODEID_END>
    <CONFIG_RSSI_M>false</CONFIG_RSSI_M>
    <CONFIG_UTILITY_M>false</CONFIG_UTILITY_M>
    <CONFIG_LOG_M>false</CONFIG_LOG_M>
    <CONFIG_KERNEL_DEBUG_M>false</CONFIG_KERNEL_DEBUG_M>
    <CONFIG_APP_NAME>sink0</CONFIG_APP_NAME>
    <CONFIG_AUTHOR_NAME>Woojin Lee</CONFIG_AUTHOR_NAME>
    <CONFIG_ADDITIONAL_INCLUDE>-I./</CONFIG_ADDITIONAL_INCLUDE>
    <CONFIG_ADDITIONAL_LIB>-lqplusn</CONFIG_ADDITIONAL_LIB>
    <CONFIG_COMPILER_FLAGS>-DQPLUS_N -DATMEGA128</CONFIG_COMPILER_FLAGS>
</CONFIG_INFO>
```

그림 8 가스 및 조도 모니터링 시스템을 구성하는 싱크 노드의 설정정보

- Step 1 - 모델 정보를 파싱하여 파싱트리를 생성한다.
- Step 2 - 파싱트리로부터 각 노드에 대한 정보를 추출한다.
- Step 3 - 각 노드에 대한 파싱트리의 정보를 각 노드에 대한 설정정보로 변환하여 XML 파일로 저장한다.

그림 7은 모델의 정보를 파싱하여 각 노드의 설정정보를 생성하는 알고리즘을 도식화한 것이다. 설계한 모델을 저장하여 생성되는 XML 파일을 파싱하면, 그림의 윗부분에 있는 것과 같은 XML 파싱트리가 생성이 된다. 이 파싱트리의 <nodes> 아래에 있는 정보가 응용 프

로그램 명, 노드의 유형을 나타내는 것이고, <nodeID>, <defaultMACAddr>과 같은 부분의 값이 응용 프로그램의 생성을 위해 필요한 속성값을 나타낸다. 따라서 그림의 아랫부분에 있는 변환알고리즘을 통하여 XML 파싱트리로부터 응용 프로그램의 생성을 위한 설정정보를 추출할 수 있다.

그림 8은 이와 같은 알고리즘을 이용하여 통합 개발도구의 설정정보 생성기를 통하여 생성된 가스 및 조도 모니터링 시스템을 구성하는 싱크 노드의 설정정보를 보여준다. 설정정보는 XML 파일로 생성이 되며, 모든 태그는 <CONFIG_XXX>와 같은 방법으로 생성된다. 여기에서 XXX 부분이 속성을 나타내며, 각 태그 사이에 있는 텍스트가 그 속성의 값을 나타낸다. 예를 들어 그림 8을 살펴보면, <CONFIG_TIMER_M> 태그와 </CONFIG_ TIMER_M> 태그 사이에 값이 "true"로 되어 있는데, 이것은 TIMER_M 속성의 값이 "true"로 설정되어 있다는 것을 말하는 것이다.

### 3.5 응용 프로그램 생성

그림 9는 센서 네트워크를 구성하는 각 노드에 대한 설정정보로부터 각 노드를 위한 응용 프로그램을 생성하는 알고리즘을 도식화한 것이다. 제안하는 통합 개발도구의 소스코드 생성기는 그림 9와 같은 알고리즘을 통하여 응용 프로그램을 생성한다. 응용 프로그램을 생성하기 위해서는 우선, 그림 8과 같은 각 노드에 대한 설정정보를 분석하여 노드의 타입 및 속성 설정값을 추출한다. 그리고 나서 추출한 정보를 바탕으로 헤더, 데이타, 함수 및 Main 코드를 생성하고, 이를 하나로 통합하여 각 노드를 위한 응용 프로그램을 생성한다. 이때, 각 노드의 응용 프로그램을 자동으로 생성하기 위하여 노드의 유형 및 속성값에 따른 헤더, 데이타, 함수의 코드는 미리 개발되어 템플릿 저장소에 저장되어 있다. 따라서 프로그램을 생성하는 알고리즘에서는 각 노드의 템플릿을 바탕으로 속성값에 따라 필요한 헤더, 데이타, 함수를 해쉬 테이블을 통해서 템플릿 저장소로부터 읽어와서 응용 프로그램을 생성하는 것이다.

표 3 해쉬 테이블의 구조

| 키 | 값 |
|---|---|
| Zigbee_Simple | "&1_Zig_Simple_&2" |
| Zigbee_MAC | "&1_Zig_MAC_&2" |
| Zigbee_MAC_StarMesh | "&1_Zig_StarMesh_&2" |
| Scheduler_FIFO | "&1_Sche_FIFO_&2" |
| Scheduler_PreemptionRR | "&1_Sche_PreemptionRR_&2" |
| Sensor_LIGHT | "&1_Sensor_LIGHT_&2" |
| Sensor_GAS | "&1_Sensor_GAS_&2" |
| Sensor_Temperature | "&1_Sensor_Temperature_&2" |

응용 프로그램을 생성하는 과정에서 사용되는 해쉬 테이블의 구조는 표 3과 같다. 해쉬 테이블의 구조에서 키의 값은 응용 프로그램을 자동으로 생성하고자 할 경우에 각 속성값에 따른 실제 코드가 저장되어 있는 파일의 이름이다. 키 값에서 &1, &2라는 문자열은 노드 및 호출되는 모듈의 유형에 따라서 동적으로 변경되는 부분이다. 각 노드의 유형이 결정되면 &1이 노드 유형으로 대체되고, 소스코드에서 필요한 모듈의 유형에 따라 &2가 변경된다. 모듈의 헤더, 데이타, 함수 코드 중 선택되는 것에 따라 &2는 "H', "D", "F" 로 대체된다. 이와 마찬가지로 응용 프로그램의 Main 함수를 생성하기 위한 해쉬 테이블도 따로 정의되어 있다.

그림 10은 이와 같은 알고리즘을 이용하여 통합 개발도구의 소스코드 생성기를 통하여 생성된 가스 및 조도 모니터링 시스템을 구성하는 싱크 노드를 위한 응용 프로그램을 보여준다.

## 4. 관련연구와의 비교

기존에 USN 응용 프로그램을 구축하기 위한 도구로는 TinyOS Plugin과 NanoEsto가 있다. TinyOS Plugin[7]은 TinyOS 기반의 센서 네트워크 응용 프로그램 생성을 위한 도구로 이클립스 기반에서 사용되는 플러그인 형태로 개발된 것이고, NanoEsto[8]는 나노 큐플러스 기반의 센서 네트워크 응용 프로그램 구축을 지원하는 도구로 TinyOS Plugin과 마찬가지로 이클립스 기

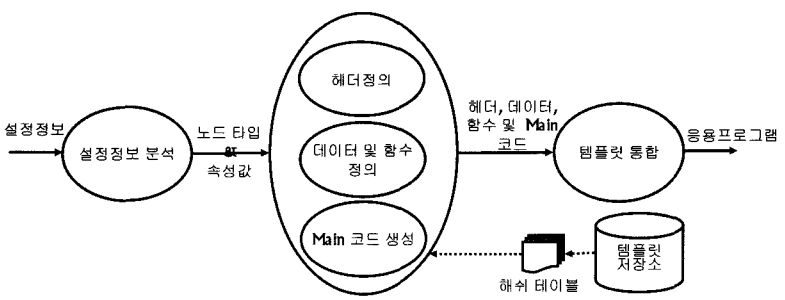



그림 9 각 노드를 위한 응용 프로그램 생성 과정

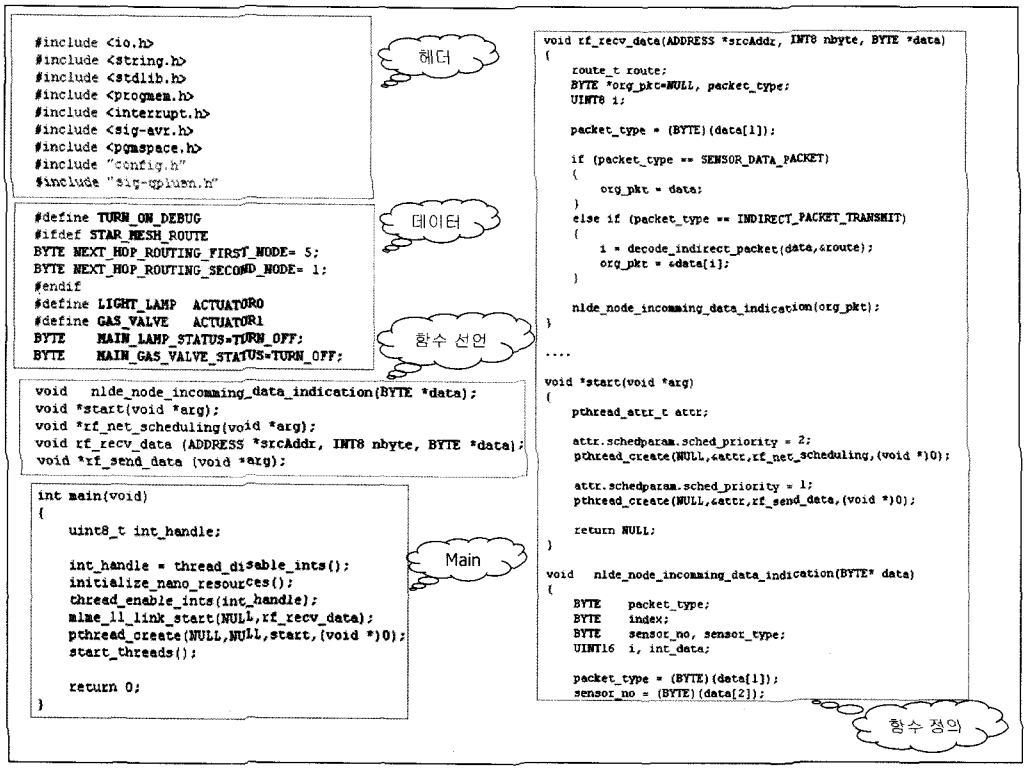



그림 10 가스 및 조도 모니터링 시스템을 구성하는 싱크 노드를 위한 응용 프로그램

반에서 사용되는 플러그인 형태로 개발된 도구이다.

TinyOS Plugin은 그래픽을 이용한 모델링을 지원하고, 프로그램의 오류를 표시해 주며, 코드를 자동으로 생성하지만, 모델에 대한 검증 기능을 제공하지 않는다. 또한 TinyOS Plugin을 사용하여 프로그램을 구축할 경우에는 nesC[9] 언어를 사용하여 코드를 작성해 주어야 하므로, nesC 언어를 배워야 하는 불편함이 있다. NanoEsto는 센서 네트워크 모델링 기능을 지원하지 않으며, 검증 기능을 제공하지 않는다. 또한 프로그램 코드의 템플릿은 자동으로 생성하지만, 실행 가능한 프로그램 코드를 생성해 주지는 않으므로, 직접 프로그램 코드를 작성해 주어야 한다. 게다가, TinyOS Plugin과 NanoEsto는 한 번에 하나의 노드에 대한 응용 프로그램만을 개발할 수 있다. TinyOS Plugin과 NanoEsto가 제공하는 모든 기능은 하나의 응용 프로그램만을 개발하는 것을 지원한다.

이에 비해 본 논문에서 제안하는 통합 개발 도구는 그래픽을 이용한 모델링 및 모델 검증 기능을 제공하고, 실행 가능한 프로그램을 자동으로 생성해주며, USN 모델에 나타나 있는 수많은 노드들에 대한 응용 프로그램을 한 번에 일괄적으로 생성할 수 있다. 또한 속성값 설정을 통하여 각 노드의 역할에 따른 응용의 각종 사항들을 설정하기만 하면 실행 가능한 코드가 생성이 되므로, 센서 네트워크에 대한 하위레벨의 정보를 자세히 알고 있지 못하고, 특정 언어에 대한 학습을 수행하지 않고도 쉽고 빠르게 USN 응용 프로그램을 생성할 수 있다.

이외에도 LabVIEW[10], RTDS[11]와 같은 임베디드 응용 프로그램을 모델로부터 쉽고 빠르게 생성하기 위한 도구들이 있으나, 이러한 도구들은 임베디드 운영체제를 지원하도록 개발되었기 때문에 센서 네트워크 응용 프로그램을 개발하기에는 적합하지 않다. 센서 네트워크 운영체제는 제한된 리소스를 사용하는 센서 노드를 제어할 수 있어야 하므로, 일반적인 임베디드 운영체제와는 다르게 저전력 오퍼레이션을 수행해야 하며, 그 크기 또한 초경량이어야 하는 등 여러 가지 면에서 다르다. 따라서 이러한 도구들을 사용하여 센서 네트워크 응용 프로그램을 개발하는 것은 쉽지 않다. 또한 앞서 언급한 TinyOS Plugin[7]이나 NanoEsto[8]와 마찬가지로 임베디드 응용 프로그램을 개발하기 위한 도구들 역시 한번에 하나의 응용 프로그램만을 생성하는 것을 지원하므로, 본 논문에서 제안하는 통합 개발 도구를 이용하여 개발하는 경우에 생산성이 더 높다.

## 5. 결 론

USN 응용 프로그램의 생성을 지원하는 기존의 도구들은 고품질의 응용 프로그램을 효율적으로 생성하는데 필요한 모델 검증 기능을 제공하지 않으며, 한번에 하나의 응용 프로그램에 대한 생성만을 지원한다. 또한 도구에서 지원하는 특정 언어를 학습해야 하는 불편함을 주기도 한다. 이러한 한계점을 보완하기 위하여, 본 논문에서는 센서 네트워크 모델로부터 센서 네트워크를 구성하는 다수의 노드를 위한 응용 프로그램을 쉽고 효율적으로 개발할 수 있도록 지원하는 모델 기반의 센서 네트워크 응용 프로그램 구축을 위한 통합 개발 도구를 제안한다. 본 논문에서 제안하는 통합 개발 도구는 다음과 같은 장점을 갖는다.

- 센서 네트워크 모델링 지원: 제안하는 통합 개발 도구는 UML의 표기법을 확장하여 그래픽을 이용하여 센서 네트워크를 모델링 할 수 있도록 지원한다.
- 속성값 설정을 통한 설계: 센서 네트워크 응용 프로그램을 생성하기 위해서 특정 언어를 학습하거나 복잡한 설계를 하지 않고, 각 노드의 역할에 따른 속성값을 설정하는 것만으로 응용 프로그램을 설계한다. 따라서 하위레벨의 정보를 자세히 알고 있지 못하고, 특정 언어에 대한 학습을 수행하지 않고도 쉽게 USN 응용 프로그램을 생성할 수 있다.
- 하나의 모델로부터 다수의 응용 프로그램을 생성: 하나의 센서 네트워크를 위해서는 많은 수의 센서 노드들이 필요하다. 따라서 센서 네트워크를 수행하기 위해서는 각 노드들마다 프로그램이 생성되어 배포되어야 한다. 제안하는 통합 개발 도구는 모델링 과정에서 작성한 센서 네트워크 모델에 포함되어 있는 모든 노드들에 대한 프로그램을 한 번에 생성할 수 있도록 지원한다. 따라서 개발자들은 프로그램 생성을 위한 시간과 노력을 절약할 수 있다.
- 모델의 검증: 모델링 과정에서 작성한 센서 네트워크 모델을 검증할 수 있는 기능을 제공한다. 센서 네트워크를 구성하는 각 센서 노드들은 각자의 역할에 따라 프로그램을 생성하기에 필요한 모듈 및 속성값이 다르다. 따라서 센서 네트워크를 구성하는 각 노드들이 자신에 역할에 맞도록 모델이 설계되었는가를 검증하는 것은 중요하다. 제안하는 통합 개발 도구는 모델 검증 기능을 통하여 모든 노드들이 공통적인 값을 가져야 하는 속성에 대하여 일치하도록 설계되었는가, 각 노드간의 연관 관계가 올바르게 설계되었는가, 각 노드의 역할에 따른 속성 설정이 올바르게 되었는가를 검사할 수 있도록 지원한다. 따라서 응용 프로그램의 오류 수정에 대한 비용을 절감할 수 있으며, 고품질의 응용 프로그램을 생성할 수 있다.

본 논문에서 제안하는 통합 개발 도구는 현재 나노큐플러스 운영체제를 기반으로 하는 센서 네트워크 응용 프로그램의 구축만을 지원하고 있다. 향후에는 나노큐플러스 뿐만이 아닌 여러 운영체제나 플랫폼을 지원하는 응용 프로그램을 구축할 수 있도록 발전시킬 것이다. 또한 모델의 검증뿐만이 아니라 시뮬레이션과 같은 기능을 추가하여 현재에는 커버하지 못하는 부분까지 검증할 수 있도록 하고자 한다.

## 참 고 문 헌


[1] Ian F. Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci, A survey on sensor networks, IEEE Communications Magazine August 2002, pp. 102-114, 2002.

[2] Shigeru Fukunaga, Tadamichi Tagawa, Kiyoshi Fukui, Koichi Tanimoto, and Hideaki Kanno, Development of ubiquitous sensor network, Oki Technical Review October 2004/Issue 200 Vol.71 No.4, pp. 24-29, 2004.

[3] 이우진, 김주일, 이광용, 정기원, "Nano-Qplus 기반의 USN 응용 프로그래밍 모델", 한국정보과학회 논문지 소프트웨어 및 응용, 제33권 제4호, pp. 378-387, 2006.

[4] Kwangyong Lee, Youngsam Shin, Heeseok Choi, and Seungmin Park, A design of sensor network system based on scalable & reconfigurable nano-OS platform, IT-SoC2004, 2004.

[5] Bill Moore, David Dean, Anna Gerber, Gunnar Wagenknecht, and Philippe Vanderheyden, Eclipse Development, International Business Machines Corporation, 2004.

[6] Eclipse Tools Project, "Graphical Modeling Framework," http://www.eclipse.org/gmf

[7] "TinyOS Plugin for Eclipse," http://www.dcg.ethz.ch/~rschuler/

[8] ETRI 임베디드 S/W 연구단, "나노Esto," http://qplus.or.kr/

[9] D. Gay, P. Levis, R. von Behren, M.Welsh, E. Brewer, and D. Culler, "The nesC language: A holistic approach to networked embedded systems," Proc. ACM SIGPLAN 2003 Conf. on Programming Language Design and Implementation (PLDI'03), ACM Press, pp. 1-11, 2003.

[10] "LabVIEW for Embedded Development," http://www.ni.com/pdf/products/us/2005-5554-821-101-LO.pdf

[11] http://www.pragmadev.com/index2.html

정 기 원

1967년 2월 서울대학교 전기공학과(공학사). 1981년 11월 미국 알라바마주립대(헌츠빌) 전산학과 석사. 1983년 12월 미국 텍사스주립대(알링턴) 전산학과 박사 1990년~현재 숭실대학교 컴퓨터학부 교수. 관심분야는 소프트웨어공학, 소프트웨어프로세스, 정보시스템감리, 전자거래(CALS/EC), 유비쿼터스 컴퓨팅

김 주 일

2004년 2월 한밭대학교 컴퓨터공학과(공학사). 2006년 2월 숭실대학교 대학원 컴퓨터학과(공학석사). 2006년 3월~현재 숭실대학교 대학원 컴퓨터학과 박사과정 관심분야는 유비쿼터스 컴퓨팅, 임베디드 시스템, 웹 서비스, 실시간 컴퓨팅, 소프트웨어공학

이 우 진

2000년 2월 숭실대학교 컴퓨터학부(공학사). 2002년 2월 숭실대학교 대학원 컴퓨터학과(공학석사). 2007년 2월 숭실대학교 대학원 컴퓨터학과(공학박사). 2006년 12월~현재 한국정보통신대학교 소프트웨어공학연구소 선임연구원(박사후과정생) 관심분야는 유비쿼터스 컴퓨팅, 임베디드 시스템, 서비스 지향 아키텍처, 소프트웨어공학