UML을 이용한 ZigBee Application Model 개발에 관한 연구

논 문
58-12-31

A Study on ZigBee Application Model Development Using UML

정 승 모* · 유 주 형** · 이 정 한*** · 임 동 진†
(Sung-Mo Jung · Ju-Hyung Yoo · Joung-Han Lee · Dong-Jin Lim)

Abstract - In this paper, a user application based on ZigBee is developed using UML. Conventional development procedure for an application using ZigBee stack requires a tremendous effort, since a developer has to study programming interfaces and analyze sample code to modify and add necessary code. In this study, a sample user application based on ZigBee is modeled using UML and embedded software code is generated using an automatic code generation tool. If the application development method using UML proposed in this paper is used, it is possible for a user to easily develop an application using powerful notations of UML diagrams without paying attention to the details of complex programming code.

Key Words : ZigBee, UML, Modeling, Embedded System

1. 서 론

오늘날 반도체 및 네트워크 기술의 발달로 인해 홈오토메이션 및 유비쿼터스 센서 네트워크에 대한 활용 범위가 증대하고 있다. 그리고 무선 기술의 발달은 통신 기기들 간 배선을 배제해 가까운 거리에 있는 각종 정보 처리 기기들 간에 정보 교환이 수월하도록 해준다. 무선의 장점으로 인한 다양한 형태의 무선 기술 발전과 더불어 근거리 무선 통신망인 WPAN(Wireless Personal Area Network)이 무선 통신 기술의 한축으로서 발전하고 있다. 특히 최근에는 IEEE802.15.4 기반으로 저속 근거리 개인 무선 통신의 국제 표준인 ZigBee 기술이 주목 받고 있다. ZigBee 기술은 소형, 저가격, 저 전력을 목표로 하고 있으며, 이 기술은 노드 구성 시 통신의 안정성이 높아 최근 가장 급속한 발전을 이루고 있다. 또한 설치나 관리가 쉽고 시스템 구조가 간결하기 때문에 가정자동화, 공장자동화, 산업자동화에 활발하게 적용될 것으로 전망하고 있다.[1] 이로 인해 ZigBee기술을 이용한 다양한 Embedded System들이 개발되고 있어 홈 및 빌딩 공장 자동화 시스템을 위한 유비쿼터스 센서 네트워크 환경 구축에 중추적인 역할을 담당하고 있다.[2]

무선 기반의 Embedded System은 소형이면서 고성능의 마이크로프로세서를 탑재하고 있다. 초기의 임베디드 시스템은 간단한 제어프로그램을 내장하고 단순 작업위주로 구성되어 있었으나 최근 생활의 편리성을 위한 사용자의 다양한 요구 조건의 증가로 인해 소프트웨어 및 하드웨어 구조가 점점 더 복잡해지고 있다. 이렇게 복잡한 기능을 수행하기 위해 Embedded System을 좀 더 빠르고 효율적이며 안정적으로 개발하기 위한 다양한 방안에 대해 연구가 진행되고 있다. 그리고 다양한 요구조건을 충족시키기 위해 개발 과정이 보다 복잡해질 경우, 개발자들 간의 불명확한 의사소통 및 신뢰성 저하로 인한 불량품의 증가와 같은 문제점이 늘어나고 있다. 그래서 이러한 문제점들 때문에 최근에 MDA(Model Driven Architecture)와 같은 UML(Unified Model Language) 객체 모델을 이용한 개발 기법이 도입되고 있다.[3][4][5] 모델을 이용한 개발 방법은 기존에 FODA(Feature Oriented Domain Analysis) 과 같은 방법이 있었으나 FODA는 모델을 만들기 위한 객체와 컴포넌트에 대한 언급이 없어 모델을 이용한 임베디드 시스템 개발 시 개발자의 경험에 의한 시스템의 식별이 요구된다. 이 이후 FODA의 단점과 소프트웨어의 재활용 및 유지보수를 향상시키기 위해 MDA가 개발되었다. 그러나 MDA는 모델을 이용하여 시스템 설계를 하기에 용이하나 이 모델을 실제 임베디드 시스템에 적용하기 위한 파일로 변환 시 권고안에 따른 수작업으로 실행 파일을 작성해야 한다. 권고안과 같은 통상FODA 기반의 문서화된 자료들은 개발자들 간의 능력차이로 인한 잘못된 해석을 유발할 수 있으며 서로의사를 확인하기 위해 불필요한 검증 및 시간을 소비한다. 그래서 모델을 이용한 임베디드 시스템 설계 시 하드웨어 플랫폼에 적합한 코들을 자동 생성하는 기능을 가진 툴을 사용한 모델 기반 개발 기법을 제시되었다. 모델 기반 개발 기법은 개발자들이 쉽게 인지할 수 있는 UML(Unified Model

* 준 회 원 : 한양대학교 전자전기제어계측공학과 석사과정
** 비 회 원 : 한양대학교 전자전기제어계측공학과 석사과정
*** 비 회 원 : 한양대학교 전자전기제어계측공학과 박사과정
† 교신저자, 정회원 : 한양대학교 전자전기제어계측공학과 교수
E-mail : limdj@hanyang.ac.kr
접수일자 : 2009년 8월 21일
최종완료 : 2009년 9월 07일

Language) 객체들을 이용하여 제품 개발 시 시스템의 이해와 같은 의사소통을 원활히 하게 해준다. 또한 모델을 이용한 임베디드 시스템 개발은 특정 컴파일러를 이용한 설계 환경이나 하드웨어와 같은 시스템에 종속적이지 않기 때문에 소프트웨어의 재사용이 가능하다. 그리고 모델을 자동으로 소스코드로 변환 시켜 주는 툴을 사용하기 때문에 다수의 개발자로부터 생산된 제품의 성능을 일정하게 유지 시킬 수 있다.[6][7]

본 논문에서는 ZigBee 어플리케이션 개발 시 다양한 요구조건을 충족시키면서 쉽게 기능을 추가 할 수 있도록 모델 개발 기법 기반의 UML툴을 이용한 독립적인 소프트웨어 모델을 개발하는데 있다. 기존의 C언어와 같은 전통적인 방법으론 개발 과정이 복잡해 질 수 있고, 복잡하게 구성된 시스템은 사용자로 하여금 한 번에 완전히 이해하기가 어렵기 때문에 어플리케이션 개발 시 잘못된 코드를 생성할 수도 있다. 잘못된 코드로 인한 오류의 발생은 시스템에 심각한 문제점을 발생 시킬 수 있다. 그래서 이를 해결하기 위해 그래픽한 객체들을 사용하여 가독성을 좋게 만드는 UML의 다양한 모델과 다이어그램을 이용, 임베디드 시스템에서 수행되는 실행 모델을 작성하고, 이를 바탕으로 하드웨어에 실행 가능한 파일을 생성해 낸 뒤 이를 실제 시스템에 다운로드하여 그 기능을 점검하고자 한다. 이를 증명하기 위해 Taxas Instrument 사에 제공된 ZigBee 프로토콜 스택인 Z-Stack을 모델 기반 개발 방법에 적용하는 과정의 일부로 ZigBee Application부분을 UML을 사용하여 모델링하였다. 사용한 모델 기반 개발 기법의 UML툴로는 자동 코드 생성 능력이 있는 Rhapsody를 사용하였다. 그리고 기존의 레거시 코드와 모델 기반의 생성된 코드로부터 생성된 실행 파일의 메모리크기 비교, 지연시간, 소비전력을 비교함으로써 이 접근방법의 가능성을 보여준다. 2절 본론에서는 ZigBee와 UML기반의 모델 기반 방법에 대해 설명하고 이를 이용한 개발 순서를 제시한다. 그리고 ZigBee Application 모델링 부분에 대한 모델 개발 및 임베디드 시스템에서 실행 가능한 파일을 생성해내는 IDF(Interrupt Driven Framework)에 대해 분석하고 3절에서는 기존의 개발 방법과 모델 기반의 개발 방법에 대한 성능평가를 위해 기존 C코드와 모델링 코드의 메모리 용량, 실행 속도 및 전력소비량을 측정하는 실험을 하였다. 마지막으로 4절에서는 결론 및 추후 연구 과제를 제시한다.

2. ZigBee 및 UML기반의 모델 개발

2.1 ZigBee 와 IEEE 802.15.4 표준

ZigBee Alliance는 저 전력 , 저가격의 무선통신을 가능하게 하는 통신 규약으로서, MAC과 PHY는 IEEE802.15.4 의 규격을 그대로 따르고 그 상위의 Layer들의 동작 및 역할을 정의하는 표준이다. ZigBee 표준에서는 라우팅 프로토콜이 정의된 NWK 계층과, Application과 NWK 계층의 연결 및 통신 서비스를 지원하기 위한 Application Support Sub-layer(APS), 노드들의 통신 설정 값 및 스택 초기화 내용등이 정의된 ZigBee Device Object(ZDO) 그리고 무선 통신에서의 보안과 관련된 내용이 기술되어 있다.

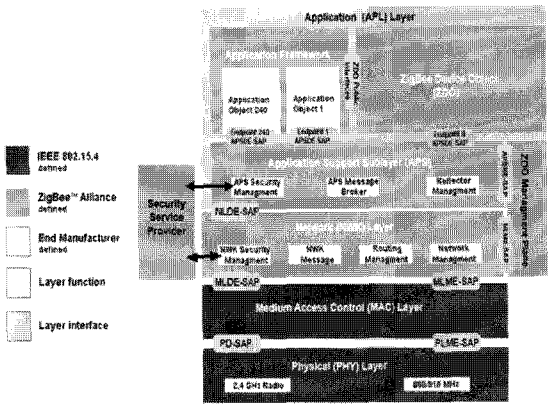

그림 1 ZigBee Stack 구조
Fig. 1 ZigBee Statck structure

2.2 UML기반의 모델 개발 기법

UML(Unified Modeling Language)은 소프트웨어 시스템을 모델링하기 위한 언어로 개발되었다. UML을 사용함으로서 소프트웨어 개발자들은 시스템에 대한 모델을 만들 수 있다. UML은 OMG(Object Management Group)에서 관리되고 있으며 2003년 8월 UML2.0 이 승인되었다.UML은 13개의 다이어그램으로 구성되어 있으며 크게 세가지로 분류된다.

- 행위 다이어그램 : 시스템 또는 비즈니스 프로세스의 행위적 측면을 표현하는 형태의 다이어그램. 액티비티,스테이트 머신, 유스케이스 다이어그램 및 4개의 상호 작용 다이어그램들이 이에 포함된다.

- 상호작용다이어그램 : 행위 다이어그램의 부분집합으로 객체들 간의 상호 운용을 표현한다. 커뮤티케이션 ,인터랙션 오버뷰, 시퀀스 ,타이밍 다이어그램이 이에 포함된다.

- 구조 다이어그램 : 시간과 상관없이 시스템 또는 비즈니스 프로세스의 구성요소들을 표현하는 형태의 다이어그램. 클래스, 합성구조, 컴포넌트, 배치 ,오브젝트, 패키지 다이어그램이 이에 포함된다.

이 UML을 이용한 모델 기반 개발 기법은 설계 환경이나 시스템에 종속적이지 않고 하드웨어에 실행 가능한 소프트웨어를 개발하기 위한 기술이다. 모델 기반 개발 기법은 사람이 직접 작성한 독립적인 모델을 특정 플랫폼에 종속적이며 실행 가능한 코드로 자동으로 변환시킨다. 이를 통해 하드웨어의 특성이나 개발자의 특이성에 영향을 받지 않고 어플리케이션이나 행위를 독립적으로 정의할 수 있고 차후 시스템 변경 시 모델을 반복해서 작성할 필요가 없다.

2.3 모델 기반 개발 절차

모델 기반 개발은 3단계를 거쳐서 개발한다. 각 모델들은 UML의 표기법을 준수한다. 본 연구에서는 자동 코드 생성 기능이 있는 UML도구로서 IBM(구 TeleLogic) 사의 Rhapsody를 사용하였다.

Step 1 : 플랫폼에 독립적 모델 개발

독립적 모델이란 어떤 하드웨어 플랫폼에 종속되지 않는 모델을 말한다. 모델로서 표현되는 UML은 주로 정적인 구조를 나타내는 오브젝트 다이어그램을 사용한다. 오브젝트 다이어그램 내에 오브젝트를 생성하고 여기에 속성과 동작

내용을 추가한다. 그리고 행위적인 측면을 정의하기 위해 스테이트 차트 다이어그램을 작성한다. 스테이트 차트는 이벤트 방식의 코드를 생성하여 준다.

Step 2 : 하드웨어에 실행 가능한 파일 생성

Rhapsody는 자체 모델을 코드로 변환해주는 컴파일러를 가지고 있지 않다. 그렇기 때문에 해당 플랫폼의 MCU에 맞는 코드를 생성하기 위해선 IDF(Interrupt Driven Framework)가 필요하다. IDF는 Rhapsody에서 만든 모델들을 해당 MCU에 맞게 코드 변환 과정을 수행한다. 다양한 MCU별로 IDF가 제작되기 때문에 상위 모델은 하드웨어와 독립적인 특성을 가진다.

Step 3 : 실행 가능한 파일을 하드웨어에 탑재

작성된 모델이 IDF와 해당 MCU의 컴파일러를 통해 기계어로 변환되면 임베디드 시스템에 다운로드하여 동작여부를 확인한다.

2.4 Interrupt Driven Framework

Rhapsody에서 자동적으로 코드를 생성하기 위해서는 Windows XP와 VxWorks등과 같은 operating system인 경우 그림 2과 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.

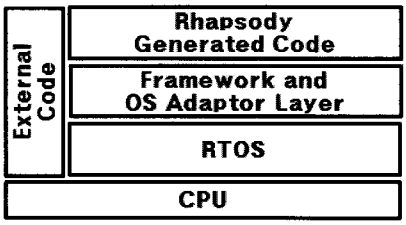

그림 2 Operating System이 있는 경우 소프트웨어 구조
Fig. 2 Software structure with an operating system

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 3)

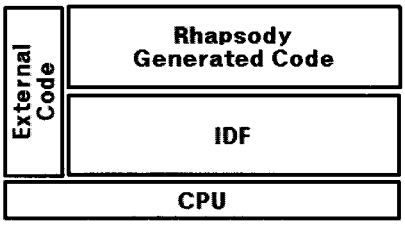

그림 3 Operating System이 없는 경우 소프트웨어 구조
Fig. 3 Software structure without an operating system

하지만 ZigBee 어플리케이션을 Rhapsody로 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 C코드들을 사용하였다.(그림 4)

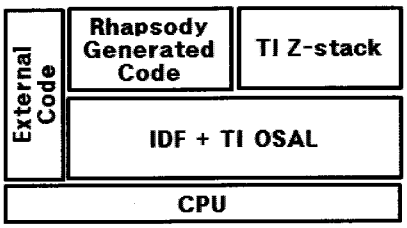

그림 4 ZigBee Application에 대해 제안된 소프트웨어 구조
Fig. 4 Proposed software structure for ZigBee Application

3. 모델 및 실험환경 구성

ZigBee 기반의 어플리케이션을 개발하기 위해 실험환경으로 그림 5와 같이 코디네이터와 라우터, 종단 디바이스 3가지 타입의 디바이스로 구성하였다. 각 디바이스의 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. 이 디바이스들을 구동하기 위한 소프트웨어로는 TI사의 ZigBee Stack인 Z-Stack를 사용하였다.(그림 5)

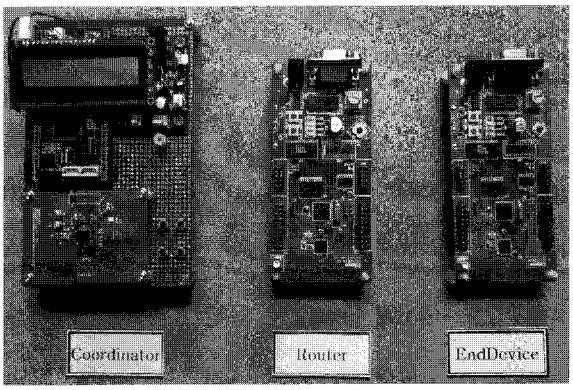

그림 5 실험 환경 구성
Fig. 5 Test hardware

모델을 이용하여 개발 할 테스트 시나리오는 코디네이터와 디바이스들의 무선을 이용한 네트워크가 형성된 후 디바이스들이 가지고 있는 스위치와 조이스틱을 이용하여 다른 디바이스의 LED를 점멸시키게 함으로써 무선통신의 성공여부를 확인한다. 테스트 프로그램은 라우터와 종단 디바이스에 기존의 C언어로 되어있는 코드를 사용하였고 코디네이터

에는 UML로 자동적으로 생성된 모델링 코드를 사용하였다. 코디네이터에 대해 모델링 하기위해 그림 5와 같이 6개의 오브젝트를 OMD(Object Model Diagram)에 표현하였다. 이것은 ZigBee Alience와 Z-Stack에서 정의한 레이어들을 참고하여 오브젝트로 표현 하였다. 각각 오브젝트의 이름은 TASK_MANAGER, SYSTEM_TASK, HAL_TASK, MONITOR_TASK, ZDAPP_TASK, SAMPLEAPP_TASK로 구성되어 있다.(그림 6)

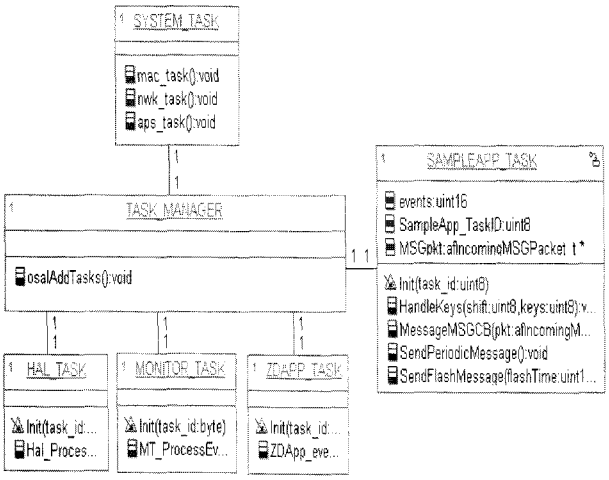

그림 6 ZigBee Application에 대한 객체 모델 다이아그램
Fig. 6 Objcet model diagram for the ZigBee Application

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 주고받기 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 7과 같이 스테이트 차트를 이용해서 표현하였다.

Taxas Instrument에서 제공된 Z-Stack의 구조를 살펴 보면 두가지의 이벤트에 의해 동작된다. Z-Stack에서 사용된 이벤트의 종류에는 message와 timer이벤트가 있다. 하드웨어

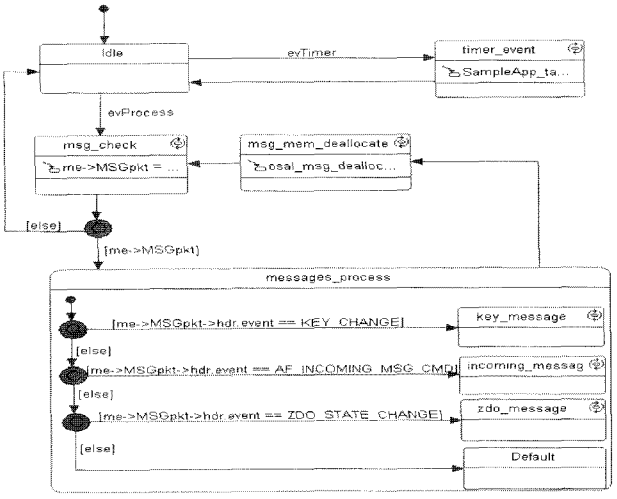

그림 7 SAMPLEAPP_TASK 객체에 대한 스테이트 차트
Fig. 7 State Chart for the SAMPLEAPP_TASK object

에 전원이 인가되고 각 디바이스가 코디네이터에 접속하여 네트워크가 구성되면 초기상태 Idle 스테이트가 된다. 만약 HAL계층을 통해 message 이벤트가 발생하면 meg_check 스테이트로 천이 하게 되고, 자체적으로 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 명시하면 된다. message_check 스테이트에서는 Z-stack의 task의해 발생된 메시지를 받는 스테이트이다. 만약 task에서 발생되는 메시지가 있다면 messages_process 스테이트로 천이하게 된다. messages_process 스테이트 에서는 외부와 연결된 하드웨어의 종류에 따라 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. 그리고 incoming_message 스테이트에서는 데이터 패킷이 성공적으로 수행되었을 때 발생하는 메시지를 처리해주는 스테이트이다. 따라서 데이터 패킷이 성공적으로 수신되었을 시 필요한 수행 작업을 이 스테이트 안에 명시해 주면 된다. 또한, zdo_message 스테이트에서는 네트워크 상태를 지속적으로 체크하여 디바이스의 연결 및 해제와 같은 네트워크의 상태가 변할 때 메시지를 발생한다. 그리고 이 스테이트이는 발생된 메시지에 대한 처리와 수행 역할을 정의하고 있다. Default 스테이트에서는 그 외의 상황을 위해 예약해놓은 확장 스테이트이다. 모든 메시지를 처리하게 되면 마지막으로 msg_mem_deallocate 스테이트로 천이되고 메시지의 의해 할당된 공간을 Z-stack에게 반환하게 된다.

4. 성능 분석

2.2.1 Memory size 비교

기존 TI에서 제공된 ZigBee 프로토콜 스택과 Rhapsody

로 모델링한 코드의 메모리 사이즈를 비교 해 보았다. 표 1에서는 메모리 사이즈를 비교한 것을 보여준다.

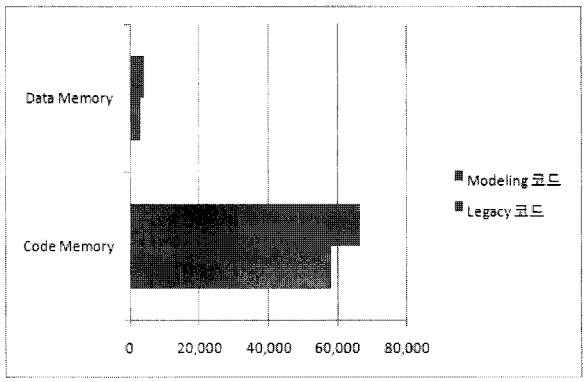

그림 8 Memory size 비교
Fig. 8 Comparison of Memory size

생성된 코드를 정확하게 비교하기 위해 코드의 구성여부를 분석하면 Rhapsody에 의해 생성된 코드는 작성된 모델의 실행 코드 변환 시 IDF가 포함되는 것을 볼 수 있다. 그래서 Rhapsody만 가지고 있는 IDF의 메모리 크기를 생성된 코드의 크기에서 제하게 되면 기존 C코드와의 차이는 Code Memory 4,124 bytes, Data Memory 259 bytes정도 이다.

표 1 Memory size 비교
Table 1 Comparison of Memory size

	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 비교를 위해 Key_message를 쓰지 않고 timer_event를 3초마다 주기적으로 디바이스들이 데이터패킷을 서로 주고받게 하여 지연시간을 측정하였다. ZigBee 프로토콜 스택은 네트워크 계층도 포함되어 있어 무선 통신 및 코디네이터와 디바이스 연결 시 외란이 발생할 수 있기 때문에 더 정확한 결과를 얻기 위해 100us의 정확도를 가지는 timer를 만들어 데이터패킷의 왕복횟수를 늘려가며 표 2와 같이 Latency를 측정하였다.

평균적으로, 기존 C 코드와 Modeling 코드의 지연시간차이는 Modeling코드가 크게 나왔지만 차이는 0.23%로 작은 값이 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.

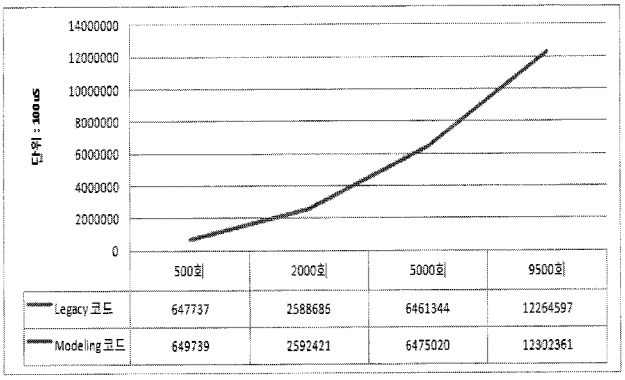

그림 9 지연시간 비교
Fig. 9 Comparison of Latency

3. 결 론

본 논문에서는 ZigBee 프로토콜 스택을 Model-Driven 방법에 적용하여 우선적으로 Application의 일부분을 UML을 사용하여 모델링하였다. 그리고 메모리 사이즈, 지연시간, 소비전력 등을 비교하였고, 기존 C로 제공된 코드에 비해 다소 코드 사이즈가 증가하지만 실행 속도에서는 별다른 차이가 없음을 보여주었다. 코드의 증가는 IDF의 추가 및 모델의 코드 변환 시 추가적으로 생성되는 코드로 판별되었고 다양한 하드웨어 플랫폼에 적용하기 위해서는 최적화할 필요가 있다. 추후 연구 과제에서는 ZigBee 프로토콜 스택에 모델 기반 개발 기법을 완전히 적용시켜 ZigBee를 처음 접해보는 사용자가 쉽게 ZigBee 프로토콜 스택을 이해하고 사용할 수 있게 만들기 위한 연구를 수행할 예정이다. 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 經營敎育論叢 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] 김우열, "MDA(Model Driven Architecture)기반의 임베디드 소프트웨어 모델링에 관한 연구", 홍익대학교 대학원 석사학위논문, 2005년.
[6] J. W. Choi and D. J. Lim, "A Study on the Development of Embedded System Software for Ubiquitous Sensor Networks using UML", Proc. FeT' 2009 IFAC International Conference, pp.230-233, 2009.
[7] Modeling tool Rhapsody from Telelogic, Homepage: http://modeling.telelogic.com/products/rhapsody/index.cfm

저 자 소 개

정 승 모 (鄭 勝 模)

1980년 10월 25일생. 2006년 강릉대학교 전자공학과 졸업(학사). 2008년 ~ 현재 한양대학교 전자전기제어계측공학과 석사과정으로 재학 중. 관심연구 분야는 Embedded Software, ZigBee Software, Model-Driven Method.
Tel : 031-400-4034
Fax : 031-407-9070
E-mail : seung8040@hanyang.ac.kr

유 주 형 (柳 周 亨)

1980년 3월 13일생. 2006년 강릉대학교 전자공학과 졸업(학사). 2009 ~ 현재 한양대학교 전자전기제어계측공학과 석사과정으로 재학 중. 관심연구 분야는 Embedded Software, ZigBee Software, Model-Driven Method.
Tel : 031-400-4034
Fax : 031-407-9070
E-mail : potato97@naver.com

이 정 한 (李 政 漢)

1973년 11월 6일생. 2001년 대진대학교 통신공학과 졸업(학사). 2003년 광운대학교 제어계측공학과 졸업(석사). 현재 한양대학교 전자전기제어계측공학과 박사과정으로 재학 중. 관심연구 분야는 IEEE1451, 임베디드 UML.
Tel : 031-400-4034
Fax : 031-407-9070
E-mail : ranties@hanyang.ac.kr

임 동 진 (林 東 進)

1956년 7월 15일생. 1979년 서울대학교 전기공학과 졸업(학사). 1981년 동 대학원 전기공학과 졸업(석사). 1988년 University of Iowa, U.S.A, Electrical and Computer Engineering 졸업(박사).
1982~1983년 LG전자 연구원.
1988~1991년 RIST 연구원
1991년~현재 한양대학교 공학대학 전자컴퓨터공학부 교수.
Tel : 031-400-5212
Fax : 031-407-9070
E-mail : limdj@hanyang.ac.kr