Nano-Qplus 기반의 USN 응용 프로그래밍 모델

(A Programming Model for USN Applications based on Nano-Qplus)

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

요 약 본 논문에서는 센서네트워크를 위한 운영체제인 Nano-Qplus 기반의 응용프로그램을 자동으로 생성하기 위한 프로그래밍 모델을 제시한다. 즉, 센서네트워크를 구성하는 센서, 라우터, 싱크, 엑츄에이터와 같은 노드들이 수행해야 하는 기능에 대한 프로그램을 자동으로 생성하기 위한 프로그래밍 모델을 제시한다. 제안한 프로그래밍 모델에 따라 센서네트워크에 대한 모델을 작성하고, 이를 바탕으로 센서네트워크의 각 노드에 대한 속성 값을 스크립트를 통하여 설정하면 각 노드를 동작시킬 수 있는 프로그램이 자동으로 생성된다. 본 논문에서는 프로그래밍 모델에서 사용하는 각 노드의 속성을 설정할 수 있는 스크립트와 프로그램을 자동으로 생성하는 알고리즘을 프로그래밍 모델과 함께 설명한다.

본 논문에서 제시한 기법을 이용하면 센서네트워크를 구성하는 각 노드에 대한 속성설정만으로 실행코드를 자동으로 생성함으로써 개발자들은 코드에 대한 상세한 내용을 알지 못하더라도 쉽게 응용프로그램을 구현할 수 있다. 또한 실행코드를 자동으로 생성함으로써 센서네트워크 응용프로그램을 개발하는데 소요되는 노력을 줄일 수 있으며, 신속한 코드생성을 통해 조기에 테스트를 수행하여 오류를 찾아내고 수정함으로써 검증된 코드를 생성할 수 있다.

키워드 : 센서네트워크, 프로그래밍 모델, 나노 큐플러스, 자동생성

Abstract A programming model for ubiquitous sensor network (USN) applications based on Nano-Qplus is proposed. USN applications mean programs of nodes which are components of sensor network such as sensor, router, sink and actuator. Developers can automatically generate programs of USN applications by setting attributes values of nodes using a script after they model a sensor network. A script for setting attributes values of a node is proposed in this paper. The algorithm of automatic code generation is also described. Developers can easily implement USN applications even if they do not know details of low-level communication, data sharing, and collective operations because the applications are automatically generated from a script. They set only attributes values of nodes using the script. Efforts for USN applications development also are reduced because of automatic code generation. Furthermore, developers can correct errors of applications in the early stage of development through early test based on rapid code generation.

Key words : USN, Programming Model, Nano-Qplus, Code Generation

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 주변환경을 감지하게 하여, 사용자는 네트워크를 통해서 언제, 어느 곳에서든 정보를 확인하고 활용할 수 있다. 센서네트워크는 생산, 유통, 물류 같은 경제 활동, 의료 서비스, 복

· 본 연구는 숭실대학교 교내연구비 지원으로 이루어졌음
† 학생회원 : 숭실대학교 대학원 컴퓨터학과
bluewj@empal.com
sespop@empal.com
†† 정 회 원 : 한국전자통신연구원(ETRI) 임베디드S/W연구단 편재형컴퓨팅미들웨어연구팀 연구원
kylee@etri.re.kr
††† 종신회원 : 숭실대학교 컴퓨터학부 교수
chong@ssu.ac.kr
논문접수 : 2005년 11월 4일
심사완료 : 2006년 2월 22일

지 서비스, 환경 감시와 같은 서비스를 위한 응용프로그램을 개발하는데 사용되고 있다.

그러나 이러한 센서네트워크 응용프로그램을 작성하는 것은 어려운 작업이다. 센서네트워크를 구성하는 노드들의 자원은 한정되어 있으며, 노드간의 무선통신은 신뢰하기 어려우며, 노드들은 저전력 오퍼레이션을 수행해야 하므로, 이러한 것들을 고려하여 응용프로그램을 작성하는 것은 매우 어려운 작업이다. 따라서, 개발자들이 하위레벨까지 알지 못하더라도 센서네트워크 응용프로그램을 쉽게 작성하도록 하는 방법이 필요하다. 즉, 하위레벨의 의사소통, 데이타 공유, 오퍼레이션에 대한 자세한 내용을 추상화하여 간단하게 응용프로그램을 설계할 수 있도록 할 필요가 있다.

이를 위하여, 본 논문에서는 센서네트워크 응용프로그램의 코드를 간단한 설계를 통하여 자동으로 생성하는 기법을 제시한다. 개발자는 본 논문에서 제시하는 스크립트를 통하여 센서네트워크를 구성하는 센서, 라우터, 싱크, 액츄에이터와 같은 노드들이 수행해야 하는 기능에 대한 값을 설정하면, 각 노드에 해당하는 프로그램이 자동으로 생성된다. 따라서 개발자들은 하위레벨까지 알 필요 없이 쉽게 센서네트워크 응용프로그램을 작성할 수 있다. 본 논문에서 제시하는 기법은 특히, 센서네트워크를 위한 운영체제인 Nano-Qplus[1,2]를 기반으로 수행되는 센서네트워크 응용프로그램에 초점을 둔다.

2. 관련연구

Cheong[3]이 제시한 TinyGALS는 이벤트 기반의 임베디드 시스템을 프로그래밍하기 위한 모델로서, 상위 수준의 명세로부터 비동기적인 메시지 전달 코드와 모듈 트리거링 메커니즘을 자동으로 생성할 수 있다. TinyGALS는 특히, 무선 센서네트워크 플랫폼인 버클리의 모트에서 실행되는 코드를 자동으로 생성하는 프로그래밍 모델이다. Welsh[4]는 추상화 지역이라는 것을 제시하여 하위 레벨의 통신, 데이타 공유, 오퍼레이션 등의 세부사항을 추상화하는 프로그래밍 원형을 제공함으로써 센서네트워크 응용프로그램의 설계를 간단하게 하는 기법을 제시하였다. Newton[5]은 Regiment라는 센서네트워크를 위한 프로그래밍 언어를 제안하여, 단지 몇 라인의 코드로 복잡한 센서네트워크 응용프로그램을 작성할 수 있도록 하였다. Boulis[6]는 응용 특화 방식에서 센서 노드에서의 계산, 의사소통, 감지 자원을 효율적으로 이용하도록 하는 경량의 모바일 제어 스크립트를 정의하고 지원하는 프레임워크인 SensorWare를 제시하였다. SensorWare 아키텍처는 스크립팅이 가능한 경량의 런타임 환경에 기반하고, 한정된 에너지와 메모리를 가진 센서 노드를 위하여 최적화되어 있다. Greenstein[7]은 효율적인 센서네트워크 응용프로그램을 쉽게 개발할 수 있도록 하는 구성언어, 컴포넌트, 서비스 라이브러리 및 컴파일러를 포함하는 SNACK (Sensor Network Application Construction Kit)을 제안하였다. SNACK을 이용하여 라우팅 트리, 주기적 감지와 같은 개념들을 구현하고, 이러한 것들을 바탕으로 효율적인 프로그램을 자동으로 생성할 수 있다. Ramakrishna Gummadi[8]는 프로그래머가 자세한 분산코드 생성이나 인스턴스화, 원격 데이타 접근과 관리, 노드 내의 프로그램 흐름 조정과 같은 것들을 자세히 알지 못하더라고 센서네트워크 응용프로그램을 구현할 수 있도록 하는 프로그래밍 모델을 제안하였다. Cheong[3]의 기법에서는 센서네트워크 응용프로그램을 작성하기 위해서 상위 수준의 스펙을 작성해 주어야 한다. Welsh [4]의 기법에서는 제공하는 API를 사용하여 센서네트워크 응용프로그램을 작성해야 하며, Newton[5], Greenstein [7], Greenstein Ramakrishna Gummadi[8]의 기법에서는 제시한 언어를 사용하여 센서네트워크 응용프로그램을 작성해야 한다. Boulis[6]의 기법을 이용하여 센서네트워크 응용프로그램을 작성하기 위해서는 제시한 스크립트 코드를 작성해야 한다. 이에 반해, 본 논문에서 제시하는 기법은 스크립트에서 값을 선택해 주기만 하면 자동으로 코드를 생성하므로, 더욱 간단하게 코드를 생성할 수 있다. 개발자들이 본 논문의 기법을 사용하여 센서네트워크 응용프로그램을 작성한다면, 새로운 언어를 배우거나, 상위 레벨의 명세를 작성할 필요가 없다. 단지 모델링을 통해 각 센서들의 역할을 정의하고, 역할에 따른 속성값을 스크립트를 통하여 설정해 주기만 하면 각 센서들의 역할 수행을 위한 프로그램이 자동으로 생성되는 것이다. 따라서 다른 기법들에 비하여 간단하면서도 응용프로그램의 생산성은 높다. 그러나 본 논문에서 제시하는 기법은 아직까지는 센서네트워크 운영체제인 Nano-Qplus에서 수행되는 응용프로그램만을 자동으로 생성할 수 있다.

3. Nano-Qplus 기반의 USN 응용 프로그래밍 모델

이 장에서는 Nano-Qplus 기반의 USN(Ubiquitous Sensor Network) 응용프로그램을 자동으로 생성하기 위하여 본 논문에서 제시하는 프로그래밍 모델을 설명한다. USN 응용프로그램을 생성하는 기법을 기존의 프로그래밍 모델들과 비교하고, 응용프로그램 자동생성을 위한 핵심이 되는 스크립트를 설명한다. 또한 스크립트

의 정보를 바탕으로 응용프로그램을 자동으로 생성하는 알고리즘을 제시한다.

3.1 USN 응용프로그램 생성 방법

그림 1은 기존에 수행한 연구들에서 제시한 USN 응용프로그램을 생성하는 방법을 정리한 것이다. 기존의 연구에서는 USN 응용프로그램을 생성하기 위하여 모델을 작성하고, 작성한 모델을 바탕으로 상위 수준의 언어를 이용하여 간단하게 프로그램을 작성하면, 이 프로그램을 기반으로 코드를 자동으로 생성한다. 여기에서 핵심이 되는 부분이 상위 수준의 언어를 이용하여 프로그램을 작성하는 것이다. 이것은 사용자가 상세한 내용을 알지 못하더라도 쉽게 응용프로그램을 생성할 수 있게 해준다. 따라서 기존의 연구에서는 사용자가 쉽게 USN 응용프로그램을 작성할 수 있는 명세 수준의 언어, 스크립트 언어나 API 등을 제시하였다. 그러나 사용자들은 이러한 방법들로 응용프로그램을 개발하기 위해서 제시하는 언어, 스크립트 언어나 API 등을 습득해야만 한다는 단점이 있다.

본 논문에서는 이를 보완하기 위하여 그림 2와 같은 방법을 제시한다. 모델을 작성하고, 작성한 모델을 바탕으로 제시하는 스크립트를 통하여 값을 설정해주기만 하면, 자동으로 응용프로그램 코드를 생성하는 것이다. 따라서 사용자는 제시하는 언어나 API 등을 습득할 필요 없이 모델을 바탕으로 스크립트의 각 항목에 대한 값을 설정하기만 하면 자동으로 응용프로그램을 생성할 수 있는 것이다.

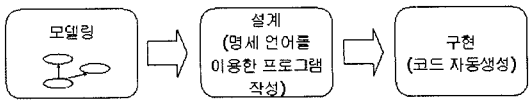

그림 1 기존의 USN 응용프로그램 생성 방법

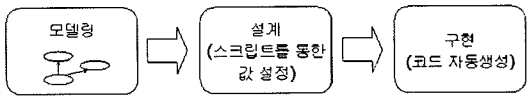

그림 2 본 논문의 USN 응용프로그램 생성 방법

3.2 Nano-Qplus 기반의 USN 응용프로그램 생성 모델

3.1에서 제시한 바와 같이 본 논문에서 제시하는 기법은 USN에 대한 모델을 작성하여 이를 바탕으로 스크립트를 통하여 값을 설정한 후에 코드를 자동으로 생성한다. 따라서 본 논문에서 제시하는 기법을 통하여 USN 응용프로그램을 생성하기 위해서는 그림 3과 같은 USN 모델을 작성하여야 한다. 본 논문에서 제시하는 USN 응용프로그램 자동생성 기법은 모델링 단계에서 작성한 모델에 나타나 있는 속성값 및 데이타 통신의 경로를 바탕으로 프로그램을 생성하는 것이므로, 모델이 올바르게 작성되지 않으면 의도한대로 수행되는 프로그램을 생성할 수 없다. 따라서 모델에는 USN을 구성하는 모든 유형의 노드와 각 노드간의 데이타 통신의 경로 및 각 노드의 역할 수행을 위한 속성값들이 나타나 있어야 한다. 이렇게 작성한 모델을 바탕으로 USN을 구성하는 각 노드인 센서, 라우터, 싱크, 엑츄에이터에 대한 스크

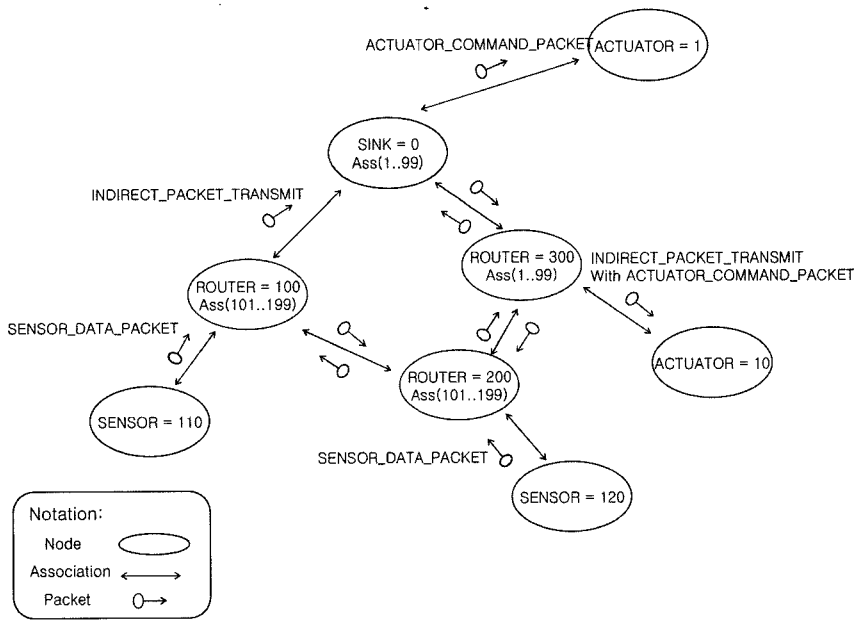

그림 3 USN 모델의 예

립트 값을 설정하고, 설정한 값에 따라 Nano-Qplus에서 제공하는 모듈을 포함하여 각 노드를 위한 프로그램을 생성할 수 있다.

USN을 구성하는 각 노드들은 기본적으로 Store& Forward 방식을 통하여 서로 통신한다. 따라서 본 논문에서는 이러한 방식을 바탕으로 각 노드를 위한 프로그램을 생성하기 위한 템플릿을 제공한다.

그림 4-7은 USN을 구성하는 센서, 라우터, 싱크, 엑츄에이터 노드를 위한 프로그램을 생성하기 위하여 Nano-Qplus에서 제공하는 모듈을 포함하고 있는 템플릿을 보여준다. 네 종류의 노드들은 기본적으로 전부 같은 모듈들을 가지고 있지만, 모듈의 내용은 노드의 유형에 따라 달라진다. 센서 노드는 감지한 데이타를 다른 노드로 전달하는 것이 주요 역할이므로 4가지 모듈 중에서 데이타를 보내기 위한 rf_send_data() 모듈이 스크립트의 설정에 따라 달라지며, 라우터 노드는 다른 노드로부터 전달받은 데이타를 전송하는 것이 주요 역할이므로 rf_net_scheduling() 부분의 내용이 달라진다. 싱크와 엑츄에이터 노드는 전달받은 데이타를 분석하여 처리하는 것이 주요 역할이므로 rf_recv_data() 모듈의 내용이 설정값에 따라 달라지게 된다. 이와 같이, 본 논문에서는 USN 응용프로그램을 자동으로 생성하기 위하여 USN을 구성하는 네 가지 유형의 노드에 대한 템플릿을 제공한다. 이러한 템플릿을 바탕으로 사용자의 설정값에 따라 세부적인 프로그램을 자동으로 생성함으로써, 사용자들은 USN을 구성하는 노드들이 기본적으로 수행하는 오퍼레이션을 이해하고, 각 노드의 유형에 맞는 속성값을 올바르게 설정해 주기만 하면 오퍼레이션의 상세한 코드를 알지 못하더라도 쉽게 프로그램을 작성할 수 있는 것이다.

3.3 USN 응용프로그램 생성을 위한 스크립트

본 논문에서는 설계한 모델을 바탕으로 USN 응용프로그램을 생성하기 위하여 USN을 구성하는 각 노드들의 속성값을 쉽게 설정할 수 있도록 그림 8과 같은 스크립트를 제공한다.

이 스크립트는 리눅스에서 환경설정을 위해 사용하는 스크립트[9]를 참고하여 개발한 것이므로 사용자들은 친

```
[ ] Enable EEPROM module (NEW)
[ ]  nable Flash memory module (NEW)
[*]  nable Timer module
[ ]    nable Digital clock module (NEW)
[*]  nable UART module
[*]    nable printf module (NEW)
[ ]    nable scanf module (NEW)
[*]  nable actuation(e.g.,Relay,LED,...) modulec
[*]    nable LED module (NEW)
[ ]  nable ADC or Sensor module (NEW)
[*]  cheduler module
[ ]    IFO scheduler (NEW)
[*]    reemption-RR scheduler (NEW)
[ ]      ower management (NEW)
[*]  nable Zigbee RF module
[*]    nable Simple Send/Recv module (NEW)
[*]    nable IEEE 802.15.4 MAC module
[*]      nable Star-Mesh route module
(SINK)  N de type?(SINK,ROUTER,...) (NEW)
(0)      efault Application Node ID?
(1)      djacent Acutator Node ID?
(TRUE)   s it PAN Cordinator node?(TRUE/FALSE) (NEW)
(NON_BEACON_ENABLE) N N_BEACON_ENABLE or not? (NEW)
(0x1111)  EFAULT_SRC_SHORT_MAC_ADDR?
[ ]      s the DEFAULT_EXTENDED_MAC_ADDR used? (NEW)
(1)      SSOCIATION_PERMIT_NODEID_START?
(99)     TART_MESH_ASSOCIATION_PERMIT_NODEID_END?
[ ]    nable RSSI module (NEW)
[ ] Enable Inter-Thread-Communication (NEW)
```

그림 8 USN 응용프로그램 자동생성을 위한 스크립트

숙하게 다룰 수 있을 것이다. 사용자는 각 노드에 대한 속성값을 스크립트를 통해서 설정해 주면, 스크립트의 내용을 바탕으로 응용프로그램이 자동으로 생성된다. 이 스크립트에서 제공하는 속성들은 3.2절에서 제시한 템플릿을 바탕으로 각 노드가 수행해야 하는 프로그램을 생성하기 위한 것이다. 앞서 이야기한 것과 같이, 각 유형의 노드가 가지는 오퍼레이션은 기본적으로 같지만, USN 상에서 각 노드마다 수행하는 행동은 다르다. 따라서, 이를 반영하기 위하여 그림 8과 같은 스크립트를 제시하는 것이다. 각 노드가 오퍼레이션에서 수행하는 상세한 행동은 제시하는 스크립트를 통하여 설정한 값을 바탕으로 생성되므로, 모델링 단계에서 작성한 모델을 반영하여 스크립트의 값을 설정해야 한다. 스크립트의 값이 잘못 설정된다면, 최종적으로 생성된 응용프로그램은 사용자가 원하던 기능을 올바르게 수행할 수 없게 된다. 따라서, 응용프로그램의 수행결과를 바탕으로

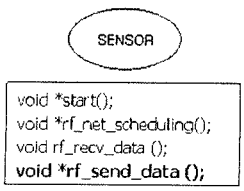

그림 4 센서 노드를 위한 모듈

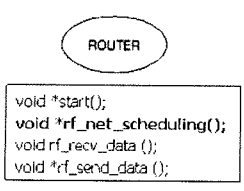

그림 5 라우터 노드를 위한 모듈

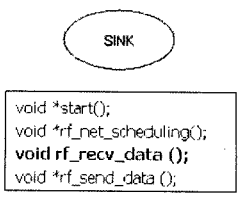

그림 6 싱크 노드를 위한 모듈

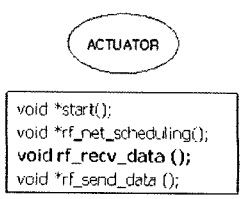

그림 7 엑츄에이터 노드를 위한 모듈

스크립트의 설정결과가 올바르지 않다는 것을 알게 되면, 모델링이 잘못 되었음을 알게 되므로, 이를 통하여 모델을 검증할 수 있다. 또한 모델링이 잘못 되었을 경우에는, 스크립트의 속성값의 변경을 통하여 응용프로그램을 생성하고 테스트해 봄으로써 작성한 모델의 문제점을 쉽게 발견하고 수정할 수 있다.

3.4 USN 응용프로그램 생성 알고리즘

스크립트를 통하여 선택한 정보를 바탕으로 각 노드의 실행을 위한 소스코드는 다음과 같은 절차를 통하여 생성된다.

Step 1. 유저가 선택한 정보를 저장한 Config_Info (.config) 파일을 입력 받는다.

Step 2. Config_Info 파일을 Parsing하여 선택된 모듈을 찾아내고 해당 모듈의 header, data, function Code를 HashTable_Module<Type>에서 읽어와 Templet에 저장한다.

Step 3. 선택된 모듈에 따른 Main Code를 HashTable_Main에서 읽어와 Templet에 저장한다.

사용자가 3.3절에서 제시한 스크립트를 통하여 각 노드에 대한 속성값을 설정하면, 설정한 정보를 저장하고 있는 .config 파일이 생성된다. 3.2절에서 제시한 노드 유형에 따른 템플릿의 오퍼레이션의 세부 내용을 생성된 .config 파일의 값을 바탕으로 작성하고, 오퍼레이션을 수행하기 위하여 필요한 헤더와 메인 코드를 추가함으로써 각 노드가 수행할 프로그램을 생성할 수 있다. 이를 위하여 스크립트의 속성값 설정으로 생성된 .config 파일을 해석하여 각 노드의 역할에 맞는 코드를 템플릿으로부터 생성하기 위한 알고리즘이 필요하다. 따라서 본 논문에서는 각 노드의 역할을 수행하기 위한 프로그램을 자동으로 생성하기 위하여, 그림 9와 같은 알고리즘을 제시한다. 이 알고리즘에서는 .config 파일을 해석한 결과와 HashTable_Module 이라는 클래스를 사용하여 각 노드에서 필요한 프로그램을 위한 헤더, 데이타, 함수의 코드를 생성한다. 또한 이와 유사한 방법으로 HashTable_Main 클래스를 사용하여 각 노드에서 필요한 메인 코드를 생성한다.

```
Templet Transformation(Config_Info config_info) {
    templet = getTemplet(config_info.NODETYPE);
    templet.setAttribute(config_info.Attribute);

    HashTable_Module  hashTable_Module = new HashTable_Module();
    HashTable_Main    hashTable_Main = new HashTable_Main();

//  HashTable_Module<NODETYPE> 설정
    hashTable_Module.setType(config_info.NODETYPE);
// 각 type에 따라 config_info에 맞도록 코드를 구성
    Iterator iterator = Parser.getIterator(conifg_info);
    while( iterator.hasNext() ) {
        templet.addHeader(
            hashTable_Module.getHeader( iterator.ModuleName ) );
        templet.addData(
            hashTable_Module.getData( iterator.ModuleName ) );
        templet.addFunction(
            hashTable_Module.getFunction( iterator.ModuleName ) ) ;
        templet.addMain(
            hashTable_Main.getMain( iterator.ModuleName ) );
        iterator.next();
    }
}
```

그림 9 USN 응용프로그램 생성을 위한 알고리즘

HashTable_Module 클래스는 노드의 각 유형에 따라 프로그램 코드를 읽어올 수 있도록 구성되어 있다.

HashTable_Module 클래스에는 각 노드에서 실행되는 프로그램에 필요한 헤더, 데이타, 함수의 코드를 읽어오기 위한 getHeader(), getData(), getFunction() 함수가 정의되어 있다. 이 함수들은 getReplaceModule() 함수를 통하여 해쉬 테이블에 정의되어 있는 키와 값을 이용하여 해당 노드를 위한 소스코드를 가져온다. HashTable_Module 클래스의 구조는 그림 10과 같다.

USN을 구성하는 각 노드의 프로그램을 자동으로 생성하기 위하여 노드의 유형 및 스크립트의 속성에 따른 헤더, 데이타, 함수의 코드는 미리 개발되어 저장되어 있다. 따라서 프로그램을 생성하는 알고리즘은 각 노드의 템플릿을 바탕으로 속성값에 따라 필요한 헤더, 데이타, 함수를 저장되어 있는 파일로부터 읽어와 프로그램을 생성하는 것이다. 이렇게 노드의 유형 및 스크립트의 속성에 따른 모듈이 저장되어 있는 파일의 이름은 해쉬 테이블에 저장되어 있다. HashTable_Module 클래스는 노드의 유형에 따라 헤더, 데이타, 함수의 코드가 다르기 때문에 동적으로 해쉬테이블을 사용하여 각 유형에 따른 코드를 생성하여 주도록 구성하였다. 정적으로 값이 고정되어 있는 해쉬테이블을 사용하면 각 노드 유형마다 해쉬테이블을 따로 생성해야 하며, 각 해쉬테이블을 제어하기 위한 코드 또한 각각 만들어 줘야 한다. 또한 노드의 유형이 추가될 경우에는 새로운 해쉬테이블을 생성해 주어야 하며, 그에 대한 코드도 새로 추가해 주어야 한다. 그러나 동적으로 해쉬테이블의 값을 바꿀 수 있도록 하면, 새로운 노드의 유형이 추가되더라도 해쉬테이블에 키와 그 값만 넣어주면 되므로, 새로운 코드를 추가하지 않고, 그대로 사용할 수 있다.

HashTable_Module 클래스에서 사용하는 해쉬테이블은 표 1과 같은 구조를 갖는다. 해쉬테이블의 구조에서

표 1 해쉬테이블의 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"

```
public class HashTable_Module{
        private Hashtable moduleTable;
        private String[] moduleKey = {"Zigbee_Simple", "Zigbee_MAC","Zigbee_MAC_StarMesh", "Scheduler_FIFO", "Scheduler_PreemptionRR",
                "Sensor_LIGHT","Sensor_GAS","Sensor_Temperature"};
        String nodeType = "";
        private final String HEADER = "H";
        private final String DATA = "D";
        private final String FUNCTION = "F";
        public void setType(String nodeType) {
                TempletHashTable templetTable = new TempletHashTable();
                moduleTable = new Hashtable();
                int size = moduleKey.length;
                String templetModule = "";
                this.nodeType = nodeType;
                for(int i = 0; i < size; i++) {
                        templetModule = (String)templetTable.get(moduleKey[i]);
                        templetModule = templetModule.replaceFirst("&1", nodeType);
                        moduleTable.put(moduleKey[i], templetModule);
                }
        }
        public String getHeader(String moduleName) {
                return getReplaceModule(moduleName, HEADER);
        }
        public String getData(String moduleName) {
                return getReplaceModule(moduleName, DATA);
        }
        public String getFunction(String moduleName) {
                return getReplaceModule(moduleName, FUNCTION);
        }
        public String getReplaceModule(String moduleName, String value) {
                String templetModule = (String)moduleTable.get(moduleName);
                templetModule = templetModule.replaceFirst("&2", value);
                return templetModule;
        }
};
```

그림 10 HashTable_Module 클래스

키의 값은 USN을 위한 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. 이것을 예로 들면 다음과 같다.

예. 노드에서 rf 통신을 위한 모듈로 Zigbee_Simple가 선택되었다고 가정하면 해쉬테이블에서 "&1_Zig_Simple_&2" 값을 가져 올 것이다. 그러면 Hash Table_Module 클래스의 각 함수를 호출함에 따라 해쉬테이블의 값이 다음과 같이 변경될 것이다.

setType("SINK"); -→ "SINK_Zig_Simple_&2"

getHeader("Zig_Simple") -→ "SINK_Zig_Simple_H"

getFunction("Zig_Simple") -→ "SINK_Zig_Simple_F"

여기에서 "SINK_Zig_Simple_H"는 싱크 노드에서 ZigBee_Simple 모듈을 사용하고자 할 경우에 필요한 헤더코드를 담고 있는 파일의 이름이며, "SINK_Zig_Simple_F"는 함수의 코드를 담고 있는 파일의 이름이다.

이와 마찬가지로 각 노드의 메인 함수를 생성하기 위하여 HashTable_Main 클래스에서 사용하는 메인 함수에 대한 해쉬테이블이 따로 정의되어 있다.

4. 사례연구

본 논문에서는 제시한 기법을 이용하여 그림 11과 같은 가스안전모니터링 시스템에서 사용하는 USN 응용프로그램을 자동 생성하여 적용시켜 보았다.

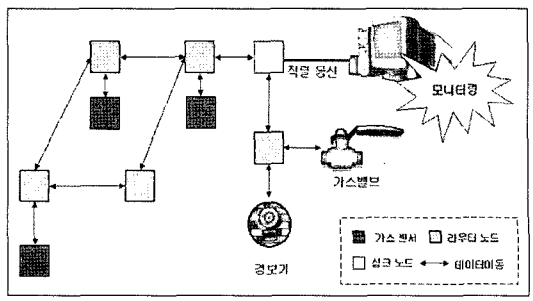

그림 11 가스안전모니터링시스템의 구조도

가스안전모니터링 시스템은 가스를 감지하기 위한 가스 센서, 센서로부터 감지된 값을 전달 받아 싱크 노드로 전송하기 위한 라우터 노드, 모니터링 시스템과 연결되어 있으며, 감지된 값에 따라 엑츄에이터의 행동을 결정하는 싱크노드, 싱크노드로부터 전달 받은 명령에 따라 경보를 울리거나 가스를 차단하는 행동을 수행하는 엑츄에이터 노드가 포함된 경보기와 가스밸브로 구성되어 있다.

이러한 구조를 가지는 시스템의 응용프로그램을 개발하기 위하여 센서 노드로부터 감지된 가스 값이 라우터를 거쳐, 싱크 노드로 전달이 되고, 한계(threshold) 값에 따라 엑츄에이터 노드가 작동되는지를 점검할 수 있도록 모델을 설계하였다. 설계한 모델을 바탕으로 스크립트를 이용하여 응용프로그램을 자동 생성한 후 그림 11의 시스템에 적용하였다. 개발한 응용프로그램이 적절하게 작용하는가를 확인하기 위하여 가스를 노출시켜 센서가 감지하도록 하였다. 가스 센서는 감지된 가스에 대한 값을 라우터를 통하여 싱크 노드로 전달하였고, 설계 단계에서 지정한 가스누출의 한계 값을 넘지 않는 데이타에 대해서는 경보기와 가스밸브가 작동하지 않았다. 그러나 한계 값을 넘는 데이타가 싱크 노드로 전달되자 싱크 노드는 경보기와 가스밸브가 수행해야 하는 행동명령을 라우터 노드를 통하여 엑츄에이터 노드에 전달하였고, 이에 따라 경보기가 울리고 가스밸브가 잠기는 것을 확인할 수 있었다. 가스 센서를 통하여 감지되는 가스의 값과 각 노드로 전달되는 값은 싱크 노드에 연결되어 있는 모니터링 시스템을 통하여 확인할 수 있었다. 그림 12(a)는 가스가 누출되었을 경우 센서가 이를 감지하고, 감지된 데이타가 라우터 노드를 통하여 싱크 노드로 전달되는 결과를 보여주는 모니터링 화면이고, 그림 12(b)는 가스가 누출되었다는 데이타를 전달받고 가스밸브를 잠그기 위하여 싱크 노드가 가스벨브의 엑츄에이터 노드로 명령을 전달하는 것을 보여주는 화면이다.

그림 13은 싱크 노드에 대한 응용프로그램을 자동으로 생성하기 위하여 스크립트의 속성값을 설정하는 예를 보여준다. 이와 같은 방법으로 센서네트워크를 구성하는 각 노드에 대하여 스크립트의 속성 값을 설정해 주면 응용프로그램을 자동으로 생성할 수 있다.

그림 14는 그림 13의 스크립트를 통하여 설정한 값을 바탕으로 생성한 config 파일의 일부를 보여준다. 각 노드의 응용프로그램은 이 config 파일의 값을 바탕으로 자동으로 생성된다.

그림 15는 그림 14의 config 파일을 바탕으로 자동으로 생성한 싱크 노드의 소스코드이다.

본 논문에서 제시한 기법에 따라 센서네트워크 응용프로그램을 개발한 결과, 생산성이 크게 향상되었음을 알 수 있었다. 개발하고자 하는 센서네트워크에 대한 모델을 작성하고 나면, 모델을 바탕으로 각 노드의 프로그램을 생성하기 위한 속성설정은 각 노드당 스크립트를 통하여 1,2분 이내에 작성할 수 있었으며, 작성된 스크립트를 바탕으로 코드를 자동으로 생성하는 것은 30초 이내로 수행되었다. 센서네트워크 응용프로그램을 개발하기 위한 대부분의 기법들은 3.1절에서 설명한대로 모델링을 수행한 후에 작성한 모델을 바탕으로 제시하는 기법들을 사용하여 프로그램을 생성한다. 따라서 다른 기법들과 본 논문에서 제시하는 기법의 차이점은 모델링 이후의 작업에서 나타난다. 본 논문에서 제시하는 기

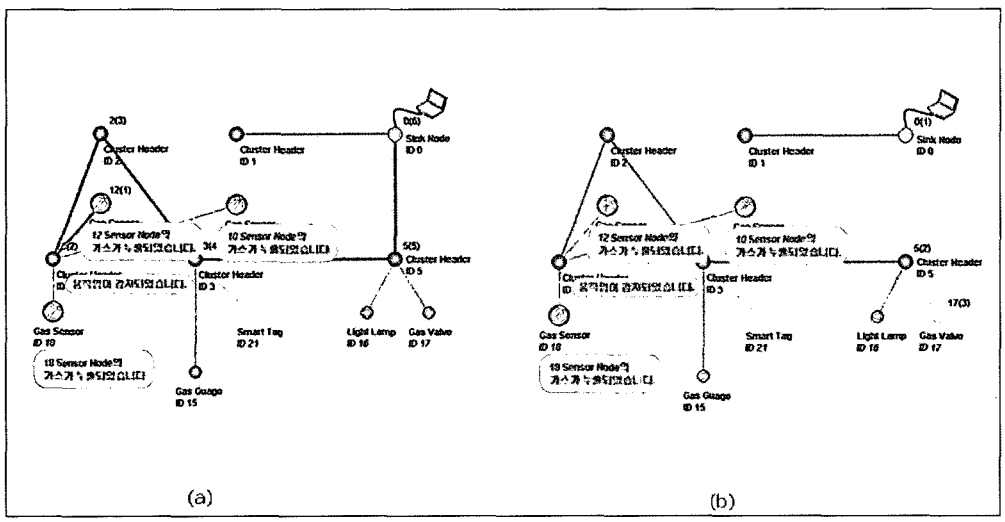

그림 12 가스안전모니터링시스템의 실험 결과

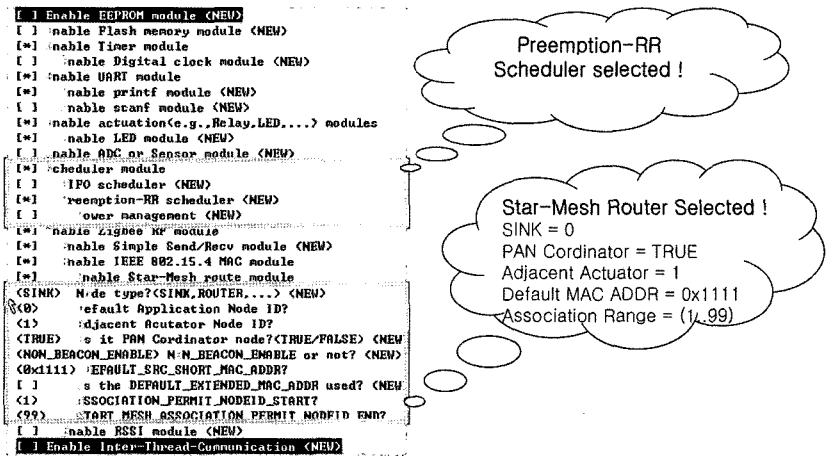

그림 13 싱크 노드의 스크립트 속성 설정 예

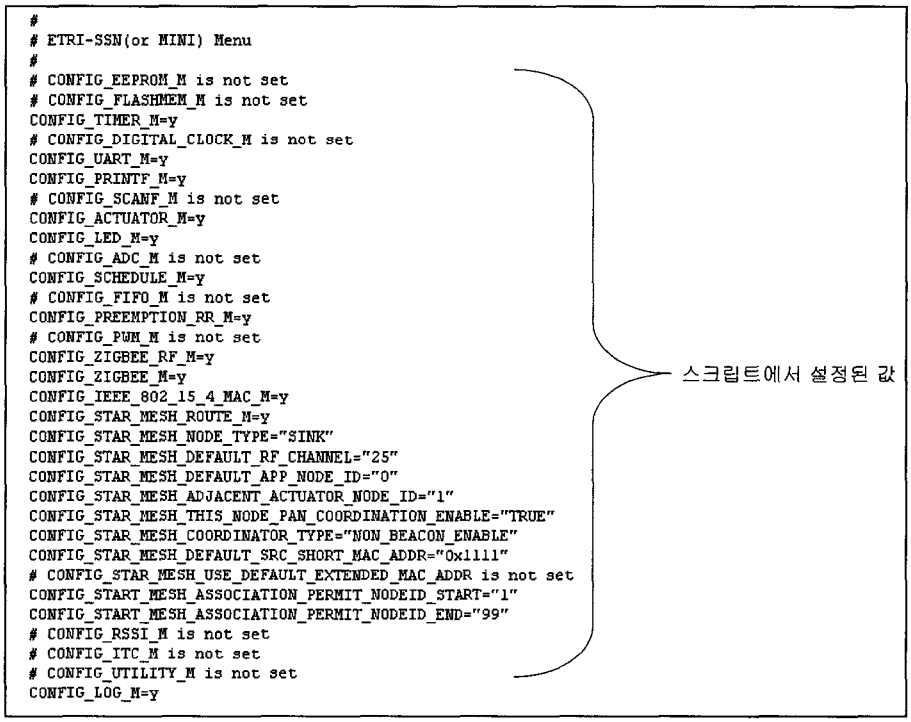

그림 14 싱크 노드의 config 파일

법을 이용하여 센서네트워크 응용프로그램을 개발한다면, 앞서 설명한대로 각 노드에서 수행해야 하는 프로그램을 생성하는 시간이 매우 크게 단축된다. 따라서 센서네트워크 응용프로그램의 개발 생산성이 다른 기법들에 비하여 더욱 크게 향상된다. 또한 본 논문의 기법은 개발자가 이용방법을 쉽게 습득할 수 있으므로, 초기의 훈련시간이 다른 기법들에 비하여 단축되는 것을 알 수 있었다.

5. 결론 및 향후 연구

본 논문에서는 센서네트워크를 지원하기 위한 운영체제인 Nano-Qplus를 기반으로 수행되는 센서네트워크 응용프로그램을 생성하기 위한 프로그래밍 모델을 제시하였다. 센서네트워크 프로그램은 모델을 작성하고, 작성한 모델을 바탕으로 USN을 구성하는 각 노드에 대한 속성 값을 스크립트를 통하여 설정하면, 스크립트 정보

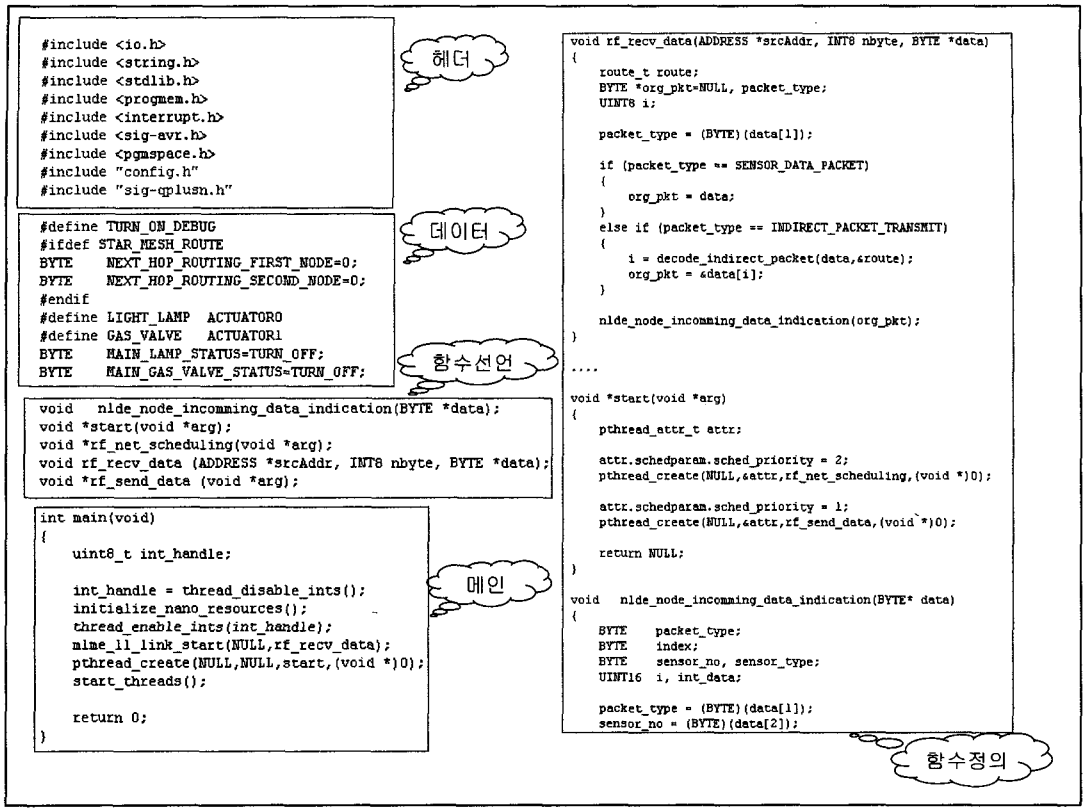

그림 15 자동 생성된 싱크 노드의 소스코드

를 기반으로 실행코드를 자동생성으로 생성함으로써 구현할 수 있다. 본 논문에서 제시한 기법을 이용하면 센서네트워크를 구성하는 각 노드에 대한 속성설정만으로 실행코드를 자동으로 생성함으로써 개발자들은 코드에 대한 상세한 내용을 알지 못하더라도 쉽게 응용프로그램을 구현할 수 있다. 또한 실행코드를 자동으로 생성함으로써 센서네트워크 응용프로그램을 개발하는데 소요되는 노력을 줄일 수 있으며, 신속한 코드생성을 통해 조기에 테스트를 수행하여 오류를 찾아내고 수정함으로써 검증된 코드를 생성할 수 있다.

향후 연구로는 센서네트워크 응용프로그램에 대한 모델 작성을 바탕으로 코드생성까지 자동화하는 도구를 개발하고자 한다.

참 고 문 헌

[1] Kwangyong Lee et al., "A Design of Sensor Network System based on Scalable & Reconfigurable Nano-OS Platform," IT-SoC2004, October 2004.

[2] ETRI 임베디드 S/W 연구단, "나노 Qplus," http://qplus.or.kr/

[3] E. Cheong, J. Liebman, J. Liu, and F. Zhao, "Tinygals: a programming model for eventdriven embedded systems," SAC, 2003.

[4] M. Welsh and G. Mainland, "Programming sensor networks using abstract regions," NSDI, 2004.

[5] R. Newton and M. Welsh, "Region streams: Functional macroprogramming for sensor networks," DMSN, 2004.

[6] A. Boulis, C. Han, and M. B. Srivastava, "Design and implementation of a framework for efficient and programmable sensor networks," MobiSys, 2003.

[7] B. Greenstein, E. Kohler, and D. Estrin, "A sensor network application construction kit (SNACK)," SenSys, 2004.

[8] Ramakrishna Gummadi, Omprakash Gnawali, and Ramesh Govindan, "Macro-programming Wireless Sensor Networks Using Kairos," LNCS 3560, pp. 126-140, 2005.

[9] Neil Matthew , Richard Stones, "Beginning Linux Programming 3rd Edition," WROX PRESS, 2003.

이 우 진

2000년 2월 숭실대학교 컴퓨터학부(공학사). 2002년 2월 숭실대학교 대학원 컴퓨터학과(공학석사). 2002년 3월~현재 숭실대학교 대학원 컴퓨터학과 박사과정 관심분야는 유비쿼터스 컴퓨팅, 임베디드 시스템, 웹 서비스, 모바일 컴퓨팅, 소프트웨어공학

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 광 용

1991년 2월 숭실대학교 전자계산학과(학사). 1993년 2월 숭실대학교대학원 전자계산학과(공학석사). 1997년 2월 숭실대학교대학원 전자계산학과(공학박사). 1997년~1998년 ETRI 컴퓨터·소프트웨어기술연구소 소프트웨어공학연구부 박사후 연수연구원. 1999년~현재 ETRI 임베디드S/W연구단 편재형컴퓨팅미들웨어연구팀 선임연구원. 2005년~현재 과학기술연합대학원대학교 겸임교수. 관심분야는 유비쿼터스 컴퓨팅, 임베디드 시스템, 나노 운영체제, 소프트웨어공학

정 기 원

1967년 2월 서울대학교 전기공학과(공학사). 1981년 11월 미국 알라바마주립대(헌츠빌) 전산학과 석사. 1983년 12월 미국 텍사스주립대(알링턴) 전산학과 박사 1971년~1975년 한국과학기술연구소 연구원. 1975년~1990년 국방과학연구소 책임연구원. 1990년~현재 숭실대학교 컴퓨터학부 교수 2001년~현재 IT감리포럼 회장. 관심분야는 소프트웨어공학, 소프트웨어프로세스, 정보시스템감리, 전자거래(CALS/EC), 유비쿼터스 컴퓨팅