TeloSIM: Telos 형 센서노드를 위한 명령어 수준 센서네트워크 시뮬레이터

(TeloSIM: Instruction-level Sensor Network Simulator for Telos Sensor Node)

조 현 우 † 김 형 신 ††
(Hyunwoo Joe) (Hyungshin Kim)

요 약 센서 네트워크의 특성상 설치 후, 사람이 직접 초소형의 센서 노드들을 일일이 관리할 수 없기 때문에, 센서 노드를 직접 설치하기 이전에 시뮬레이션을 통해 각 센서노드들의 네트워크 환경을 미리 확인하고 점검하는 작업은 매우 중요하다. 센서네트워크 통신 프로토콜이나 어플리케이션은 데이터의 송수신 타이밍이 매우 중요하다. 하드웨어의 동작타이밍을 정확히 모델링 하여 시간에 데이터를 처리 송수신하는 사이클이 정확한 시뮬레이션이 요구된다. 이를 위해 잘 알려진 방법은 명령어 수준의 시뮬레이션 방법이다.

본 연구에서는 Telos 형 센서노드를 위한 명령어 수준의 센서네트워크 시뮬레이터인 TeloSIM을 구현했다. Telos 는 중앙처리장치인 MSP430과 라디오모듈인 CC2420를 사용하며 최근 가장 많이 쓰이고 있는 센서노드이다. MSP430은 센서노드에서 사용되고 있는 중앙처리장치 가운데 가장 적은 에너지를 소모하며, CC2420은 Zigbee를 지원하기 때문이다. 하지만 현재까지 개발된 명령어 수준의 센서네트워크 시뮬레이터는 대부분 Atmega128을 지원하는 시뮬레이터이거나 CC2420을 지원하지 못하는 시뮬레이터들이다. 따라서 본 논문에서는 소개하는 TeloSIM은 Telos를 이용하여 센서네트워크를 연구하는 개발자에게 도움을 줄 수 있다. TeloSIM은 명령어 수준의 시뮬레이터로 사이클이 정확한 장점을 갖고 있고 하드웨어를 정확히 모델링 하여 운영체제나 특정 기능 구현에 상관없이 하드웨어를 직접 이용하는 것과 동일하게 사용할 수 있으며, 다수의 센서노드를 동시에 시뮬레이션 할 수 있다. 그리고 GUI 도구를 제공하여 사용자가 시뮬레이션 결과를 쉽게 볼 수 있도록 하였다.

키워드 : 센서네트워크, 시뮬레이터, 명령어 수준 시뮬레이터, 센서네트워크 시뮬레이터, Telos, MSP430, CC2420

Abstract In the sensor network, many tiny nodes construct Ad-Hoc network using wireless interface. As this type of system consists of thousands of nodes, managing each sensor node in real world after deploying them is very difficult. In order to install the sensor network successfully, it is necessary to verify its software using a simulator beforehand. In fact Sensor network simulators require high fidelity and timing accuracy to be used as a design, implementation, and evaluation tool of wireless sensor networks. Cycle-accurate, instruction-level simulation is the known solution for those purposes.

In this paper, we developed an instruction-level sensor network simulator for Telos sensor node as named TeloSIM. It consists of MSP430 and CC2420. Recently, Telos is the most popular mote because MSP430 can consume the minimum energy in recent motes and CC2420 can support Zigbee. So that TeloSIM can provide the easy way for the developers to verify software. It is cycle-accurate in instruction-level simulator that is indispensable for OS and the specific functions and can simulate scalable sensor network at the same time. In addition, TeloSIM provides the GUI Tool to show result easily.

Key words : Sensor network, Simulator, instruction-level simulator, Sensor network simulator, Telos, MSP430, CC2420

· 이 연구는 2009년도 충남대학교 학술연구비에 의해 지원되었음

† 학생회원 : 충남대학교 컴퓨터공학과
jhwzero@cnu.ac.kr
†† 종신회원 : 충남대학교 컴퓨터공학과 교수
faith.kim.cnu@gmail.com
(Corresponding author임)

논문접수 : 2010년 6월 9일
심사완료 : 2010년 10월 6일

정보과학회논문지: 컴퓨팅의 실제 및 레터 제16권 제11호(2010.11)

1. 서 론

센서네트워크는 인간과 컴퓨터, 사물이 유기적으로 연계되어 유비쿼터스 컴퓨팅의 핵심 인프라로 주목 받고 있다. 센서네트워크는 다수의 초소형 컴퓨터들이 무선통신을 이용하여 망을 구성하며, 각 컴퓨터들은 소형 배터리를 사용하여 전원을 공급하고 자신이 가지고 있는 센서들을 이용하여 측정한 데이터를 다중 홉 방식으로 기지국까지 전송할 수 있는 시스템이다. 센서네트워크는 그 특성상 사람이 직접 갈 수 없는 곳에 설치될 수 있다. 그리고 일반적으로 설치 후 센서 노드들을 일일이 관리하기 힘들다. 때문에 센서노드 직접 설치하기 이전에 시뮬레이션을 통해, 각 센서노드들의 네트워크 환경을 미리 확인 하고 점검하는 작업이 필요하다. 하지만 센서네트워크 기술분야는 최근 10년 동안 급속한 발전을 이루었으나, 센서네트워크 응용 프로그램을 쉽고 간편하게 개발하기 위한 시뮬레이터는 많이 부족한 상태이며, 이는 응용 프로그램을 빠르게 개발하고, 검증하는데 장애가 되어 결국 핵심 애플리케이션(Killer Application)의 부재로 이어질 수 있다. 따라서 응용프로그램 개발자에게 도움을 줄 수 있는 좀더 편리하고 실제 사용이 가능한 센서네트워크 시뮬레이터에 대한 요구가 커지고 있다.

최근까지 센서네트워크 시뮬레이션을 위한 다양한 연구들이 진행 중이며, 센서네트워크 시뮬레이터는 그 구현 방법에 따라 네트워크 시뮬레이터, 운영체제 기반 시뮬레이터, 명령어 수준 시뮬레이터로 크게 나눌 수 있다. 네트워크 기반 시뮬레이터와 운영체제 기반 시뮬레이터들은 센서네트워크 동작을 이해하는데 도움을 주고 있으나, 명령어기반 시뮬레이터와 달리 하드웨어가 동작하는 것과 완벽히 동일한 시뮬레이션은 불가능하다. 실제로 센서네트워크 시뮬레이션의 타이밍 정확도(Timing accuracy)는 사용자가 네트워크 동작이나 패킷의 송수신 시간의 동작 확인을 검증하는데 필요한 요소이고, 시스템 설계, 구현, 평가를 위한 도구로써 높은 정확도와 타이밍 정확도가 요구된다. 명령어 수준 시뮬레이터들은 이러한 목적을 갖고 개발되었다. 실제로 다수의 센서노드 간 동기화 문제를 해결하거나, 보안에서의 패킷의 송수신 시간을 정확히 요구하는 연구 등에서는 명령어 수준의 정밀한 시뮬레이터가 특히 요구된다.

본 연구에서는 Telos[1] 형 센서노드를 위한 명령어 수준의 시뮬레이터를 구현하는 것을 목적으로 한다. 일반적으로 센서네트워크에서 사용되는 센서노드는 중앙처리장치와 무선라디오 모듈로 구성되어있다. 표 1에서 보는 바와 같이 MSP430[2]은 다른 Atmega128에 비해 전력 소모가 낮기 때문에 최근에는 Atmega128[3]을 사용하는 센서노드에 비해 MSP430을 사용하는 센서노드를 많이 이용하는 추세이다. 그리고 무선라디오 모듈의 경우 센서네트워크 초기에 많이 사용되었던 CC1000[4]에 비해 최근에는 Zigbee(IEEE 802.15.4)[5]를 지원하는 CC2420[6]이 많이 사용되고 있다. Telos는 MSP430과 CC2420으로 구성되어 있는 센서노드로서 Atemega128로 구성된 센서노드인 MICA2[7], MICAz[8] 비해 많이 사용되고 있다. 하지만 Telos를 하드웨어 수준에서 동일하게 모델링 하여 명령어 수준으로 구현한 시뮬레이터는 Cooja/MSPSim[9]이 유일하다. 하지만 CC2420의 모델의 경우 하드웨어 수준에서의 모델링이 아닌 특정 기능 동작 시뮬레이션으로 TinyOS[10]와 ContiKi[11]가 지원하는 기능만이 구현되어 있다. 따라서 본 연구에서는 MSP430을 명령어 기반으로 하드웨어 에뮬레이션이 가능하고 CC2420의 내부 하드웨어가 정밀하게 모델링 되어 있는 TeloSIM을 구현하였다. TeloSIM은 MSP430용으로 컴파일된 실행파일을 가상의 센서노드에 각각 적재하여 다수의 센서노드들을 시뮬레이션 할 수 있으며, GUI 환경을 제공하여 사용자가 좀 더 편리하게 사용할 수 있다.

표 1 센서노드 중앙처리장치의 전력소모 비교

중앙처리장치	동작 중	저전력 모드 (32KHz Clock)	전원 정지
Atmega128L (Atmel)	10mA (8MHz, 3V)	20μA (3V)	< 10μA (3V)
MSP430F149 (TI)	0.42mA (1MHz, 3V)	1.6μA (3V)	0.1μA (3V)

본 논문의 구성은 다음과 같다. 1장 서론에 이어 2장에서는 관련연구를 알아보고, 3장에서는 시스템의 구성과 4장에서는 시뮬레이터의 실제구현에 대해 살펴본다. 5장에서는 구현한 시뮬레이터의 결과를 보여주고 6장에서 결론을 맺고 향후 계획을 밝힌다.

2. 관련 연구

센서네트워크 시뮬레이터는 구현 방법에 따라 크게 네트워크 시뮬레이터, 운영체제 기반 시뮬레이터, 명령어 수준 시뮬레이터로 나눌 수 있다.

네트워크 시뮬레이터는 센서네트워크 시뮬레이션을 위해 최초로 시도된 시뮬레이터들로써, 이들은 실제 센서네트워크 플랫폼을 시뮬레이션 하는 것이 아니라 통신계층(communication layer)에서 통신 프로토콜을 모델링하고 이를 검증하는 도구로 사용된다. 이들은 하드웨어에서 직접 동작하는 이진 코드를 직접 시뮬레이션 하는 것이 아닌 네트워크 프로토콜 모델을 검증하기 위한 것으로, 센서네트워크 코드를 직접 검증하고 디버깅 할 수 없다. 대표적으로 NS-2[12], SensorSim[13], Glo-

MoSim[14], QualNet[15]등이 있다.

운영체제 기반 시뮬레이터는 운영체제에서 시뮬레이터용 코드를 만들어낼 수 있도록 지원한다. 사용자가 특정 운영체제를 기반으로 컴파일 할 때 센서노드용 이진 코드가 아닌 호스트에서 시뮬레이션이 가능한 코드로 만들어주고, 이를 시뮬레이션 할 수 있도록 한다. 대표적으로 TinyOS에서 제공하는 개발플랫폼 패키지인 TOSSIM[16]이 있다. 센서노드의 상태변화만을 시뮬레이션 함으로 속도가 빠른 장점이 있으나, 사이클을 정확하게 시뮬레이션할 수 없고 특정 운영체제에서 만들어진 프로그램만 시뮬레이션 가능하다.

명령어 수준 시뮬레이터는 일종의 에뮬레이터로 명령어(instruction)를 시뮬레이션 한다. 이들은 사이클이 정확한 시뮬레이션을 할 수 있고 다양한 운영체제를 지원할 수 있다 그리고 네트워크 모델이나 시뮬레이션 언어 등의 영향을 받는 것이 아닌 센서노드에서 실제 동작하는 이진 코드를 이용하는 것으로 코드 수준의 디버깅과 검증이 가능하다. 하지만 정확한 기계코드 시뮬레이션에 따른 계산 오버헤드가 크다는 단점이 있다. 그럼에도 불구하고 센서네트워크는 센서노드 간의 패킷 송수신 타이밍이 매우 중요한 시스템으로 최근에는 명령어수준 시뮬레이터를 이용하여 검증하는 것이 일반적이다. 지금까지 연구된 명령어 수준의 시뮬레이터는 ATEMU[17], Avrora[18], AvroraZ[19], PolarLite[20], DiSenS[21], WorldSense[22], MISS[23], NQEM[24], MSPSim[25] 등이 있다.

ATEMU는 최초의 명령어 수준의 센서네트워크 시뮬레이터로 Atemu128, CC1000으로 구성된 Mica2를 기반으로 한다. 시스템 클락 간격으로 가상 센서 노드들에 적재된 실행이미지를 순차적으로 시뮬레이션 하기 때문에 시뮬레이션에 따른 오버헤드가 크고 센서노드 128개 이상의 시뮬레이션은 힘들다는 단점이 있다.

Avrora도 Mica2을 기반으로 구현된 명령어 수준의 시뮬레이터로 Atemu와 달리 멀티스레딩 방식을 적용하였다. 가상 노드들은 각각 스레드가 할당되어 동작되므로 멀티코어를 사용할 경우 속도가 향상된다. 에너지소모 추정, 인터럽트 추적, 컨트롤플로우(control flow) 등 다양한 기능을 제공하며 오픈 소스로 현재 가장 많이 이용되고 있는 센서네트워크 시뮬레이터이다. 이를 확장하여 Atmega128, CC2420으로 구성된 MICAz를 시뮬레이션할 수 있도록 만든 것이 AvroraZ이다. 하지만 Avrora 계열의 시뮬레이터들은 아직까지는 MSP430을 지원하지 못한다. 따라서 Telos형 센서노드는 시뮬레이션할 수 없다.

국내에서는 MISS가 Atmega128, CC2420으로 구성된 센서노드를 지원하는 최초의 명령어 수준의 시뮬레이터로 이산 사건 시뮬레이션(discrete-event simulation) 방식을 이용한다.

NQEM(Nano Qplus EMulator)은 우리의 이전연구결과로 MISS를 기반으로 동일한 전략을 통해 개발한 NanoQplus[26]를 위한 MICAz 기반 시뮬레이터이다. MISS에 비해 안정성을 크게 높였으며 하드웨어를 보다 정밀하게 개선하였다. 특히 무선라디오 모듈인 CC2420 모델의 구조를 새롭게 바꿔 더욱 정밀하게 구현하였다. NQEM은 국내의 센서네트워크 운영체제인 NanoQplus 최신버전에 최적화 되어있다. 즉 NanoQplus를 위한 통합개발도구로서 개발자의 코드 개발을 위한 코드에디터, 컴파일러, 디버거를 연동시켜 준다. 그러므로 코드 개발 중 시뮬레이터를 이용해 연동된 디버거를 통해 C코드와 어셈블리 코드상에서의 디버깅할 수 있으며, 다양한 패킷의 송수신 타이밍도 확인할 수 있다. 게다가 IPEN[27]으로 확장할 경우 프로그램 함수단위의 평균 95% 정도의 정확도를 갖는 전력소모량을 예측할 수 있다. 하지만 이 역시 MSP430을 지원하지 못하여 Telos 형 센서노드는 시뮬레이션 할 수 없다.

지금까지 설명했던 명령어 수준의 시뮬레이터들은 Atmega128을 이용하는 것이었으나 이와는 달리 WorldSens는 MSP430를 지원하는 시뮬레이터다. 하지만 라디오 모듈은 CC1100[28]으로 구성된 센서노드를 지원하기 때문에 MSP430과 CC2420으로 구성된 Telos는 시뮬레이션할 수 없다.

MSPSim은 MSP430용 에뮬레이터로 Java로 작성되었다. 중앙처리장치인 MSP430 만을 위한 에뮬레이터이다. 라디오 모듈을 포함한 주변 장치와의 연결은 인터페이스 부분만을 확장할 수 있도록 하였지만 실제로 CC2420 모델이 존재하지 않는다. 하지만 MSPSim의 확장인 CooJa/MSPSim의 경우에는 ContiKi OS 용 시뮬레이터인 CooJa를 MSPSim과 통합하여 앞서 설명했던 네트워크, 운영체제, 명령어 기반 시뮬레이션 모두를 지원하는 새로운 개념인 크로스-레벨(cross-level) 시뮬레이터이다. 네트워크 시뮬레이션의 경우 NS-2와 비슷한 방법으로 네트워크 알고리즘을 정의하고 이를 시뮬레이션 하며 운영체제 시뮬레이션의 경우 TinyOS의 TOSSIM과 동일한 전략을 사용한다. 마지막으로 명령어의 수준의 시뮬레이션의 경우 중앙처리장치 시뮬레이션으로는 MSPSim을 기본 에뮬레이터로 이용하고, 라디오 모델에서는 TOSSIM 과 비슷한 전략으로 비트수준(bit level) 시뮬레이션을 한다. 특히 무선라디오 모듈인 CC2420의 시뮬레이션에서는 센서네트워크 운영체제인 ContiKi와 TinyOS에서 필요한 기능, FIFO, FIFOP, SFD 인터럽트 캡쳐 등과 같은 기능을 특정적으로 TOSSIM 에서와 비슷한 비트수준 시뮬레이션 전략을

이용한 것으로 하드웨어 수준의 모델을 이용한 것이 아니다. 따라서 운영체제에서 필요한 기능만을 시뮬레이션 하는 것이라 볼 수 있으며 CC2420의 하드웨어의 내부 동작 자체를 정확히 모델링한 것은 아니다.

TeloSIM은 MSP430과 CC2420로 구성되어 있는 Telos 센서노드를 기반으로 한다. TeloSIM은 NQEM과 동일한 전략을 이용하여 이산-사건 시뮬레이션 방법을 사용하였으며, 기능 모델인 아닌 MSP430, CC2420의 정밀한 하드웨어 내부 모델을 갖고 있으므로 운영체제의 특정기능과는 무관하게 하드웨어 동작과 동일한 시뮬레이션이 가능하다. 또한 GUI를 따로 지원하여 시뮬레이션의 편의를 제공할 수 있다.

3. 시스템 구성

시스템 구조는 그림 1과 같다. 가상의 센서노드들은 MSP430 모델, CC2420 모델을 각각 갖고 있으며 시뮬레이션을 위한 이벤트처리기를 통해 이산사건 시뮬레이션 엔진에서 시뮬레이션을 처리한다. 사용자는 커맨드 창을 이용하여 실제 센서노드에서 동작하는 실행코드를 직접 선택하여 가상의 센서노드에 적재할 수 있으며, 실행 파일이 적재된 후 시작 명령을 통해 시뮬레이션 된다. GUI 도구는 커맨드 환경에서 명령을 통해 TCP/IP를 통해 연결된다. 따라서 사용자는 GUI를 이용하여 가상의 센서노드를 설정하거나 시뮬레이션을 위한 명령, 위치변경 등이 가능하며, 시뮬레이션 과정, 결과를 볼 수 있다.

3.1 시뮬레이션 엔진

시뮬레이션은 이산 사건 시뮬레이션방식을 사용한다. 이 방법은 시뮬레이션 결과를 얻기 위해서 시뮬레이션의 시작부터 끝까지 모든 시간들에 대해서 시뮬레이션을 할 필요 없이 발생할 수 있는 이벤트들을 추상화하고 각 이벤트의 발생시간이나 주기를 표본화(sampling)하고 이 값을 이용하여 시뮬레이션을 하는 방법이다. 따라서 이산 사건 시뮬레이션 방법은 시간의 흐름이 시스템의 상태 변화에 의한 이벤트 발생으로 이루어지는 시뮬레이션이므로 특히 대부분의 시간을 저전력모드(sleep mode)로 유지하고 있는 센서네트워크의 경우 일반적인 명령어 수준의 중앙처리장치 시뮬레이션이나 임베디드 보드 시뮬레이션에 비해 명령어 처리 이벤트 발생 빈도가 작아 고속의 시뮬레이션이 가능하다. 따라서 센서네트워크에 잘 맞는 시뮬레이션 기법이다. 실제 이전연구 결과의 대부분의 명령어 수준 센서네트워크 시뮬레이터에서는 이 방법을 사용한다.

3.2 Telos 센서노드

TeloSIM이 기반으로 하는 Telos 센서노드는 최근에 사용되고 있는 센서노드용 중앙처리장치 중 가장 적은

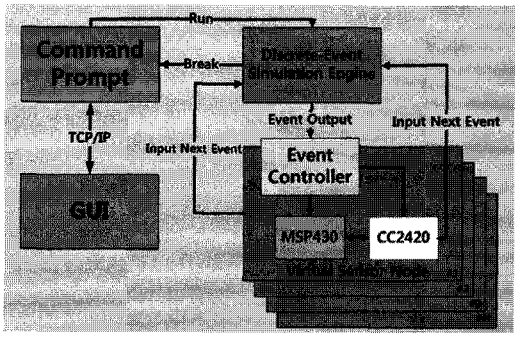

그림 1 시스템 구조

전력을 소모하는 MSP430과 Zigbee를 지원하는 무선 라이오 모듈인 CC2420으로 구성된 센서노드이다. MSP430은 8MHz로 동작하는 16비트 RISC 구조로 27개의 기본 명령어, 7개의 주소지정모드, 16개의 레지스터를 갖고 있으며 특히 슬립모드 지원을 위한 절전기능을 갖고 있다. 또한 DCO(Digitally Controlled Oscillator)에 의한 클락 과 메모리맵형(Memory mapped) 형태의 구조, 하드웨어 곱셈기, 직접메모리접근(DMA: Direct memory access), 아날로그-디지털변환회로, 디지털-아날로그 변환회로, 비교측정기(Comparator), 타이머, I/O포트, USART/SPI/I2C 등을 지원한다. CC2420은 2.4GHz ZigBee를 하드웨어적으로 지원하는 저전력의 무선라디오로서 센서 네트워크와 같은 소형의 시스템에서 널리 쓰이는 대표적인 하드웨어이다. 2440~2483.5 MHz내에서 최대 250kbps의 데이터 전송률을 가지고 동작 한다. 또한 데이터의 송신 및 수신을 위해서 128 바이트의 송신 데이터 FIFO와 128 바이트의 수신 데이터 FIFO를 가지고 있다. 또한 CC2420은 SPI 인터페이스를 통해서 128 바이트의 송수신 FIFO 혹은, CC2420의 제어에 관련된 레지스터나 상태 레지스터 등에 쉽게 접근할 수 있다는 특징을 가지고 있다. 그림 1에서 보는 바와 같이 TeloSIM에서는 MSP430과 CC2420의 정밀한 하드웨어 모델을 갖고 있으며 시뮬레이션 엔진을 통해 시뮬레이션 된다.

4. 시뮬레이터 구현

4.1 시뮬레이터 엔진

그림 2는 TeloSIM 에서 다수의 센서노드들이 명령어를 실행하고 시뮬레이션 모델에 따라 시뮬레이션 되는 과정을 보여준다. 각각의 가상의 센서노드들은 기본적으로 그림 3과 같은 순서로 시뮬레이션 된다. TeloSIM은 명령어 기반의 일종의 에뮬레이터로 그림 2와 같은 순서로 스케줄링 된다. 모든 동작의 기본은 명령어에서 시작되며 명령어를 통해 인터럽트, 타이머와 같은 각종 내부 I/O 장치모델 또는 센서나, 라디오 모델등과 같은

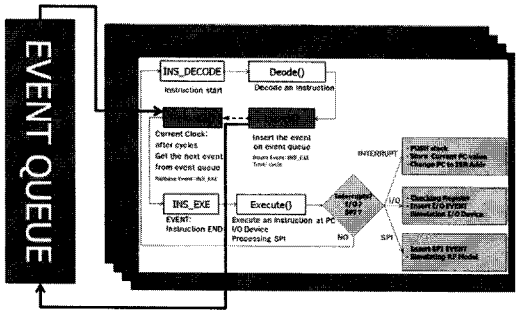

그림 2 시뮬레이션 동작

외부 장치 모델들이 동작하고 이 모델들 역시 이벤트 큐에 시뮬레이션 시간과 이벤트를 기록하고 모델링 된 시간에 시뮬레이션이 진행되는 방식이다. 각 명령어는 실행에 따른 사이클의 소모로 기본 시뮬레이션 시간이 기록되며 특정 명령어로 인해 주변장치가 동작될 경우, 주변 장치의 상태에 따라 시뮬레이션 시간이 모델링 된다. 가상의 센서노드들은 0~50ms 안에서 무작위로 서로 다른 시작 시간을 갖는다. 각 센서노드의 첫 번째 명령어 이벤트를 이벤트 큐에 넣어주면 그림 2의 과정을 반복하여 동작하며 모든 센서노드는 이벤트 큐를 공유함으로 시뮬레이션 순서에 있어 논리적으로 동기화된다. 무선라디오 모듈인 CC2420의 경우 중앙처리장치 모델과는 별도로 MSP430과 연결된 외부 I/O 핀과 연결되어 있으며 UART가 SPI와 같이 동작되어 전송된 명령에 의해 시뮬레이션 되며 이때 역시 공유된 이벤트 큐를 이용하여 각각의 가상의 센서노드들의 CC2420 모델이 동기화되어 시뮬레이션 된다.

4.2 MSP430모델링

MSP430의 모델은 명령어 처리를 기본으로 하여 센서 노드 하드웨어의 동작과 동일하게 동작한다. 그림 3은 일반적으로 명령어 수준 시뮬레이션 모델의 동작과정이다. Fetch, Decode, Execute 과정과 같이 하드웨어 동작 그대로 처리하며, 시뮬레이션 순서와 시간의 진행 과정은 위의 과정을 추상화하여 만든 이벤트를 이벤트 처리기를 통해 이산 사건 시뮬레이션 엔진에서 시뮬레이션을 진행한다. 단 본 시뮬레이터에서는 Fetch 과정과 Decode 과정은 둘을 따로 분리하지 않고 동일한 이벤트로 Fetch, Decode 이벤트 처리 후 Execute 과정이 실행된다. 그림 2와 그림 3은 시뮬레이션 과정을 보여준다. Decode 이벤트가 발생하며, Decode 과정 후 다음 이벤트인 Execute를 이벤트 큐에 적재하고 이벤트 큐에서 해당 이벤트가 방출되면 프로그램카운터(PC: program Counter) 값을 증가시키고 해당 명령어를 실행한다. 인터럽트검사 후, 인터럽트가 없었을 경우 다음 Decode 이벤트를 이벤트 큐에 저장한다. 일반적인 명령어 시뮬

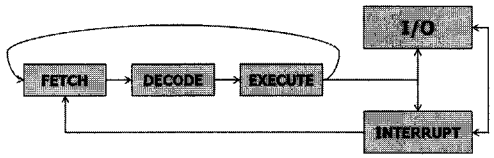

그림 3 명령어 수준 시뮬레이션(Instruction Set Simulation: ISS) 동작과정

레이션은 위의 순서를 따르게 된다. 하지만 입출력(I/O) 장치나 SPI 통신 이벤트 등을 제어하는 레지스터가 변경될 경우에는 해당 장치를 시뮬레이션하기 위한 함수들이 호출되며 해당 장치의 동작 이벤트들을 이벤트 큐에 적재하여 시뮬레이션이 진행된다. 만약 인터럽트가 발생되어야 할 경우 프로그램 카운터 값을 해당 인터럽트 서비스 루틴의 주소로 바꾸고, 실제 하드웨어 동작과 동일하게 인터럽트를 처리한다.

MSP430의 주소지정모드는 소스 부 오퍼랜드(operand)에 사용되는 7개의 주소지정모드와 목적 부 오퍼랜드에 사용되는 4개의 주소지정 모드로 이루어져 있다. 본 연구에서는 MSP430의 7개의 주소지정 모드, 목적 부 오퍼랜드에 사용되는 4개의 주소지정 모드를 분석하여 그림 3의 과정을 구현함은 물론 각 명령어 실행에 필요한 사이클을 정확히 계산하여 동일한 동작을 하도록 하였다.

MSP430은 DCO에 의해 다양한 속도의 시스템 클락을 제공받으므로 본 연구에서는 다양한 클락 소스를 구현하였다. DCO는 자체적으로 총 64단계의 클락을 발생시킬 수 있으며 내부/외부 resistor 사용 여부에 따라 다시 클락 속도가 결정되기 때문에 DCO 만으로도 최대 128단계의 클락을 사용할 수 있다. 여기에 외부 크리스털까지 사용될 수 있기 때문에 소프트웨어적으로 다양한 클락 소스를 선택할 수 있는 강점이 있다. 클락 소스의 선택은 저 전력 구현에 있어 필수적인데, 이러한 장점을 시뮬레이터에서 동일하게 구현 하였다. 다만, 온도 및 여러 상황에 따라 클락의 변화가 생기기 때문에 데이터시트에 명시된 비율에 따라 평균적인 값을 클락 속도로 사용하였다.

MSP430에서는 3개의 타이머가 사용되고 있다. 감시계기(Watchdog) 타이머의 경우 예기치 않은 시스템 오류가 발생하여 정상적인 작동이 불가능할 때 리셋(reset)을 통해 빠른 시간 내에 회복할 수 있는 기법을 제공하는 타이머로서 TeloSIM 에서는 감시계기 타이머도 하드웨어와 동일하게 구현되어있다. 또한 일반적인 프로그램 수행 및 저전력 모드에 있어 필수적인 타이머 A, B 또한 구현되어있다. 타이머가 제공하는 up, continuous, up/down 모드를 모두 구현하였고, 타이머의 다중 capture/compare 레지스터와 이것의 인터럽트 구현

을 위해 사용되는 TAIV, TBIV등 타이머 인터럽트 벡터 또한 하드웨어 동작과 동일하게 구현하여 타이머 관련 소프트웨어가 동작할 수 있도록 하였다.

인터럽트는 센서 네트워크의 동작에 있어 핵심적인 역할을 수행한다. 대부분의 시간을 저 전력 모드에서 대기 상태로 머물다가 외부의 신호에 의해 인터럽트가 발생되고 이에 따라 원하는 기능을 수행하는 메커니즘이 사용되므로 인터럽트의 처리는 무엇보다도 중요하다. 이러한 중요성에 따라 필수적인 인터럽트를 구현하였고 데이터시트에 명시된 하드웨어 특성을 그대로 구현하였다.

4.3 CC2420모델링

본 연구의 CC2420 모델링은 기존연구에서 한 단계 개선시킨 NQEM의 정밀화된 CC2420 모델링 방법과 동일한 전략으로 개발하였다. CC2420 모델은 중앙처리장치인 MSP430 모델과는 별도로 동작하는 모델로 MSP430으로부터 명령을 받고 실행되며, 스스로 다음 이벤트가 발생할 상황을 정의하고 별도의 이벤트 수행 시간을 지니게 된다. Zigbee 프로토콜을 지원하기 위해서는 정밀한 이벤트 타이밍 계산이 필요하며 우리의 CC2420 모델은 이를 정확히 구현하였다.

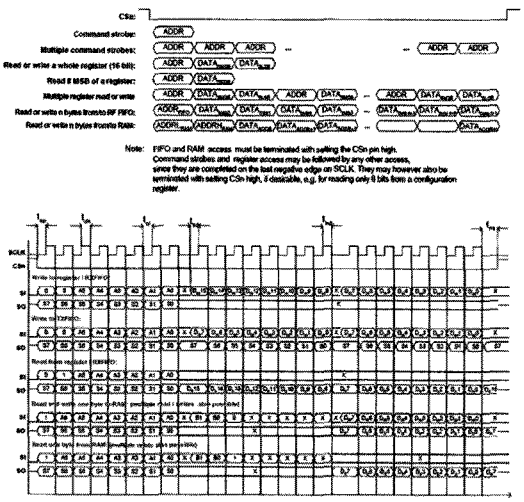

그림 4 CC2420 명령어 종류와 명령처리 타이밍[6]

CC2420은 MSP430과 SPI 통신으로 제어된다. 하지만 MSP430에는 SPI 모듈이 존재하지 않으며 UART를 SPI 처럼 동작시켜 사용한다. SPI에서 사용되는 MOSI, MISO, SCK, SS 등과 같은 4개의 통신 선이 MSP430의 I/O포트에 연결되어 있다. 그리고 두 모듈간 통신에서 MSP430이 마스터(master)가 되고 CC2420은 슬레이브(slave)가 된다. 마스터 쪽에서 클락을 공급하므로 MSP430 쪽에서 클락을 공급한다. MSP430은 UART의 송신버퍼(TxBuffer)를 통하여 명령어와 데이터를 CC2420 쪽으로 전송하고, CC2420은 해당 명령어에 따라 동작된다. CC2420은 다른 센서노드로 부터 받은 데이터를 MCU 쪽으로 전송할 수 있으며 이때 데이터는 UART의 수신버퍼(RxBuffer)로 전송한다.

MSP430과 CC2420은 데이터 송수신 타이밍과 데이터 송수신 알림 인터럽트 등을 지원하기 위한 FIFO, FIFOP 그리고 SFD, CCA 제어신호를 갖고 있다. 그리고 MSP430으로부터의 CC2420으로 전송하는 데이터가 명령어인지 데이터인지를 구분하기 위한 Csn 신호가 있다. 우리는 TeloSIM에 모든 신호 타이밍을 정확히 구현하였다. 또한 MSP430과 CC2420이 연결된 각 외부신호들을 별도의 인터럽트 서비스 루틴을 가질 수 있으며, 신호의 방향, 인터럽트 플래그, 인터럽트 엣지, 인터럽트 허용 등을 선택할 수 있고 TeloSIM은 이 모든 것을 지원한다. TeloSIM은 위에 설명한 모든 동작과정이 구현되어 있으며 실제 CC2420 하드웨어를 제어하는 것과 동일한 한 과정으로 제어할 수 있다.

그림 4는 CC2420의 명령어에 따른 동작방식과 타이밍을 보여주는 그림이다. 이 명령어들을 크게 두 가지로 구분할 수 있다. 첫 번째는 상태변화 즉 CC2420의 특정 동작을 요구하기 위한 Command Strobe이고 두 번째는 CC2420의 레지스터 TX_FIFO, RX_FIFO나 RAM 영역의 엑세스하기 위한 명령어이다. Command Strobe와 레지스터의 경우 데이터 크기가 일정하게 정해져 있기 때문에 그림 4의 첫 번째에서 보듯 Csn을 사용하지 않고 연속적으로 명령어와 데이터를 받을 수 있다. 하지만 FIFO나 RAM의 경우 FIFO나 RAM에 접근하기 위한 명령어 이후에 송수신 되는 데이터의 크기가 정해져 있지 않기 때문에 MSP430에 외부 신호로 연결되어 있는 P4.2의 Csn을 사용해야 한다. FIFO와 RAM은 데이터 수신이 종료하면 반드시 Csn 신호를 세트 함으로써 MSP430으로부터의 데이터 송신이 종료되었음을 알려야 한다. MSP430에서 이 신호를 이용하여 CC2420 제어가 가능하다. Command strobe를 제외한 나머지 명령어들은 최상위 2비트를 구분하여 결정할 수 있으며 시뮬레이터에서도 이를 그대로 반영하여 최상위 비트에 따라 분기되도록 구현하였다. 그리고 UART의 의한 송수신 속도를 레지스터에 따라 구현하여, 송수신 시 속도에 따라 이벤트 시간을 지정하여 시뮬레이션 되도록 하였다.

MSP430과 CC2420은 UART를 SPI와 같이 이용하여 서로 통신하도록 구성되어 있다. TeloSIM에서 MSP430이 CC2420을 제어하는 시뮬레이션 과정은 다음과 같다. MSP430 이 송신버퍼를 이용하여 CC2420에 데이터가 전달되도록 이벤트를 발생시킨다. 이벤트가 이벤트처리기와 이산사건 시뮬레이션 엔진을 통해 이벤트

큐에서 방출되면서 시뮬레이션 시간이 흐르고, CC2420 모델의 수신부가 이벤트에 의해 호출된다. CC2420 모델은 MSP430으로부터 전송 받은 데이터를 확인하고 명령어인지 데이터인지 외부신호에 따라 구분하고 그에 따라 CC2420을 동작시킨다. 이때 명령어에 의한 동작이나 데이터 구분뿐 만이 아니라 CC2420의 레지스터 설정, 신호변경, Zigbee 프로토콜 설정 등을 할 수 있으며 TeloSIM은 앞서 밝힌 모든 기능을 수행할 수 있다. 송수신에 따른 내부 상태 변화 과정은 다음과 같다. CC2420 모델은 송신을 위해 걸리는 시간을 표본화한 후 다른 노드들이 수신할 시간을 설정한다. 다른 센서노드들에게 수신이벤트가 발생할 수 있도록 이벤트 큐에 이벤트를 적재한다. CC2420의 수신이벤트를 받는 센서노드들은 CC2420의 Zigbee 프로토콜 모듈을 거쳐 수신버퍼에 저장하거나 무시한다. 만약 수신버퍼에 저장되면 CC2420의 내부 레지스터 설정상태와 RAM, 송수신 FIFO에 따라 MSP430에 FIFO, FIFOP 인터럽트 신호를 설정한다. 또한 데이터 송수신에 과정에서 TeloSIM의 CC2420 모델은 SFD, CCA의 기능 역시 충실하게 수행할 수 있다. 만약 송신 전 채널이 깨끗하지 못하다면 다시 시도하기 전 짧은 시간 동안 라디오를 백 오프(back off)한다. 일반적으로 TinyOS의 경우 CCA 핀이 하이상태라면 패킷을 전송하고, 채널이 사용 중이라면 CCA에 일반적인 기본 동작을 계속해서 수행한다. 따라서 TeloSIM에서는 CCA의 기능을 충실히 지원하기 위해 CCA 동작 모델을 만들고 송수신 상태변화에 따라 CCA 핀 값을 변경해 줌으로서 CCA 기능을 구현하였다. 그리고 이때 핀 값을 MSP430에 정확한 타이밍으로 신호를 전달하고, MSP430은 레지스터 설정에 따라 경우에 따라서는 인터럽트를 발생시킨다. 그리고 패킷의 목적지인 자신이 패킷을 성공적으로 받은 경우 CC24240의 레지스터 설정을 확인하고 해당 패킷이 응답(ACK) 메시지를 필요로 한다면, 응답 메시지를 송신하도록 구현하였다. TeloSIM은 CC2420 모델에서 실제 하드웨어 동작과 거의 유사하게 구현되어 있다.

4.4 GUI 구현

TeloSIM은 커맨드 창에서뿐만이 아니라 GUI 도구를 사용하여 가상 센서 네트워크의 구성 및 시뮬레이션을 쉽게 조작할 수 있다. 또한 GUI를 사용하면 복잡한 센서 네트워크의 구성 및 시뮬레이션 결과를 직관적으로 파악할 수 있게 되므로 사용자의 편의성을 높였다.

GUI 도구는 자바 GUI 빌더인 Jigloo[29]를 이용하여 개발하였다. Jigloo는 CloudGarden에서 만든 스윙과 SWT GUI를 개발할 수 있도록 지원하며, 이클립스 환경에서 개발할 수 있도록 플러그 인으로 제공하고 있다.

TeloSIM의 GUI 도구는 TelosB(Tmote Sky)[30] 이미지를 이용하였으며 보드 이미지와 별도의 LED 이미지를 사용하여 시뮬레이터로부터 TCP/IP를 통해 전송되어오는 메시지를 적절히 처리하여 실제 하드웨어와 동일하게 동작 하도록 하였다. 시뮬레이션 엔진으로부터 전송되어온 메시지를 버퍼에 저장하여 그 내역을 볼 수 있도록 하였으며 간단한 결과 기능을 제공하여 실행 완료된 노드의 시뮬레이션 정보를 볼 수 있도록 하였다. 또한 GUI 내부에서 스레드(Thread)를 사용하여 여러 개의 노드를 동시에 실행할 경우에도 원활하게 작동하도록 설계하였으며, 호스트(Host) PC의 과부하를 막기 위해 패킷 수신 부를 제외한 스레드는 일정시간 슬립(sleep)하도록 하여 GUI 동작에 따른 CPU 점유율을 낮추었다.

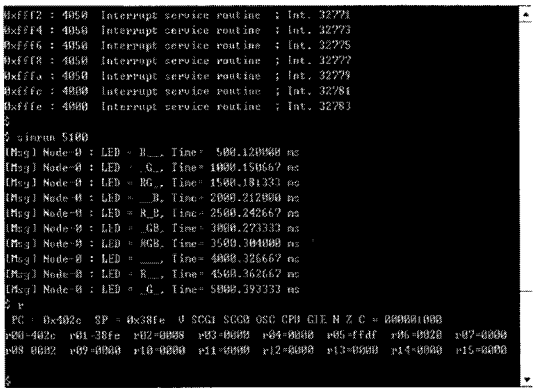

그림 5 카운트 값 점멸 시뮬레이션 결과 화면

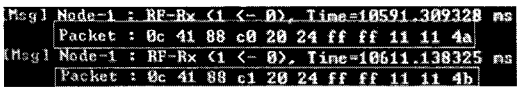

그림 6 CountSend/Receive 송수신 시뮬레이션 화면

```
typedef struct TOS_Msg {

  uint8_t length;
  uint8_t fcfhi;
  uint8_t fcflo;
  uint8_t dsn;
  uint16_t destpan;
  uint16_t addr;
  uint8_t type;
  uint8_t group;
  int8_t data[28];

  uint8_t strength;
  uint8_t lqi;
  bool crc;
  bool ack;
  uint32_t time;
} __attribute((packed)) TOS_Msg;
```

그림 7 TOS Msg - TinyOS Boomrang

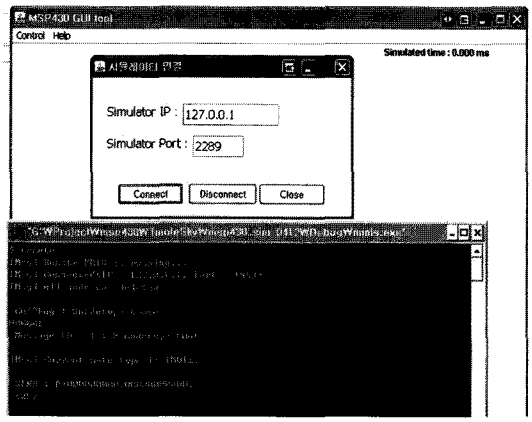

그림 8 GUI 연동

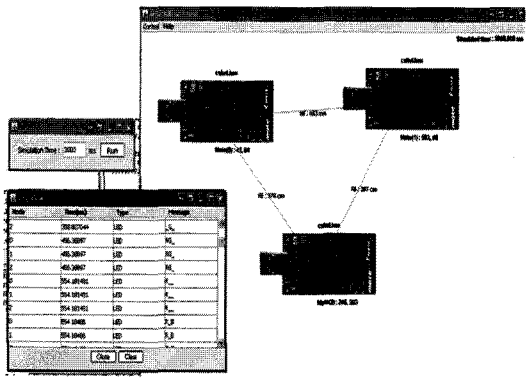

그림 9 GUI를 이용한 시뮬레이션 실행

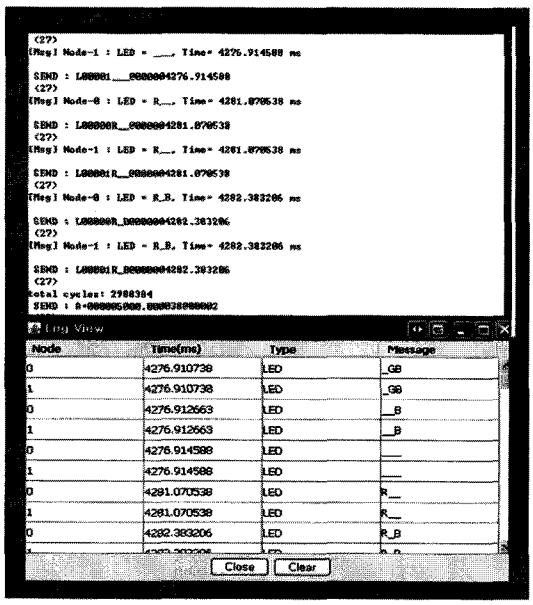

그림 10 Log View를 이용한 결과 확인

5. 실행 결과

TeloSIM은 윈도우환경에서 동작되며 약 3만5천 라인의 코드로 구성된다. 약 800KB의 시뮬레이션 엔진과 약 42KB의 GUI를 각각 실행시켜 서로 연동하여 동작된다. 실행결과를 위해 Intel Core 2 Duo 3.0Ghz, 3G RAM를 사용했다. 시뮬레이션을 하기 위해서는 MSP430을 기반으로 컴파일한 .rom, .srec과 같은 확장자를 갖는 형식의 파일이 필요하다.

TeloSIM은 MSP430 모델과 CC2420 모델이 결합되어 동작되므로 MSP430 모델과 CC2420 모델의 검증을 위해 라디오 모듈을 쓰지 않는 어플리케이션과 라디오를 쓰는 어플리케이션을 이용했으며, 결과비교를 위해 Tmote Sky 센서노드를 사용했다. 그리고 Tmote Sky의 제작회사인 Motive에서 기본으로 제공하는 어플리케이션을 주로 이용하였다.

그림 5는 TeloSIM을 사용하여 TinyOS로 작성된 프로그램을 크로스 컴파일 하여 생성된 .srec 형식의 파일을 실행한 결과이다. 1초에 2번 카운트 값으로 LED를 점멸하는 어플리케이션으로 LED 점멸 정보가 1초에 두 번 카운트 값을 화면에 출력 하는 것을 확인할 수 있다. GUI를 이용 시 텍스트환경이 아닌 실제 센서노드와 동일한 결과를 볼 수 있다. 또한 내부 레지스터 현재 값 정보 확인이 가능하므로, 디버깅 시 사용이 가능하다.

그림 6은 라디오 모듈을 사용하는 어플리케이션이다. 결과 비교를 위해 Tmote Sky를 만든 Motive의 TinyOS Boomerang 버전[31]에서 Count라는 어플리케이션의 이용했다. Count는 Count 값을 브로드캐스트로 주변 노드에게 전달하는 기본 기능을 검증하는 어플리케이션으로 가장 최근 발표된 센서네트워크 시뮬레이터 논문인 PolarLite의 검증실험을 위해 사용될 정도로 잘 알려진 어플리케이션이다. Boomerang 버전에서 Tos msg 형태는 그림 7과 같으며 실제 패킷을 그림 7의 형태에 맞게 보내게 된다. TeloSIM에서 Count 어플리케이션을 두 개의 가상의 센서노드에 각각 적재하고, 이를 시뮬레이션 하면서 패킷을 캡쳐했다. 그림 6에서 보는 바와 같이 수신 받은 패킷과 그림 7을 비교한 결과 정확히 일치하는 것을 확인할 수 있었다. 단 RSSI의 경우 TeloSIM은 RSSI 기능을 제공하지 않으므로 따로 표시하지 않으며 CRC 값은 CC2420 모델 내부에서 사용된다.

그림 8과 같이 GUI를 이용하여 시뮬레이션 엔진을 연동하기 위해서는 그림에서와 같이 -remote 옵션을 주거나 remote 명령어를 실행하여 원격모드로 전환하여 네트워크간 연결을 한다. GUI 도구의 제어메뉴에는 Connect, Add, Lod, Init, Run, Log View, Exit 가 있는데 각기 메뉴명칭이 의미하는 기능을 수행한다. 시뮬

레이터가 원격모드로 실행되고 있을 때 그림 8과 같이 Connect 메뉴를 통해 상호간 접속이 성립되며 이후 시뮬레이션 동작에 필요한 명령어의 전송순서는 <노드생성 - 실행 이미지 적재 - 초기화 - 시뮬레이션 시작>와 같이 동일하다. 그림 9는 GUI와 연동하여 Motive의 TinyOS Boomerang 버전의 CountLed프로그램을 동작시킨 모습이다. 각 노드별로 무선통신의 거리도 확인할 수 있으며, 시뮬레이션 시간과 시뮬레이션 엔진으로부터 전송되어 오는 메시지를 분석하여 노드번호, 시간, 타입, 메시지 내용 등을 출력한다. 그리고 그림 10과 같이 Log View 창을 통해 시뮬레이션 결과를 확인할 수 있다.

6. 결론 및 향후 계획

본 연구의 결과물인 TeloSIM은 MSP430, CC2420을 사용하는 Telos 형 센서노드를 위한 명령어 수준 시뮬레이터로 국내 최초로 개발되었으며 MSP430과 CC2420을 기존의 시뮬레이터들보다 정밀하게 모델링하여 시뮬레이션의 정확도를 높인 것이 장점이다.

센서 네트워크의 특성상 설치 후, 사람이 직접 초소형의 센서 노드들을 일일이 관리할 수 없기 때문에 센서 노드를 직접 설치하기 이전에 시뮬레이션을 통해, 각 센서 노들의 네트워크 환경을 미리 확인하고 점검하는 작업은 매우 중요하다. 따라서 본 연구 결과물인 TeloSIM은 하드웨어 수준의 정교한 에뮬레이션이 가능함으로 응용프로그램 개발자나 센서 네트워크를 연구하는 연구자 들에게 실제로 센서 노드에 프로그램을 적재하고 설치하지 않아도, 연구 또는 검증할 수 있는 도구로 활용이 가능할 것이다.

하지만 아직 초기 버전이므로 다수의 버그를 갖고 있어 점차 개선하고 있는 상황이다. 그리고 인터럽트 추적, 컨트롤플로우(control flow) 등 다양한 기능 제공하지 못하고 있으며, 특히 IPEN과 같은 정밀한 하드웨어 전력모델이 없어 정확한 전력소모가 추정이 불가능하다. 또한 패킷 송수신시 수학적 모델을 사용하지 않아 패킷 에러에 대한 부분은 시뮬레이션 할 수 없다.

따라서 향후 연구 계획으로는 버그를 수정하여 좀 더 안정화될 수 있도록 할 것이며 명령어 수준 시뮬레이션을 통한 다양한 서비스 기능을 구현할 것이다. 또한 MSP430 의 다양한 어드레스 모드에 따른 정밀한 중앙처리장치의 전력모델링, CC2420과 MSP430 간의 통신에 따른 센서보드 전력모델링 등을 통해 정확한 에너지 소모 추정이 가능하도록 할 것이다.

참 고 문 헌

[1] Joseph Polastre, Robert Szewczyk, and David Culler, "Telos: Enabling ultra-low power wireless research," *in The Fourth International Conference on Information Processing in Sensor Networks IPSN 2005*, LA, CA Apr. 2005.

[2] TI MSP430 - http://www.ti.com/

[3] ATmega128L - http://atmel.com/products/AVR

[4] CC1000: Radio Module http://focus.ti.com/docs/prod/folders/print/cc1000.html

[5] Zigbee - http://www.zigbee.org/

[6] CC2420: Radio Module http://focus.ti.com/docs/prod/folders/print/cc2420.html

[7] MICA2 - http://www.xbow.com/products/product_pdf_files/wireless_pdf/mica2_datasheet.pdf

[8] MiCAz http://www.xbow.com/products/product_pdf_files/wireless_pdf/micaz_datasheet.pdf

[9] J. Eriksson, F. Österlind, N. Finne, N. Tsiftes, A. Dunkels,T. Voigt, R. Sauter, and P. J. Marrón. Cooja/mspsim: Interoperability testing for wireless sensor networks. *In Proceedings 2nd International Conference on Simulation Tools and Techniques (SIMUTOOLS'09)*, Rome, Italy, Mar. 2009.

[10] TinyOS - http://tinyos.net

[11] ContiKi - http://www.sics.se/contiki/

[12] NS2 - http://isi.edu/nsnam/ns/

[13] S Park, A Savvides, MB Srivastava, "SensorSim: a simulation framework for sensor networks- *International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems Proceedings of the 3rd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, Boston, Massachusetts, United States pp.104-111.

[14] GloMoSim - http://pcl.cs.ucla.edu/projects/glomosim/academic/license.html

[15] QualNet - http://www.scalable-networks.com

[16] P. Levis, et. al., "TOSSIM: Accurate and scalable simulation of entire TinyOS applications," *In Proceedings of the First ACM Conference on Embedded Networked Sensor Systems*, Nov. 2003.

[17] Manish Karir, et. al., "ATEMU: A Fine-grained Sensor Network Simulator," *Proceedings of First IEEE International Conference on Sensor and Ad Hoc Communication Networks*, Santa Clara, CA, October 2004.

[18] Ben Titzer, et. al., "Avrora: Scalable Sensor Network Simulation with Precise Timing," *Proceedings of the Fourth International Conference on Information Processing in Sensor Networks (IPSN'05)*, LA, CA 2005, pp.477-482.

[19] R.Alberola and D. Pesch, "Avroraz: Extending avrora with an IEEE 802.15.4 compliant radio chip mode," *In 3rd ACM International Workshop on Performance Monitoring, Measurement, and Evaluation of Heterogeneous Wireless and Wired Networks*, Vancouver, Canada, October 2008.

[20] Z. Jin, R. Gupta, "Improving the Speed and Scalability of Distributed Simulations of Sensor

networks," *Proceedings of the 8th International Symposium on Information Processing in Sensor Networks (IPSN'09)*, pp.169-180, San Francisco, CA, 2009. ACM/IEEE.

[21] Y. Wen, R. Wolski, and G. Moore. "Disens: scalable distributed sensor network simulation. *In PPoPP '07: Proceedings of the 12th ACM SIGPLAN symposium on Principles and practice of parallel programming*, pp.24-34, New York, NY, USA, 2007. ACM Press.

[22] A. Fraboulet, et. al., "Deveopment and Prototyping Tools for Application Specific Wireless Sensor Networks," *Proceedings of the 6th International Symposium on Information Processing in Sensor Networks (IPSN'07)*, Cambridge, Massachusetts 2007. ACM/IEEE.

[23] Banh-hyun Kim, et. al., "Development of sensor network simulator for estimating power consumption and execution time," *Korean Journal of Simulation*, vol.15, no.1, 2006, pp.35-42. (in Korean)

[24] Hyunwoo Joe, et. al., "A High-Fidelity Sensor Network Simulator Using Accurate CC2420 Model," *Proceedings of the 8th International Symposium on Information Processing in Sensor Networks (IPSN'09)*, San Francisco, CA, 2009. ACM/IEEE pp.429-430.

[25] Eriksson, Joakim and Dunkels, Adam and Finne, Niclas and Österlind, Fredrik and Voigt, Thiemo (2007), "Mspsim - an extensible simulator for msp430-equipped sensor boards," *In: European Conference on Wireless Sensor Networks (EWSN)*, January 2007, Delft, The Netherlands.

[26] NanoQplus - http:// www.qplus.or.kr

[27] Hyunwoo Joe, et. al., "Instruction-level Power estimator for Sensor Networks," *ETRI Journal*, vol.30, no.1, pp.47-58, Feb 2008.

[28] CC1100 - http://focus.ti.com/docs/prod/folders/print/cc1100.html

[29] Jigloo - http://www.cloudgarden.com/jigloo/

[30] Tmote Sky - http://www.sentilla.com/moteiv-transition.html

[31] Boomerang: low power reliable mesh-networking http://docs.tinyos.net/index.php/Boomerang

조 현 우

2005년 충남대학교 컴퓨터전공(학사). 2007년 충남대학교 컴퓨터공학과(석사). 2008년~현재 충남대학교 컴퓨터공학과(박사과정). 관심분야는 embedded system simulation, energy-aware embedded systems, system modeling

김 형 신

1990년 한국과학기술원(학사). 1991년 Univ. of Surrey, UK.(석사). 1992년~2001년 한국과학기술원 인공위성연구센터, 선임연구원. 2003년 한국과학기술원(박사) 2003년~2004년 Carnegie Mellon University, 박사후연구원. 2004년~현재 충남대학교, 컴퓨터공학과 부교수. 관심분야는 energy-aware embedded system, resource-aware computing, real-time computing, system modeling