Journal of the Korea Academia-Industrial cooperation Society
Vol. 15, No. 7 pp. 4536-4544, 2014

http://dx.doi.org/10.5762/KAIS.2014.15.7.4536
ISSN 1975-4701 / eISSN 2288-4688

센서 네트워크 환경에서 다양한 운영체제 플랫폼을 위한 노드 소프트웨어의 효율적인 개발을 지원하는 도구

이우진[1], 최일우[2*]
[1]세종대학교 정보통신공학과, [2]강남대학교 교양학부

A Tool to Support Efficient Development of Node Software for Various Operating System Platforms in Sensor Network Environment

Woo-Jin Lee[1], Il-Woo Choi[2*]
[1] **Dept. of Information and Communication Engineering, Sejong University**
[2]**Division of General Studies, Kangnam University**

요 약 본 논문에서는 센서 네트워크에서 다양한 운영체제 플랫폼을 위한 노드 소프트웨어를 효율적으로 개발하기 위한 개발 도구를 제안한다. 제안하는 도구는 그래픽 모델 다이어그램 작성, PIM 및 PSM 설계, 코드 자동 생성, 배포 파일 자동 생성 등을 위한 모듈로 구성된다. 제안하는 도구를 통하여 사용자들이 전체적인 센서 네트워크 모델을 작성하고, 속성값을 설정함으로써 각 노드 소프트웨어에 대한 PIM과 타겟 플랫폼에 대한 PSM을 설계하면, PSM에 대한 정보와 타겟 플랫폼에 대한 코드 템플릿을 바탕으로 소스코드가 자동으로 생성된다. 그리고, 응용 소프트웨어 코드를 바탕으로 각 노드에 설치할 수 있는 배포파일을 자동으로 생성할 수 있다. 제안하는 도구는 사용자들이 센서 네트워크에 대한 low-level의 정보를 상세히 알지 못하더라도 손쉽게 다양한 플랫폼에 대한 노드 소프트웨어를 생성할 수 있도록 해준다.

Abstract This paper proposes a development tool to efficiently develop node software for various operating system platforms in a sensor network. The proposed tool consisted of several modules, such as writing graphical model diagram, PIM and PSM design, code generation, and deployment file generation. Through the proposed tool, the users can graphically draw a sensor network model and design the PIM and PSM of the node software by setting the values of the predefined attributes. The source code of the node software is generated automatically from the PSM using the code templates of the target platform. The deployment files for installing node software on each node are generated automatically. The proposed tool helps the users to develop node software easily for a range of target platforms, even though they do not have details of the low-level information for a sensor network.

Key Words : Ubiquitous Sensor Network, Software Development Tool, MDA, Attribute, Multi-Platform

1. 서론

유비쿼터스 센서 네트워크(Ubiquitous Sensor Network, USN)[1]는 저전력, 초경량의 센서들로 구성된 네트워크로 사물과 주변 환경을 감지하여 사용자에게 정보를 전달해 줌으로써, 사용자들은 센서 네트워크를 통하여 언제, 어디서나 정보를 전달받을 수 있는 USN 환경을 구축할 수 있으며 다양한 분야에서 연구, 개발되어지고 있다.

센서 네트워크를 구성하는 노드들은 저전력, 초경량으로 여러 가지 제약사항을 가지므로 노드 소프트웨어는 이러한 제약사항을 고려하여 개발해야 한다. 또한 노드 소프트웨어는 운영체제를 기반으로 개발되는데 센서 네트워크 운영체제는 여러 종류가 존재하므로, 노드 소프트웨어는 타겟 운영체제의 종류 또한 고려되어야 한다.

이와 같이 여러 제약사항 및 다양한 운영체제를 고려

이 논문은 강남대학교 교내연구비 지원을 받아 연구된 것임.
*Corresponding Author : Il-Woo Choi(Kangnam Univ.)
Tel: +82-31-2803-662 email: iwchoi@kangnam.ac.kr
Received March 25, 2014 Revised (1st April 21, 2014, 2nd June 2, 2014) Accepted July 10, 2014

하여 소프트웨어를 개발하는 것은 매우 번거로운 일이다. 이러한 문제를 해결하기 위하여, 센서 네트워크 노드 소프트웨어의 효율적인 개발을 지원하는 여러 가지 도구[2-5]가 개발되었다. 그러나 이러한 도구들은 대부분 특정 운영체제 플랫폼만을 지원하기 때문에 타겟 플랫폼에 따라 서로 다른 도구를 사용해야 하는 불편함이 있다.

따라서 본 논문에서는 다양한 플랫폼에 대한 노드 소프트웨어를 효율적으로 개발할 수 있는 MDA 및 속성 기반의 노드 소프트웨어 개발 지원도구를 제안한다.

2. 관련연구

2.1 센서 네트워크 운영체제 플랫폼

센서 네트워크 운영체제 플랫폼은 센서 노드가 가지는 자원의 제약을 극복하고 다양한 분야의 응용 소프트웨어에 대한 개발을 지원하기 위하여 만들어졌다. 센서 네트워크 운영체제는 운영체제가 가지는 전반적인 기능을 포함하도록 발전하였으며, 다양한 종류의 운영체제[6-13]가 만들어졌다.

이러한 센서 네트워크 운영체제 플랫폼은 센서 네트워크 응용 소프트웨어를 개발하는데 매우 중요한 역할을 한다. 센서 네트워크 응용 소프트웨어는 운영체제의 모듈과 운영체제의 모듈을 사용하는 응용 코드가 합쳐져서 개발되기 때문에, 센서 네트워크 응용 소프트웨어를 개발하는 경우에는 타겟 운영체제 플랫폼이 제공하는 기능을 포함하도록 설계하고 코드를 생성해야 한다.

2.2 기존의 센서 네트워크 응용 소프트웨어 개발 도구

기존의 센서 네트워크 응용 소프트웨어 개발 도구로는 .NET Micro Framework[2], MoteWorks[3], NanoEsto[4], TinyOS Plugin[5] 등을 비롯하여 여러 가지 도구가 있다.

이러한 도구들은 사용자들이 센서 네트워크 응용 소프트웨어를 편리하게 개발할 수 있도록 지원하기 위하여 비주얼한 모델링을 제공하고, 코드의 자동 생성 등을 지원한다.

그러나 대부분의 도구는 하나의 플랫폼에 대한 응용 소프트웨어 생성만을 지원하며, 한 번에 하나의 노드에 대한 응용 소프트웨어만을 개발할 수 있다는 단점이 있다.

본 논문에서는 이러한 기존 도구들의 단점을 극복하기 위하여 다양한 플랫폼에 대한 응용 소프트웨어를 생성할 수 있으며, 한 번에 여러 개의 노드에 대한 응용 소프트웨어를 설계하여 생성할 수 있도록 하는 도구를 제안한다.

3. 다양한 운영체제 플랫폼을 지원하는 속성 기반의 센서 네트워크 노드 소프트웨어 개발 도구

본 장에서는 제안하는 도구의 특징과 도구를 구성하는 모듈 및 도구를 이용한 소프트웨어 개발 절차에 대하여 설명한다.

3.1 도구의 특징

이 절에서는 본 논문에서 제안하는 센서 네트워크 노드 소프트웨어의 개발을 위한 도구의 특징을 설명한다.

3.1.1 MDA 기반 소프트웨어 개발 개념의 적용

MDA(Model Driven Architecture)[14] 기반 개발은 플랫폼에 독립적인 모델인 PIM을 재사용하여 구현 환경에 맞는 PSM을 생성하고, 그로부터 소프트웨어를 개발할 수 있도록 하는 것이다. 따라서 MDA 기반 개발 방법을 사용하면 하나의 모델로부터 다양한 구현 환경을 위한 소프트웨어를 개발할 수 있으므로, 개발 생산성이 증가한다.

센서 네트워크를 위한 운영체제 플랫폼은 매우 다양하므로, 본 논문에서 제안하는 도구는 MDA 기반 개발의 개념을 적용하여 PIM을 재사용하여 다양한 센서 네트워크 운영체제 플랫폼에서 수행 가능한 응용 소프트웨어를 생성하도록 함으로써 센서 네트워크 응용 소프트웨어의 개발 생산성을 증가시킬 수 있도록 하였다.

3.1.2 속성 기반 프로그래밍 개념의 적용

속성 기반 프로그래밍은 응용 소프트웨어 코드 작성 시에 속성을 이용하여 메타 데이터를 추가하여, 실행 시에 동적으로 메타 데이터의 값을 변경하도록 함으로써 코드를 직접 변경하거나 재검파일을 하지 않고 응용 소프트웨어를 재구성할 수 있도록 한다[15].

또한 속성 기반 프로그래밍은 프로그램의 추상화 수

준을 높이는 방법으로, 개발자들이 속성을 이용하여 프로그램을 작성하면 구현 시에 속성의 값에 따라 프로그램 코드가 삽입되도록 함으로써 개발자들이 쉽게 프로그램을 작성하도록 한다[16].

본 논문에서 제안하는 도구는 이러한 속성 기반 프로그래밍의 개념을 확장 적용하였다. 프로그램의 복잡성을 줄이기 위하여 프로그램 작성 시에 속성과 그에 대한 값을 직접 써 넣는 것이 아니라 미리 정의되어 있는 속성의 값을 선택하도록 하였다. 제안하는 도구는 이렇게 선택된 속성의 값에 따라 코드의 템플릿을 조합하여 응용 소프트웨어 코드를 생성할 수 있도록 하였다.

3.2 도구의 구성 모듈

이 절에서는 본 논문에서 제안하는 센서 네트워크 노드 소프트웨어 개발을 위한 도구의 구성 모듈을 설명한다. 제안하는 도구는 그래픽 사용자 인터페이스, 속성 생성기, 모델 작성기, PIM-to-PSM 속성 변환기, 설정정보 생성기, 소스코드 생성기, 템플릿 저장소, 코드 편집기 및 배포 파일 생성기로 구성된다.

3.2.1 그래픽 사용자 인터페이스(GUI)

사용자가 도구를 편리하게 사용할 수 있도록 제공되는 인터페이스이다. 그래픽 사용자 인터페이스를 통하여 사용자는 센서 네트워크 모델 작성 및 노드 소프트웨어의 설계를 비주얼하게 수행할 수 있다.

3.2.2 속성 생성기

본 논문에서 제안하는 도구는 속성을 기반으로 소프트웨어를 설계할 수 있도록 해주는 도구이다. 따라서 소프트웨어 설계에 필요한 속성을 생성해야 한다.

속성 생성기는 노드 소프트웨어를 설계하는데 필요한 속성들을 생성해주는 역할을 한다. 속성 생성기를 통해서 PIM 설계를 위한 공통속성과 각 플랫폼에 맞는 PSM 설계를 위한 종속속성을 생성한다.

센서 네트워크 소프트웨어를 설계하기 위한 속성은 Table 1과 같은 타겟 플랫폼들의 기능으로부터 도출된다.

타겟 플랫폼들의 공통된 기능들을 바탕으로 센서 네트워크 소프트웨어의 PIM 설계를 위한 속성이 생성되고, 각 플랫폼들의 고유한 기능들을 바탕으로 PSM의 설계를 위한 속성이 생성된다.

[Table 1] The functions of target platforms

Function \ Platform	Nano-Qplus	TinyOS
Scheduling	v	v
RF communication	v	v
Real Time Clock		v
Timer	v	v
Sensing(ADC)	v	v
UART	v	v
EEPROM read/write	v	v
Flash Memory read/write	v	
LED	v	v
I2C		v
Hardware ID access		v
WatchDog		v
Hardware potentiometer		v
Encryption/decryption		v
Utility function	v	
System log function	v	
Random number generator		v

Table 2는 타겟 플랫폼들의 공통된 기능으로부터 PIM을 설계하기 위해 필요한 공통속성을 생성한 결과를 보여준다.

[Table 2] Common attributes for PIM design

Common functions and program elements	Attribute
Node name	*nodeName*
Node identifier	*nodeID*
Type of node	*nodeType*
Scheduling	*Scheduler_Enable*
RF communication	*RF_Enable*
Timer	*Timer_Enable*
ADC (sensing)	*ADC_Enable*
UART	*UART (none, printf, scanf, printf&scanf)*
EEPROM read/write	*EEPROM_Enable*
LED	*LED_Enable*

Table 3은 센서 네트워크 소프트웨어의 PSM 설계를 위해 생성한 속성의 예로 Nano-Qplus[6] 플랫폼을 위한 속성을 보여준다.

[Table 3] A part of platform-specific attributes for PSM design for Nano-Qplus

Platform-specific functions and program elements		Attribute
Kind of scheduler		*Scheduler (none, FIFO, PreemptionRR)*
Kind of RF		*RF (Simple, IEEE802.15.4MAC, StarMesh)*
Kind of sensor	Temperature	*Sensor_Temperature_Enable*
	Light	*Sensor_Light_Enable*
	Humidity	*Sensor_Humidity_Enable*

PAN coordinator or not	*isPANCoordinatorNode*
Basic MAC address	*defaultMACAddr*
ID of association permission node (start, end node)	*associationPermitStartNodeID, associationPermitEndNodeID*
ID of next routing node (first, second node)	*nextHopRoutingFirstNodeID, nextHopRoutingSecondNodeID*

Table 4는 TinyOS[7] 플랫폼을 타겟으로 하는 노드 소프트웨어의 PSM 설계를 위해 생성된 종속속성을 보여준다.

[Table 4] A part of platform-specific attributes for PSM design for TinyOS

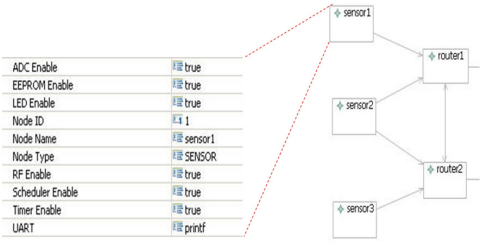
Platform-specific functions and program elements		Attribute
Kind of scheduler		*Scheduler (none, FIFO)*
Kind of RF		*RF (BasicMAC, Broadcast, MultiHop)*
Real Time Clock		*Clock_Enable*
TinySec (encryption/decryption)		*TinySec_Enable*
Kind of sensor	Temperature	*Sensor_Temperature_Enable*
	Light	*Sensor_Light_Enable*
	Humidity	*Sensor_Humidity_Enable*
	Pressure	*Sensor_Pressure_Enable*
	Microphone	*Sensor_Mic_Enable*

제안하는 도구에서, 이러한 노드 소프트웨어를 설계하는데 필요한 속성들은 XML을 통해 정의되고, 속성 생성기는 정의된 XML을 이용하여 속성을 생성한다.

3.2.3 모델 작성기

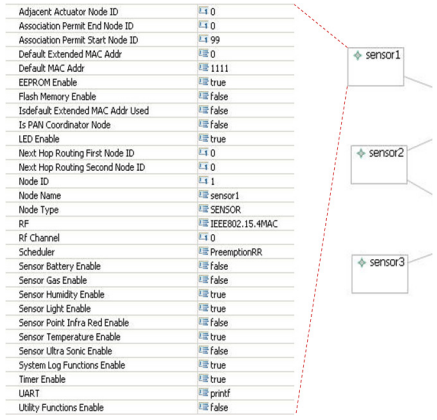
센서 네트워크 모델을 작성할 수 있도록 하는 모듈이다. 모델 작성기는 플랫폼에 독립적인 PIM과 타겟 플랫폼에 종속적인 PSM을 설계하기 위한 모듈로 구성된다.

모델 작성기에서는 모든 플랫폼에 공통적인 속성의 값에 대한 설정을 통하여 PIM을 설계하고, 각 플랫폼에 종속적인 속성의 값에 대한 설정을 통하여 원하는 플랫폼에 대한 PSM을 설계한다.



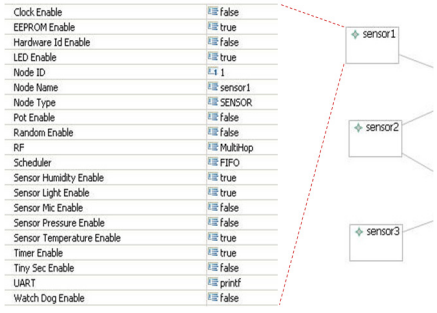
[Fig. 1] Setting of common attributes values for PIM design

Fig. 1은 Table 2와 같이 생성된 공통속성을 이용하여 농작물 관리 시스템의 PIM을 설계한 일부를 보여준다. PIM 설계 시에는 그림에서 sensor1 노드와 같이 센서 네트워크를 구성하는 모든 노드에 대하여 공통속성의 값을 설정해 준다.



[Fig. 2] Setting of platform-specific attributes values for PSM design for Nano- Qplus platform

Fig. 2는 Table 3와 같이 생성된 Nano-Qplus 플랫폼의 종속속성을 이용하여 농작물 관리 시스템의 PSM을 설계한 일부를 보여준다. 그림에서는 sensor1 노드에 대한 종속속성의 값을 설정한 것을 보여주는데, 이와 같은 방법으로 센서 네트워크 모델에 있는 모든 클래스에 대하여 타겟 플랫폼에 따른 종속속성의 값을 설정함으로써 응용 소프트웨어의 PSM을 설계할 수 있다.



[Fig. 3] Setting of platform-specific attributes values for PSM design for TinyOS platform

Fig. 3은 Table 4와 같이 생성된 TinyOS 플랫폼의 종속속성을 이용하여 농작물 관리 시스템을 위한 센서 네

트워크 응용 소프트웨어의 PSM을 설계한 일부를 보여준다.

모델 작성기는 센서 네트워크 모델을 작성하기 위하여 각 노드와 노드 간의 관계를 나타내는 표기법을 제공한다. 이러한 표기법을 이용하여 Fig.5와 같은 센서 네트워크 모델을 작성할 수 있고, 센서 네트워크 모델을 작성하면 각 노드에 대한 속성 값의 설정을 통하여 Fig.2과 같이 PIM을, Fig.3이나 Fig.4와 같이 PSM을 설계할 수 있다.

3.2.4 PIM-to-PSM 속성 변환기

본 논문에서 제안하는 도구는 PIM과 PSM을 설계할 때, 속성의 값을 설정해야 한다. 그리고, PSM을 설계하기 위한 속성은 PIM을 설계할 때 설정한 속성의 값을 바탕으로 타겟 플랫폼에 따라 생성되어야 한다.

PIM-to-PSM 속성 변환기는 PIM 설계 시에 설정한 속성의 값을 바탕으로 타겟 플랫폼을 반영하는 PSM을 설계할 수 있도록 PIM의 속성을 PSM의 속성으로 변환시키는 모듈이다.

모델 작성기를 통하여 PIM이 설계되면, 설계된 PIM에 대한 정보는 XML로 저장된다. PIM-to-PSM 속성 변환기는 PIM에 대한 XML 정보를 바탕으로 타겟 플랫폼에 맞는 PSM 속성으로 변환시킨다.

즉, PIM-to-PSM 속성 변환기는 PIM의 설계 정보를 각 플랫폼에 해당하는 PSM의 속성과 연결해주는 역할을 한다.

3.2.5 설정정보 생성기

설정정보 생성기는 PIM과 PSM 설계 시에 설정한 속성의 값에 대한 정보를 생성하는 모듈이다. 설정정보 생성기에 의해 생성된 PIM에 대한 설정정보는 PSM을 위한 속성 생성에 사용되고, PSM에 대해 생성된 설정정보는 소스코드 생성을 위한 코드 템플릿을 선택할 때에 사용된다.

설정정보 생성기는 모델 작성기를 통하여 설계된 PIM과 PSM에 대한 설정정보를 XML 파일로 생성한다.

3.2.6 소스코드 생성기

소스코드 생성기는 센서 네트워크 모델에 존재하는 각 노드에 대한 응용 소프트웨어의 코드를 자동으로 생성하는 모듈이다.

소스코드 생성기는 PSM에 대한 설정정보와 코드 템플릿을 바탕으로 타겟 플랫폼에 따른 응용 소프트웨어의 코드를 생성한다.

소스코드 생성기는 PSM에 대한 설정정보가 저장되어 있는 XML 파일을 분석하여 각 노드 소프트웨어에서 필요로 하는 모듈코드와 실행코드를 알아내고, 그에 맞는 코드 템플릿을 템플릿 저장소로부터 가져와서 통합하여 노드 소프트웨어에 대한 소스코드를 생성한다.

3.2.7 템플릿 저장소

템플릿 저장소는 노드 소프트웨어에 대한 소스코드의 생성을 위해 필요한 코드 템플릿을 저장하는 모듈이다. 각 운영체제 플랫폼이 제공하는 모듈코드와 실행코드를 바탕으로 설계된 코드 템플릿을 저장한다.

코드 템플릿은 타겟 플랫폼에 따라서 함수, 클래스, 컴포넌트의 형태를 가질 수 있다.

3.2.8 코드 편집기

코드 편집기는 소스코드 생성기에 의해 자동으로 생성된 노드 소프트웨어의 코드를 사용자가 편집하고자 할 때 사용한다.

자동으로 생성된 노드 소프트웨어의 코드를 간단하게 수정하거나 필요한 내용을 추가하고자 할 때 코드 편집기를 통하여 편집할 수 있다.

3.2.9 배포 파일 생성기

최종적으로 생성된 응용 소프트웨어는 각 센서 노드에 배포하여 설치될 수 있는 파일로 만들어져야 한다.

배포 파일 생성기는 생성된 응용 소프트웨어를 물리적인 센서 노드에 설치할 수 있는 파일로 만들어주는 역할을 한다.

3.3 노드 소프트웨어 구축 절차

센서 네트워크를 구성하는 각 노드에 설치될 응용 소프트웨어를 구축하기 위해서는 먼저 전체적인 센서 네트워크 모델을 작성하고, 각 노드 소프트웨어에 대한 PIM을 설계한 후 타겟 플랫폼에 대한 PSM을 설계하여, PSM에 대한 정보와 타겟 플랫폼에 대한 코드 템플릿을 이용하여 소스코드를 생성한다. 그리고 마지막으로 응용 소프트웨어를 각 노드에 설치할 수 있는 배포파일을 만든다.

본 논문에서 제안하는 도구를 이용하여 노드 소프트웨어를 구축하는 절차를 정리하면 다음과 같다.

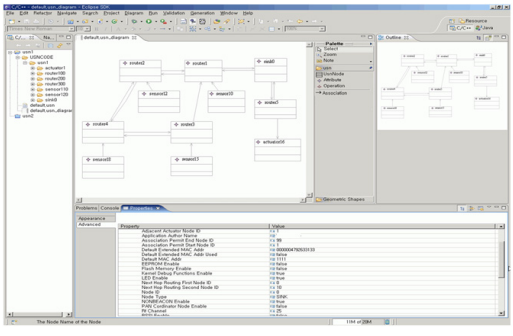
1. USN 모델에 대한 다이어그램을 작성한다.
2. 공통 속성에 대한 값을 설정한다. (PIM 모델링)
3. 타겟 플랫폼에 따라 속성을 변환시킨다. (PIM-to-PSM 매핑)
4. 플랫폼에 종속적인 속성의 값을 설정한다. (PSM 모델링)
5. 노드 소프트웨어의 설정정보를 생성한다.
6. 설정정보와 코드 템플릿을 이용하여 노드 소프트웨어의 소스 코드를 생성한다.
7. 필요한 경우 소스 코드를 편집하여 보완, 완성한다.
8. 노드 소프트웨어의 배포 파일을 생성한다.

4. 사례 및 평가

4.1 개발도구 적용사례

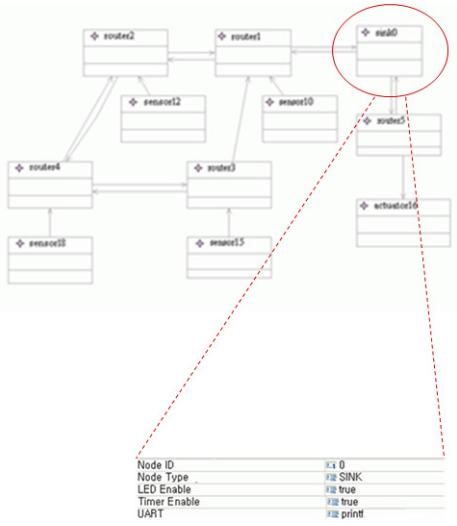
본 장에서는 논문에서 제안한 도구를 개발하여 센서 네트워크 노드 소프트웨어를 생성한 결과를 보인다.

Fig. 4는 개발한 도구의 전체적인 모습을 보여준다. 제안하는 도구는 이클립스 플러그인[17]으로 개발하였다. 도구의 왼쪽 부분은 프로젝트 관리를 위한 부분인데, 하나의 센서 네트워크 모델은 하나의 프로젝트로, 각각의 노드 소프트웨어를 포함하고 있다. 도구의 윗부분은 비주얼하게 센서 네트워크 모델을 작성할 수 있는 부분이며, 아랫부분은 각 노드 소프트웨어의 설계를 위해 필요한 속성과 그에 대한 값을 설정할 수 있는 부분이다. 도구의 메뉴 부분에는 PIM, PSM 설계, 설정정보 생성, 소스코드 생성, 배포파일 생성 등과 같은 기능을 실행할 수 있는 메뉴들이 존재한다.



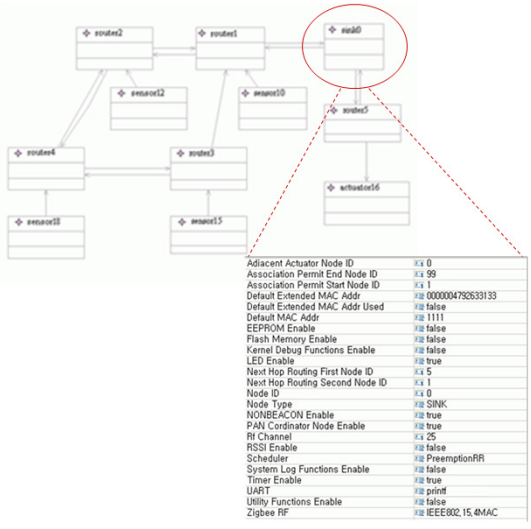
[Fig. 4] Screenshot of the tool

Fig. 5는 PIM을 설계한 예를 보여준다. PIM을 설계하기 위해서는 모델 부분에서 해당하는 노드를 클릭하고, 속성 부분에서 각 속성에 대한 값을 설정하면 된다.



[Fig. 5] An example of the PIM design

Fig. 6은 PSM을 설계한 예를 보여준다. PSM은 PIM을 설계하고, 타겟 플랫폼을 선택한 후에 설계할 수 있다. PIM과 마찬가지로 모델 부분에서 해당 노드를 클릭하고, 속성 부분에서 각 속성에 대한 값을 설정하면 된다.



[Fig. 6] An example of the PSM design

Fig. 7은 PSM 설계 후에 설정정보 생성기가 생성한 설정정보의 예를 보여준다. PSM을 설계한 후에 설정정보 생성 메뉴를 이용하여 그림과 같이 XML로 된 설정정보를 생성할 수 있다. 설정정보에는 PIM과 PSM을 통해 설정한 각 노드 소프트웨어에 대한 속성과 그에 대한 값이 저장되어 있다.

```
<?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>
                <CONFIG_DIGITAL_CLOCK_M>false</CONFIG_DIGITAL_CLOCK_M>
        </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_STAR_MESH_NODE_TYPE>ROUTER</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>100</CONFIG_STAR_MESH_DEFAULT_APP_NODE_ID>
        <CONFIG_STAR_MESH_ADJACENT_ACTUATOR_NODE_ID>0000</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_ASSOCIATION_PERMIT_NODEID_START>101</CONFIG_STAR_MESH_ASSOCIATION_PERMIT_NODEID_START>
        <CONFIG_STAR_MESH_ASSOCIATION_PERMIT_NODEID_END>199</CONFIG_STAR_MESH_ASSOCIATION_PERMIT_NODEID_END>
        <CONFIG_RSSI_M>false</CONFIG_RSSI_M>
        <CONFIG_UTILITY_M>false</CONFIG_UTILITY_M>
        <CONFIG_LOG_M>true</CONFIG_LOG_M>
        <CONFIG_KERNEL_DEBUG_M>true</CONFIG_KERNEL_DEBUG_M>
        <CONFIG_APP_NAME>router100</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>
```

[Fig. 7] An example of the generated configuration information

Fig. 8은 설정정보를 바탕으로 필요한 코드 템플릿을 조합하여 생성된 노드 소프트웨어의 소스 코드의 예를 보여준다. 노드 소프트웨어 코드는 센서 네트워크 운영체제 플랫폼의 특징에 따라 객체지향형 코드, 함수형 코드 등으로 생성된다.

```
#include <io.h>
#include <string.h>
#include <stdlib.h>
#include <progmem.h>
#include <interrupt.h>
#include <sig-avr.h>
#include <pgmspace.h>
#include "config.h"
#include "sig-qplusn.h"

#define TURN_ON_DEBUG
#ifdef STAR_MESH_ROUTE
BYTE NEXT_HOP_ROUTING_FIRST_NODE= 5;
BYTE NEXT_HOP_ROUTING_SECOND_NODE= 1;
#endif
#define LIGHT_LAMP  ACTUATOR0
#define GAS_VALVE   ACTUATOR1
BYTE    MAIN_LAMP_STATUS=TURN_OFF;
BYTE    MAIN_GAS_VALVE_STATUS=TURN_OFF;

void   nlde_node_incoming_data_indication(BYTE *data);
void *start(void *arg);
void *rf_net_scheduling(void *arg);
void rf_recv_data (ADDRESS *srcAddr, INT8 nbyte, BYTE *data);
void *rf_send_data (void *arg);

int main(void)
{
    uint8_t int_handle;

    int_handle = thread_disable_ints();
    initialize_nano_resources();
    thread_enable_ints(int_handle);
    mlme_ll_link_start(NULL,rf_recv_data);
    pthread_create(NULL,NULL,start,(void *)0);
    start_threads();

    return 0;
}
```

```
void rf_recv_data(ADDRESS *srcAddr, INT8 nbyte, BYTE *data)
{
    route_t route;
    BYTE *org_pkt=NULL, packet_type;
    UINT8 i;

    packet_type = (BYTE)(data[1]);

    if (packet_type == SENSOR_DATA_PACKET)
    {
        org_pkt = data;
    }
    else if (packet_type == INDIRECT_PACKET_TRANSMIT)
    {
        i = decode_indirect_packet(data,&route);
        org_pkt = &data[i];
    }

    nlde_node_incoming_data_indication(org_pkt);
}

....

void *start(void *arg)
{
    pthread_attr_t attr;

    attr.schedparam.sched_priority = 2;
    pthread_create(NULL,&attr,rf_net_scheduling,(void *)0);

    attr.schedparam.sched_priority = 1;
    pthread_create(NULL,&attr,rf_send_data,(void *)0);

    return NULL;
}

....
```

[Fig. 8] An example of the generated source code of a node software

4.2 다른 도구와의 비교평가

Table 5는 본 논문에서 제안하는 개발 도구와 센서 네트워크 소프트웨어 개발을 위한 기존의 다른 도구들과 주요 항목을 비교한 표이다. 앞서 살펴본 .NET Micro Framework[2], MoteWorks[3], NanoEsto[4], TinyOS Plugin[5]을 본 논문에서 제안하는 도구와 여러 가지 측면에서 비교를 해보면, 제안하는 도구는 이클립스 플러그인으로 개발하기 때문에 Windows와 Linux 환경을 모두 지원함으로써 사용자에게 높은 융통성을 제공하고, 하나의 특정 플랫폼이 아닌 여러 센서 운영체제 플랫폼을 지원하며, 각각의 타겟 플랫폼이 지원하는 언어를 사용할 수 있기 때문에 타겟 플랫폼에 따라 사용 도구를 바꿀 필요가 없으므로 활용도가 높다. 또한 graphical modeling을 지원하고, 속성 설정으로 통하여 소프트웨어를 설계하므로 사용자는 쉽게 소프트웨어를 설계할 수 있으며, 모델 설계 레벨이 네트워크 레벨이기 때문에 센서 네트워크를 구성하는 모든 노드에 대한 소프트웨어 설계를 한 번에 수행할 수 있어 개발 생산성이 증가된다.

[Table 5] The comparison of the proposed tool with other tools for the development of sensor network software

	.NET Micro Framework	Mote Wokrs	Nano Esto	TinyOS Plugin	Proposed Tool
Language	C#	nesC	C	nesC	C, nesC, etc
Target OS Platform	.NET Micro Framework	TinyOS	NanoQplus	TinyOS	Various OS
Execution Environment	windows	windows	windows Linux	windows Linux	windows Linux
Graphical Modeling	Not Support	Not Support	Not Support	Support	Support
Automated Code Generation	Not Support	Not Support	Support	Support	Support
Design level	node	node	node	node	network
Number of software designed at a time	one	one	one	one	many
Design method	none	none	point-and-click	Component graph	attribute setting

5. 결론

본 논문에서는 다양한 플랫폼에 대한 노드 소프트웨어를 효율적으로 개발할 수 있는 속성 기반의 노드 소프트웨어 개발 지원도구를 제안하였다.

논문에서 제안하는 도구를 사용하면, 플랫폼에 독립적인 모델로부터 다양한 플랫폼을 위한 응용 소프트웨어를 개발할 수 있으며, 단순한 속성 값의 설정을 통하여 모델을 설계하므로 속성의 의미만 알고 있으면 손쉽게 응용 소프트웨어를 설계할 수 있다. 또한 센서 네트워크 모델에 포함되어 있는 모든 노드들에 대한 응용 소프트웨어를 한 번에 설계하고 생성할 수 있어 생산성이 증가된다.

본 논문에서는 노드 소프트웨어의 효율적인 개발을 위한 도구를 제안하였는데, 향후에는 개발된 노드 소프트웨어들의 실행을 모니터링하고 유지보수를 위해 실시간으로 노드 소프트웨어를 업데이트 하는 방안을 연구하여, 개발뿐만이 아니라 유지보수의 효율성도 높일 수 있는 연구를 진행할 예정이다.

References

[1] C. Buratti, A. Conti, D. Dardari, R. Verdone, "An Overview on Wireless Sensor Networks Technology and Evolution," Sensors, vol. 9, no. 8, pp. 6869-6896, Aug. 2009.
DOI: http://dx.doi.org/10.3390/s90906869

[2] D. Thompson, C. Miller, "Introducing the .NET Micro Framework," Microsoft, 2007.

[3] "MoteWorks Getting Started Guide,"
http://www.memsic.com.cn/index.php?option=com_phocadownload&view=category&download=270%3Amoteworks-getting-started-guide&id=6%3Auser-manuals&Itemid=86&lang=zh, 2013.

[4] I. Chun, C. Lim, "NanoEsto Debugger:The Tiny Embedded System Debugger," In Proceedings of the 8th International Conference on Advanced Communication Technology(ICACT), 2006.

[5] "Yeti 2 - TinyOS 2 Plugin for Eclipse,"
http://tos-ide.ethz.ch, 2013.

[6] K. Lee, Y. Shin, H. Choi, S. Park, "A Design of Sensor Network System based on Scalable & Reconfigurable Nano-OS Platform," In Proceedings of the IT SoC Conference, 2004.

[7] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, Kristofer Pister, "System architecture directions for network sensors," In Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), 2000.

[8] C. Han, R. Rengaswamy, R. Shea, E. Kohler, M.Srivastava, "SOS: A dynamic operating system for sensor networks," In Proceedings of the Third International Conference on Mobile Systems, Applications, And Services (Mobisys), 2005.

[9] S. Bhatti, J. Carlson, H. Dai, J. Deng, J. Rose, A.Sheth, B. Shucker, C. Gruenwald, A. Torgerson, R.Han, "MANTIS OS: An Embedded Multithreaded Operating System for Wireless Micro Sensor Platforms," ACM/Kluwer Mobile Networks & Applications, Special Issue on Wireless Sensor Networks, vol. 10, no. 4, pp. 563-579, 2005.

[10] A. Dunkels, B. Grönvall, T. Voigt, "Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors," In Proceedings of the First IEEE Workshop on Embedded Networked Sensors (EmNets), 2004.
DOI: http://dx.doi.org/10.1109/LCN.2004.38

[11] L. Gu, J. Stankovic, "t-kernel: Provide Reliable OS Support for Wireless Sensor Networks," In Proceedings of the 4th ACM Conference on Embedded Networked Sensor Systems (Sensys), 2006.
DOI: http://dx.doi.org/10.1145/1182807.1182809

[12] Q.Cao, T. Abdelzaher, J. Stankovic, T. He, "The LiteOS Operating System: Towards Unix-Like Abstractions for Wireless Sensor Networks," In Proceedings of the 7th International Conference on Information Processing in Sensor Networks, 2008.

[13] A. Eswaran, A. Rowe, R. Rajkumar, "Nano-RK: an energy -aware resource-centric RTOS for sensor networks," In Proceedings of the 26th IEEE International Real-Time Systems Symposium (RTSS), 2005.
DOI: http://dx.doi.org/10.1109/RTSS.2005.30

[14] A. Kleppe, J. Warmer, W. Bast, The Model Driven Architecture: Practice and Promise, Addison-Wesley, 2003.

[15] "attribute-based programming,"
http://webopedia.com/TERM/A/attribute_based_programming.html

[16] G. Wasson, M. Humphrey, Attribute-based programming for grid services, In Proceedings of the GGF9 Workshop on Designing and Building Grid Services, 2003.

[17] Eric Clayberg, Dan Rubel, eclipse Plug-ins, 3rd edition, Addison Wesley, 2008.

이 우 진(Woo-Jin Lee) **[정회원]**

- 2002년 2월 : 숭실대학교 대학원 (컴퓨터공학석사)
- 2007년 2월 : 숭실대학교 대학원 (컴퓨터공학박사)
- 2009년 3월 ~ 현재 : 세종대학교 정보통신공학과 초빙교수

<관심분야>
유비쿼터스 컴퓨팅, 모바일 컴퓨팅, 소프트웨어개발

최 일 우(Il-Woo Choi) **[정회원]**

- 1997년 2월 : 숭실대학교 대학원 (컴퓨터공학석사)
- 2004년 2월 : 숭실대학교 대학원 (컴퓨터공학박사)
- 2007년 3월 ~ 현재 : 강남대학교 교양학부 교수

<관심분야>
개발 프로세스, 재사용, SOA, USN, 모바일 컴퓨팅