2009년도 대한전기학회 하계학술대회 논문집 2009. 7. 14 - 17

UML을 이용한 지그비 어플리케이션모델개발에 관한 연구

정승모*, 유주형, 임동진
한양대학교 전자전기제어계측공학과

A Study on ZigBee Application Model Development using UML

Seung-Mo Jung, Joo-Hyoung Yoo, Dong-Jin Lim
Dept. of Electrical Engineering & Compter Science at Hanyang University

Abstract - ZigBee is a technology that is being rapidly developed since its power consumption is low and the stability of its communication is high. However, documented data which is coded using conventional programming languages such as C or assembly programming language would not be able to fulfill the various requirements upon application development by ZigBee. Unified Modelling Languge (UML) could be one of the alternatives to solve this problem. UML provides a variety of diagrams by which the results of the software development can be presented visually and by which the developers can communicate more spontaneously. This paper shows the results of an ongoing study into the application of model-driven methods for ZigBee Application. Also, this paper shows that this approach is feasible by comparing memory usage, latency, and power consumption of UML modelling code with those of handwritten code.

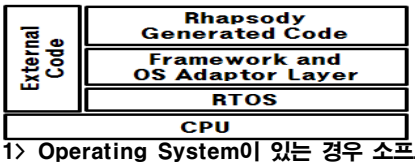
1. 서 론

오늘날 홈오토메이션 및 유비쿼터스 네트워크에 대한 요구가 증대하면서, 통신 기기들 간 배선을 배제해 가까운 거리에 있는 각종 정보처리 기기들 간에 정보를 교환하기 위한 근거리 무선통신기술이 주목받고 있다[1]. 많은 근거리 무선통신기술 중에서 특히 IEEE802.15.4기반으로 저속 근거리 개인 무선통신의 국제 표준 스펙인 ZigBee 기술은 저가격, 저전력을 목표로 하고 ZigBee 기술을 이용하여 노드구성 시 통신의 안정성이 높아 최근 가장 급속한 발전을 하고 있는 기술이다[2]. 또한 설치나 관리가 쉽고 시스템 구조가 간결하여 가정자동화, 공장자동화, 산업자동화에 활발하게 적용될 전망이여서 홈오토메이션과 유비쿼터스 센서네트워크 환경 구축에 중추적 역할을 담당하고 있다[2]. 하지만 기존 C언어와 같은 전통적인 방법으로 ZigBee를 이용한 어플리케이션 개발 시 다양한 요구조건을 충족시키기 위해 개발과정이 복잡질수 있다. 또한, 사용자가 ZigBee를 이용한 어플리케이션 개발 시 기능을 추가할 때 어려움을 가질 수 있다. 그 이유는 기존의 시스템이 C언어와 같은 전통적인 방법으로 코딩되어있을 경우 사용자가 복잡한 시스템을 한 번에 완전히 이해할 수 없기 때문이다. 이로 인한 잘못된 코드해석은 에러를 발생시킬 수 있고 시스템에 심각한 문제를 발생 시킬 수 있다. 이를 해결하기 위해 그래픽한 객체들을 사용하여 가독성을 좋게 만드는 UML(Unified Modeling Language)은 한 가지 대안이 될 수 있다[3][4]. 많은 UML툴 중 몇 가지는 자동적으로 코드 생성 능력을 가지고 있어 독립적인 소프트웨어 모델을 만들 수 있다[5]. 이는 ZigBee기반 임베디드 소프트웨어 시스템에 Model-Driven 방법을 적용시키기에 중요한 역할을 가진다[6]. 본 논문에서는 ZigBee 프로토콜 스택을 Model-Driven 방법에 적용하는 과정의 일부로 ZigBee Application부분을 UML을 사용하여 모델링하였다. 사용한 UML툴로는 자동 코드 생성 능력이 있는 Rhapsody[7]를 사용하였다. 그리고 메모리크기 비교, 지연시간, 소비전력을 기존 C로 작성된 코드와 비교함으로써 이 접근방법의 가능성을 보여준다. 2절 본론에서는 UML을 이용한 Application 모델링 부분에 대해서 설명과 기존 C코드와 모델링 코드의 성능평가를 몇 가지 실험으로 비교해 보았다. 마지막으로 3절에서는 결론 및 추후 연구 과제를 제시한다.

2. 본 론

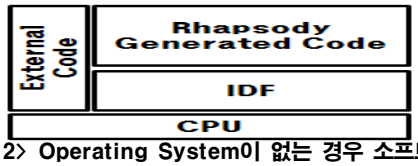
2.1 UML을 이용한 ZigBee 어플리케이션 모델링

Rhapsody에서 자동적으로 코드를 생성하기 위해서는 Windows XP와 VxWorks등과 같은 operating system인 경우 그림 1과 같이 운용체제 종속적인 모델인 OXF(Object eXecution Framework)를 사용하여야 한다. RTOS에서 제공된 API을 OS Adaptor Layer에서 wrapping하여 사용자가 어느 운용체제에서든 동일한 인터페이스로 보이게 만들어 준다. 그리하여 사용자는 모델자체에 집중할 수 있게 되어 프로그램의 완성도를 높일 수 있고 개발 시간 또한 단축하게 만들어 준다.



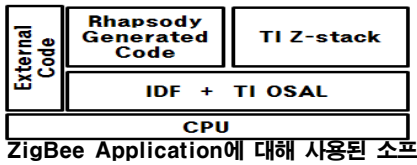
<그림 1> Operating System이 있는 경우 소프트웨어 구조

그리고 operating system이 없는 작은 임베디드 시스템인 경우에는 그림 2와 같이 Rhapsody에서 자동적으로 코드를 생성하기 위해서는 하드웨어 종속적 모델인 IDF(Interrupt Driven Framework)가 필요하다. 이 IDF를 사용하여 operating system이 없는 작은 임베디드 시스템의 OS를 대체하는 역할을 하게 된다.



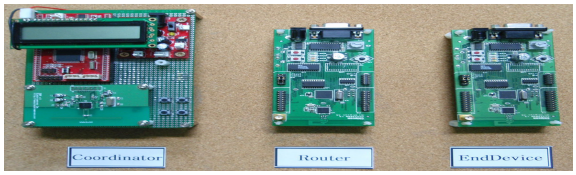
<그림 2> Operating System이 없는 경우 소프트웨어 구조

하지만 ZigBee 어플리케이션을 Rhapsod.3y로 모델링하기 위해 그림 3과 같은 소프트웨어 구조를 사용하였다. TI에서 OS역할을 하는 OSAL에서 중복되는 부분들을 IDF와 통합시켜주었다. UML로 모델링할 때 모든 코드를 모델링 할 필요가 없다. 사용자의 선택에 따라 모델링 코드와 기존 C코드를 함께 사용할 수 있다. 이 소프트웨어 구조를 사용하여 ZigBee 어플리케이션 부분만 모델링하고 어플리케이션을 제외한 나머지 부분들은 기존 C 코드들을 사용하였다.



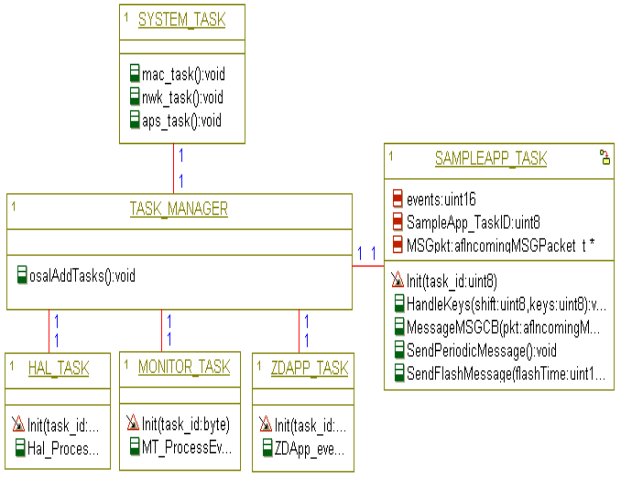
<그림 3> ZigBee Application에 대해 사용된 소프트웨어 구조

테스트 노드로는 그림 4와 같이 코디네이터와 라우터 종단기기로 구성하였다. 각 테스트노드의 CPU는 Atmel의 ATmega128L 마이크로 컨트롤러를 사용하였고 RF 트랜시버 칩으로는 2.4GHz 대역에서 작동하는 TI Chipcon사의 CC2420을 사용하였다. 그리고 4KB의 내부메모리와 확장을 위한 32KB의 외부메모리로 구성되어 있다.



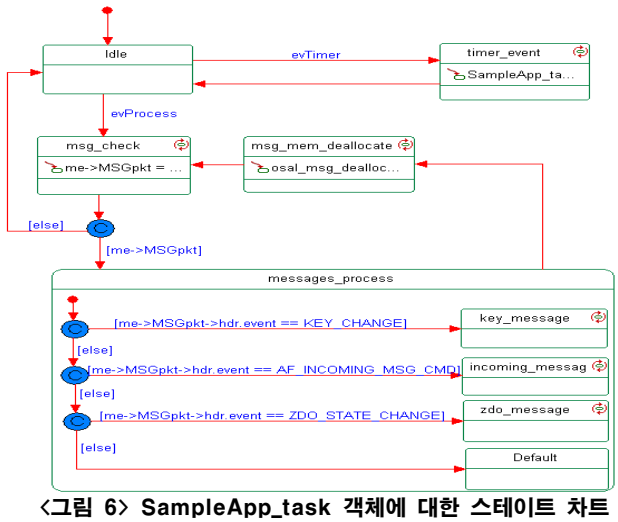
<그림 4> 테스트 노드

테스트 시나리오는 디바이스들의 네트워크가 형성된 후 디바이스들이 가지고 있는 스위치와 조이스틱을 이용하여 다른 디바이스의 LED를 점멸시키게 함으로써 무선통신의 성공여부를 확인하였다. 그리고 테스트 프로그램은 라우터와 종단기기에는 기존의 C언어로 되어있는 코드를 사용하였고 코디네이터에는 UML로 자동적으로 생성된 모델링 코드를 사용하였다. 코디네이터에 대해 모델링 하기위해 그림5와 같이 6개의 오브젝트를 OMD(Object Model Diagram)에 표현하였다. 이것은 ZigBee Alience에서 정의한 레이어들을 참고하여 오브젝트로 표현 하였다. 각각 오브젝트의 이름은 TASK_MANAGER, SYSTEM_TASK, HAL_TASK, MONITOR_TASK, ZDAPP_TASK, SAMPLEAPP_TASK로 구성되어 있다.



<그림 5> ZigBee Application에 대한 객체 모델 다이아그램

각 오브젝트의 역할은 다음과 같다. TASK_MANAGER 오브젝트에서는 테스크들을 관리해주는 역할을 한다. 즉 스케줄러와 같은 역할을 하므로 모든 테스크들을 Association 해 주었다. SYSTEM_TASK 오브젝트에서는 세 개의 테스크를 하나로 묶어주었다. 그 이유는 세 개의 테스크는 소스코드가 아닌 컴파일된 라이브러리로 되어 있기 때문에 하나의 오브젝트로 정의하였다. HAL_TASK에서는 하드웨어 부분을 제어하는 오브젝트로 디바이스 드라이버 역할을 하고 MONITOR_TASK 오브젝트에서는 모든 테스크들의 상태를 감시하기 위한 목적을 가지고 있다. 즉, RS232C를 통해 PC와 테스크들의 상태를 주고 받기 위해서는 꼭 정의해주어야 하는 오브젝트이다. ZDAPP_TASK 오브젝트에서는 네트워크 안에서 디바이스의 기능을 정의하고 디바이스 간의 보안관계를 설정해주는 기능을 담당하는 ZDO(ZigBee Device Object)를 위한 이벤트들을 처리해주는 역할을 한다. 마지막으로, SAMPLEAPP_TASK 오브젝트에서는 사용자가 사용하는 어플리케이션에서 최소한 필요한 변수와 함수들을 오브젝트에 구현하였다. 그리고 SAMPLEAPP_TASK 오브젝트 안에서의 behavior를 묘사하기 위해 어플리케이션에 필요한 이벤트들은 그림 6과 같이 스테이트 차트를 이용해서 표현하였다.



<그림 6> SampleApp_task 객체에 대한 스테이트 차트

TI에서 사용된 이벤트의 종류에는 message이벤트와 timer이벤트가 있다. 초기상태에는 Idle 스테이트에 있다. 만약 message 이벤트가 발생하면 meg_check 스테이트로 천이 하게되고 timer 이벤트가 발생하게 되면 timer_event 스테이트로 천이하게 된다. timer_event 스테이트에서는 사용자가 주기적으로 어떠한 작업을 하고 싶을 때 사용하는 스테이트로 주기적으로 필요한 작업을 이 스테이트 안에 명시해주였다. message_check 스테이트에서는 Z-stack 및 task에서 올라오는 메시를 받는 스테이트이다. 만약 Z-stack 및 task에서 올라오는 메시지가 있다면 messages_process 스테이트로 천이하게 된다. messages_process 스테이트 에서는 네 가지의 서브 스테이트로 구성되어 있다. 먼저 key_message 스테이트 에서는 key의 상태를 주기적으로 검사하다가 key의 상태가 변화되면 감지된 key가 어떤 것인지에 대한 메시지를 처리해주는 스테이트이다. 즉 감지된 key에 대한 작업을 이 스테이트 안에 명시해 주면 된다. 그리고 incoming_message 스테이트에서는 데이터 패킷이 성공적으로 수행되었을 때 발생하는 메시지를 처리해주는 스테이트이다. 따라서 데이터 패킷이 성공적으로 수신되었을 시에 대한 작업을 이 스테이트 안에 명시해 주면 된다. 또한, zdo_message 스테이트에서는 네트워크 상태를 지속적으로 체크하여 네트워크의 상태가 변할 때 Z-stack에서 메시지를 보내게 되는데 그 메시지에 대한 처리를 담당하는 스테이트이고 Default 스테이트에서는 아무런 정의를 해주지 않았다. 모든 메시지를 처리하게 되면 마지막으로 msg_mem_deallocate 스테이트로 천이되어 메시지 할당 공간을 반환하게 된다.

2.2 성능평가 비교

2.2.1 Memory size 비교

기존 TI에서 제공된 ZigBee 프로토콜 스택과 Rhapsody로 모델링한 코드의 메모리 사이즈를 비교 해 보았다. 표1에서는 메모리 사이즈를 비교한 것을 보여준다. 하지만 더 정확하게 비교하기 위해서는 똑같은 환경에서 해야 하므로 Rhapsody만 가지고 있는 IDF의 메모리 크기를 빼게 되면 기존 C코드와의 차이는 Code Memory 4,124 bytes, Data Memory 259 bytes정도 이다.

<표 1> Memory size 비교

	Code Memory	Data Memory
Legacy 코드	58,328 bytes	2,898 bytes
Modeling 코드	66,317 bytes	4,080 bytes
IDF	3,865 bytes	923 bytes
IDF를 제외한 Modeling 코드	62,452 bytes	3,157 bytes

2.2.2 지연시간과 소비전력 비교

지연시간을 비교하기 위해 다음과 같은 실험을 하였다. 원래 TI에서 제공된 어플리케이션은 Key_message를 보내게 되면 다른 디바이스에서는 AF_Incoming_message를 발생시켜 LED를 점멸시킴으로써 무선통신 상태를 확인한다. 하지만 정확한 비교를 위해 Key_message를 쓰지 않고 timer_event를 3초마다 주기적으로 디바이스들이 데이터패킷을 서로 주고받게 하여 지연시간을 측정하였다. 하지만 ZigBee 프로토콜 스택은 네트워크 계층도 포함되어 있기 때문에 외란이 발생할 수 도 있기 때문에 더 정확한 결과를 얻기 위해 100us의 정확도를 가지는 timer를 만들어 데이터패킷의 왕복횟수를 늘려가며 표2와 같이 Latency를 측정하였다.

<표 2> 지연시간 비교

단위:100㎲

왕복 횟수	500회	2000회	5000회	9500회
Legacy 코드	647737	2588686	6461344	12264597
Modeling 코드	649739	2592421	6475020	12302361

평균적으로, 기존 C 코드와 Modeling 코드의 지연시간차이는 Modeling코드가 크게 나왔지만 차이는 0.23%로 작은값이 나왔다. 마지막으로 소비전력을 비교해보았다. 9V의 전압을 가한 후 디바이스들이 계속적으로 데이터 패킷을 주고받도록 하여 소비전력을

측정해 보았다. Modeling코드가 Legacy코드보다는 소비전력이 더 크게 나왔지만 그 차이는 1%미만으로 무시할 수 있는 정도로 작은 값이 나왔다.
>

3. 결 론

본 논문에서는 ZigBee 프로토콜 스택을 Model-Driven 방법에 적용하는 과정의 일부로 우선적으로 Application부분을 UML를 사용하여 모델링하였다. 그리고 메모리 사이즈, 지연시간, 소비전력 등을 비교하여 기존C로 제공된 코드와 별다른 차이가 없음을 보여주었다. 그리하여 이 접근방법이 가능성하는 것을 보여주었다. 추후연구과제에서는 ZigBee 프로토콜 스택에 Model-Driven방법을 완전히 적용시켜 ZigBee를 처음 접해보는 사용자가 쉽게 ZigBee 프로토콜 스택을 이해하고 사용할 수 있게 만들기 위한 연구가 필요하다. 또한 ZigBee 프로토콜 스택 중 사용자가 원하는 layer만 사용할 수 있도록 모델링하기 위해 더 많은 연구가 필요하다.

감사의 글

본 연구는 경기도에서 지원하는 경기도지역협력 연구센터 사업의 지원에 의하여 이루어진 연구로서, 관계부처에 감사드립니다.

[참 고 문 헌]

[1] 유영환, 송형규, “유비쿼터스 근거리 무선 통신 기술” 전자공학회지, vol. 31, no. 12, pp.52-61, 2004

[2] Heon Sik Joo, "The Design and Implementation of Mobile base on Access Control System using ZigBee Method" 한국컴퓨터정보학회 논문지 韓國컴퓨터 情報學會論文誌 第13卷 第2號, 2008. 3, pp. 211 ~ 220

[3] Seo Young-Suk, Kim Ki-Su, Han Young-Choon), "A Study on Factors Affecting the Adoption of UML" 한국경영교육학회, 경영교육논총 經營教育論叢 第43輯, 2006. 8, pp. 135 ~ 152

[4] 강문설, 김태희, “객체지향 소프트웨어 개발방법론의 표준화(UML: Unified Modeling Language),” 「정보처리」, 제5권 5호, 1998년 9월, pp. 64-73.

[5] 이민규, “Understanding MDA(Model Driven Architecture)

[6] Jong-Won Choi, Dong-Jin Lim, "A Study on the Development of Embedded System Software for Ubiquitous Sensor Networks using UML"

[7] Modeling tool Rhapsody from Telelogic, Homepage: http://modeling.telelogic.com/products/rhapsody/index.cfm