논문 2006-01-03

Nano Esto: USN 응용 소프트웨어 개발을 위한 통합개발환경

(Nano Esto: An IDE for USN Application Developers)

정창희, 우덕균, 김용상, 전인걸, 임채덕*
(Changhee Jung, Duk-Kyun Woo, Yongsang Kim, Ingeol Chun, Chaedeok Lim)

Abstract : Integrated development environment (IDE) plays an important role in reducing the developm ent time, thereby improving software development productivity. In recent years, ubiquitous sensor networks (USNs) have become increasingly popular. However their application software is developed mostly using command-line-based tools. Such a development process is not only likely to be error-prone but also inconvenient to debug errors. This paper describes a novel IDE for USN application developers called Nano Esto that provides a single, consistent, and integrated environment for building and executing USN applications. The Nano Esto helps the developers edit and cross-compile source code and download the executable image to the program memory of sensor nodes with just a few button clicks. It also provides support for configuring an application-specific kernel, simulating a wireless sensor network, and estimating power consumption in each sensor node. Currently, Nano Esto runs on Linux as well as on Windows with the same look and feel.

Keywords : Nano Esto,IDE, USN, Eclipse

Ⅰ. 서 론

임베디드 시스템을 위한 응용프로그램 개발 도구로서 통합개발환경을 이용하는 것은 소프트웨어 개발의 생산성과 코드의 완성도를 향상시킬 수 있다는 점에서 매우 중요하게 인식되고 있다[1,2]. 최근에는 USN에 대한 관심이 높아지면서 이를 위한 여러 가지 응용 소프트웨어들이 개발되고 있으나, 이렇다 할 통합개발환경의 부재로 명령어 라인 기반의 개발 방식이 사용되고 있는 실정이다. 이와 같은 방식은 불편함을 야기할 뿐만 아니라 개발 시간을 증가시킬 수 있으며, 궁극적으로 USN 응용 소프트웨어의 개발을 어렵게 만드는 요인이 된다. 본 연구에서는 이와 같은 문제점을 해결하기 위하여 USN 응용 소프트웨어를 빠르고 편리하게 개발할 수 있는 통합개발환경인 Nano Esto를 제안한다. Nano Esto는 소스 편집, 컴파일, 이미지 적재 및 실행 기능과 소스 수준의 오류 정정 기능을 비롯해 USN 응용 소프트웨어의 개발에 필요한 모든 기능들을 제공한다. 또한 자체에 내장된 네트워크 환경의 시뮬레이션 기능을 통해 센서 노드에 적재되는 프로그램의 정확성 및 안정성을 검증할 수 있다. 뿐만 아니라 센서 노드의 자원이 충분치 않음을 고려하여 응용의 성격에 따라서 최적의 커널을 구성할 수 있는 기능을 제공한다.

* 교신저자 (Corresponding Author)
논문접수 : 2006. 5. 16., 채택확정 : 2006. 6. 5.
정창희, 우덕균, 김용상, 전인걸, 임채덕 : 한국전자통신연구원 임베디드 S/W연구단

Ⅱ. Nano Esto를 이용한 USN 응용 소프트웨어 개발 환경

USN을 구성하는 센서 노드는 제한된 메모리와 MCU(Micro Controller Unit) 자원을 갖기 때문에, 응용 프로그램 개발 환경을 직접 구동할 수 없다. 따라서 USN 응용 소프트웨어의 개발은 데스크탑 PC와 같은 고 사양의 호스트 시스템에서 수행되고, 테스팅, 디버깅을 위하여 완성된 이미지를 타겟 센서 노드에 적재하여 실행하는 크로스 개발 환경을 이용한다. 이러한 크로스 개발 방식에서는 통합 개발도구의 부재로 인해 대부분의 개발자들이 자신이 사용하던 개별적인 도구를 이용해 프로그램 편집

및 컴파일, 이미지 적재 및 디버깅을 수행하기 때문에 사용자에게 불편함을 초래할 뿐만 아니라, 소프트웨어 개발의 생산성을 저하를 초래한다.

이러한 문제를 해결하기 위해 본 논문에서는 USN 응용 소프트웨어 개발을 위한 통합환경으로써 Nano Esto를 제안하고자 하며, Nano Esto를 이용한 개발 환경을 <그림 1>에 나타낸다. 본 연구에서는 하드웨어 플랫폼으로 ATMEL사의 AVR MCU를 탑재한 센서 노드를 선정했다[3].

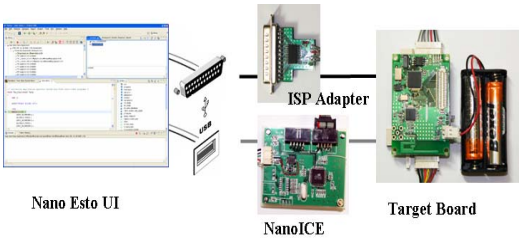

그림 1. Nano Esto의 소프트웨어 개발환경
Fig. 1. Software development environment of Nano Esto

타겟 센서 노드는 이미지 적재를 위한 ISP 인터페이스와 하드웨어 디버깅을 위한 JTAG 포트를 포함하고 있으며, 각각 ISP 어댑터와 NanoICE에 의해 호스트 컴퓨터와 연결된다.

III. Nano Esto의 설계 및 구현

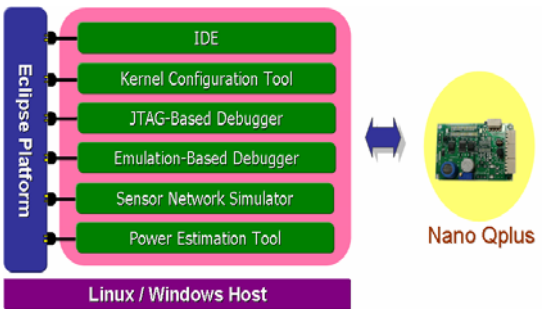

그림 2. Nano Esto 구조도
Fig. 2. Nano Esto architecture

Nano Esto는 <그림 2>와 같은 구조를 가지고 있으며, 각 도구의 인터페이스는 Eclipse 플랫폼[4]에 플러그인 되어 있다. Eclipse 플랫폼은 기본적으로 Java 언어로 구현되어 있기 때문에 운영체제에 독립적인 사용자 인터페이스를 지원한다. 따라서 본 연구의 결과물은 리눅스와 윈도우즈 뿐만 아니라, Java 언어가 실행될 수 있는 모든 환경에서 동일한 사용자 인터페이스를 제공한다.

1. 원격 개발 IDE

원격 개발 IDE는 크로스 개발 환경에서 타겟 시스템에서 수행되는 USN 응용 소프트웨어를 개발하기 위한 목적으로 개발되었으며, 프로젝트 관리자, 소스 편집기, 크로스 컴파일 툴체인과 실행 이미지의 적재를 위한 퓨징 엔진을 포함한다. 소스 편집기와 프로젝트 관리자의 경우 Nano Qplus 응용 소프트웨어의 개발 언어인 C 언어에 특화되어 있으며, Eclipse 플랫폼이 기본적으로 지원하는 텍스트 편집기와 프로젝트 관리자를 확장하여 원격 응용 프로그램 개발에 적합한 기능을 수행하도록 구현하였다.

크로스 컴파일 툴체인은 GNU의 binutils, gcc와 같은 공개 소스를 기반으로 타겟 센서 노드를 구동하는 AVR 코어를 지원하도록 패키징된 도구를 말한다. 크로스 컴파일 툴체인은 C/C++ 소스 프로그램을 컴파일, 링크 할 수 있은 컴파일러, 컴파일된 바이너리 코드를 조작할 수 있는 바이너리 유틸리티들을 포함한다. 이는 Eclipse 플랫폼에서 기본적으로 사용되는 컴파일러를 대체하며, 사용상의 편이를 위해 동일한 인터페이스를 통해 제공된다.

퓨징 엔진은 개발이 완료된 실행 이미지를 ISP 어댑터를 이용하여(<그림1>) 타겟 센서 노드의 프로그램 메모리에 다운로드 하는 기능을 담당하며, 오픈 소스 프로젝트인 avrdude[7]를 기반으로 구현되었다.

2. 커널 설정 도구

일반적으로 USN을 구성하는 타겟 센서 노드의 프로그램 메모리 사이즈는 8~128 KB로 극히 제한적이다. 따라서 운영체제가 지원하는 모든 모듈들을 센서 노드에 적재하는 것은 현실적으로 불가능하며, 개발하는 응용 소프트웨어의 성격에 따라 요구되는 모듈들만으로 커널을 구성하는 것이 필수적이다. 이러한 문제를 해결하기 위해 Nano Esto는 GUI 기반의 커널 설정 도구를 포함한다. 특히 ETRI에서 개발된 USN용 초소형 운영체제인 Nano Qplus를 적용한 타겟 설정도구를 지원하고 있으며, 이는 각 모듈들의 의존성을 자동으로 검사하고 원격 개발 IDE와 연계하여 선택된 모듈에 대한 기본적인 스켈레톤 코드를 생성한다.

3. 디버거

Nano Esto는 JTAG 기반 디버거[6]와 에뮬레이터 기반 디버거를 지원한다. 전자는 응용 프로그램이 직접 타겟 센서 노드에서 동작하는 on-line 환경에서의 디버깅 기능을 담당하며, 후자는 타겟 센서 노드가 없는 off-line 환경에서의 디버깅 기능을 담당한다. 두 디버거는 동일한 사용자 인터페이스를 갖으며, 개발자의 요구에 따라 변수, 레지스터, 메모리, 스택 등의 정보를 제공한다.

3.1 JTAG 기반 디버거

JTAG 기반 디버거는 IEEE1149.1 표준에 정의되어 있는 JTAG 포트를 이용하여 실제 타겟 센서 노드에서 동작하는 Nano Qplus 용용 소프트웨어의 디버깅을 담당한다. JTAG 기반 디버거는 Eclipse 사용자 인터페이스, 디버깅 엔진, NanoICE(<그림 1>)로 구성되며, 그 구조는 다음과 같다.

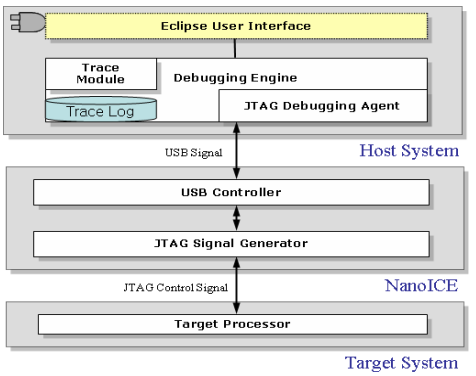

그림 3. JTAG 기반 디버거 구조도
Fig. 3. JTAG-based debugger architecture

- 이클립스 사용자 인터페이스 : 사용자 인터페이스는 개발자에게 편리한 환경을 제공하고 다른 도구와의 통합을 위해 이클립스 플러그인으로 개발되었으며, 이클립스 플랫폼에서 기본적으로 제공하는 디버거와 동일한 사용환경을 제공한다.
- 디버깅엔진 : 디버깅엔진은 GNU GDB를 기본으로 하여 개발되었다. 사용자 인터페이스를 통해 받은 명령을 수행하며, NanoICE를 통해 타겟 시스템을 제어하고 필요한 정보를 가져온다.
- NanoICE : 임베디드 시스템 개발에 이용하는 호스트 컴퓨터에서 타겟 시스템의 MCU를 제어하기 위해 필요한 하드웨어 장비이다. 타겟 시스템의 JTAG 포트와 개발 호스트 컴퓨터를 하드웨어적으로 연결시켜주며, 디버깅엔진에서 받은 타겟 제어 명령어를 하드웨어 시그널로 변환시켜준다. 이 외에도 NanoICE는 디버깅을 위해 디버깅엔진에서 필요로하는 다양한 기능을 제공한다.

3.2 에뮬레이터 기반 디버거

Nano Esto는 타겟 하드웨어가 없는 상황에서도 일반적인 디버깅 기능을 지원하며, 이를 위해 에뮬레이터 기반 디버거를 제공한다. 에뮬레이터 기반 디버거는 내부의 MISS 엔진(3.4장 참조)을 이용하여 실행 이미지를 타겟 센서 노드에 적재하지 않고도 에뮬레이터 콘솔을 통해 디버깅 및 실행 결과를 출력한다. <그림 4>는 에뮬레이터 기반 디버거의 구조도를 나타낸다.

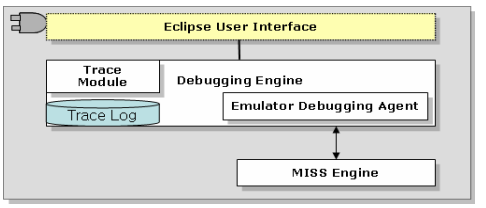

그림 4. 에뮬레이션 기반 디버거 구조도
Fig. 4. Emulation-based debugger arch -itecture

- 이클립스 사용자 인터페이스 : JTAG 디버거와 동일한 사용자 인터페이스를 가지고 있으며 이클립스 플러그인으로 동작한다.
- 에뮬레이션 기반 디버깅 에이전트 : GNU GDB 기반의 디버거에서 에뮬레이터(센서네트워크시뮬레이터인 MISS에 포함되어 구현이 되어있으며 자세한 설명은 4절에서 하도록 하겠다)에게 RSP 신호를 전달하는 역할을 담당한다.

4. 센서 네트워크 시뮬레이터

센서 네트워크 시뮬레이터 (MISS: Machine Instruction-level Sensor network Simulator)는 센서 네트워크 환경에서 센서 노드 개발 이전에 USN 응용 프로그램을 검증할 수 있는 도구로서,기계어 수준의 명령어 시뮬레이션 방법을 사용하여 실제 타겟 센서 노드의 동작을 시뮬레이션한다[5]. 이를 통해 센서 노드에 탑재되는 응용 소프트웨어

및 운영 체제의 동작과 네트워크의 안정성을 검증할 수 있으며, 일례로 센서 노드의 위치 및 통신 환경을 고려한 네트워크 시뮬레이션이 가능하다. 센서 네트워크 시뮬레이터는 <그림 5>와 같이 구성되며, 가상 센서 노드(Virtual Sensor Node)는 노드의 개수만큼 존재한다. 각각의 가상 노드는 실제 센서에서 동작하는 실행 파일(Executable Image)을 입력으로 동작한다. 센서 노드간의 Zigbee 통신은 화살표로 표시되며, 각 센서 노드의 시리얼 데이터는 별도의 뷰를 통해 출력된다. 뿐만 아니라, 센서 하드웨어에 부착된 LED 출력 표시 기능을 통해서 USN 응용 프로그램을 보다 현실적으로 검증할 수 있도록 구현하였다.

MISS Engine Agent(MEA) 모듈은 디버거나 MGA(MISS GUI Agent)로부터 수신된 정보를 해석하여 가상 센서 노드로 전달하고, 가상 센서 노드로부터 발생되는 정보를 디버거나 MGA로 송신하는 기능을 제공한다.

Virtual Network 모듈은 센서 노드들 사이에서 발생하는 RF 통신(Zigbee기반)에 대하여 전송 에러율과 전파 수신 거리를 적용하여 실제 환경에서의 RF 통신과 같은 시뮬레이션을 제공한다.

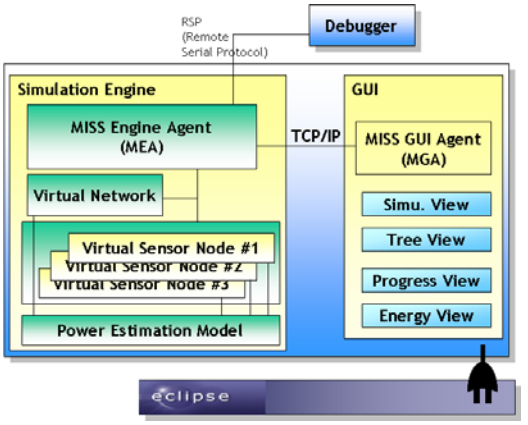

그림 5. 센서 네트워크 시뮬레이터 구조도
Fig. 5. MISS architecture

Virtual Sensor Node 모듈은 타겟 센서 H/W를 S/W로 동일하게 구현한 가상 센서 노드로서 응용 소프트웨어를 메모리에 적재하여 기계어 수준으로 실행한다.

MGA 모듈은 MEA와의 TCP/IP 패킷 교환을 통하여 시뮬레이션에서 발생하는 센서 노드의 상태 정보와 센서 노드 간의 통신상태 등을 User Interface, Shell, 모듈로 전달을 하며, User Interface, Shell, Report 모듈에서 시뮬레이션 엔진으로 발생하는 요청을 MEA로 전달하는 기능을 한다. MGA의 구현은 기존의 도구와의 통합 및 사용자에게 편리한 환경 제공을 위하여 이클립스 플러그인 형태로 구축한다.

5. 전력 분석 도구

Nano Esto의 전력 분석 도구는 기계 명령어 수준 시뮬레이션 기반으로 센서 네트워크 환경의 전력 소모량을 분석하는 도구이다. 본 도구는 센서 네트워크 환경의 각 노드들에 대하여 노드를 구성하는 하드웨어 모듈별, 노드에서 수행되는 프로그램의 함수 단위로 전력 소모량을 분석한다. 시뮬레이션은 앞에서 소개된 센서 네트워크 시뮬레이터를 기반으로 구현되었다.

시뮬레이터와 연동하여 전력 소모량 분석 및 결과 출력은 다음과 같이 진행된다.

1. 시뮬레이터 UI로부터 시뮬레이션 대상 프로그램의 실행 파일 이미지 입력
2. 실행 파일 이미지를 분석하여, 함수 심볼 정보를 추출하여 각 함수의 주소 공간 정보 유지
3. 시뮬레이터 UI에서 시뮬레이션 시작 버튼을 누름
4. 시뮬레이터 엔진이 실행 이미지의 기계어 명령을 시뮬레이션하는 과정에서 전력 분석 도구는 전력 분석 모델을 참조하여 각 명령에 대한 전력 소모량 값을 계산하고, 해당 주소에 해당하는 함수에 전력 소모량 값을 더함
5. 시뮬레이션 수행이 끝날 때까지 4단계의 과정을 반복
6. 시뮬레이터 UI에서 시뮬레이션 정지 버튼이 눌러지면 분석된 함수 단위의 전력 소모량 값을 시뮬레이터 UI의 뷰(<그림 6> 우측 하단)를 통하여 출력

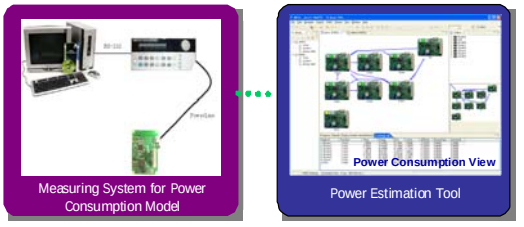

그림 6. 전력 측정 시스템 및 분석 결과
Fig. 6. Power measuring system and power analysis result

전력 분석 모델은 <그림 6> 좌측의 전력 측정 시스템을 통하여 개발되었다. 전력 측정 시스템은 좌측의 호스트 시스템, 우측의 멀티미터, 하단의 센서 노드로 구성된다. 멀티미터는 센서 노드의 파워 라인과 연결되어 센서 노드에서 소비되는 전류를 측정하고, 측정된 전류 값은 호스트 시스템에 전달된다.

센서 노드에 분석에 필요한 명령을 반복적으로 수행하면서, 명령에 대한 전류 값을 구하고, 전압과 명령 수행 시간은 알고 있으므로, 해당 명령에 대한 에너지 값을 구할 수 있다. 일례로 <그림 7> 과 같

그림 7. mov 명령어의 소모 전류 측정 과정
Fig. 7. Process of measuring power consumption of mov instruction

이 mov 명령을 반복적으로 수행시키며, 멀티미터를 사용하여 mov 명령에 대한 전류 값을 측정한다.

Ⅳ. 결 론

본 연구에서는 USN 응용 소프트웨어를 빠르고 편리하게 개발할 수 있는 통합개발환경(Nano Esto)을 개발하였다. 본 연구의 결과물은 USN 응용 소프트웨어의 개발에 필요한 모든 기능들을 Eclipse 사용자 인터페이스를 통해서 제공하고 있으며, 특별히 네트워크 환경의 시뮬레이션 기능을 통해서 타겟 센서 노드에 적재되는 프로그램의 정확성 및 안정성을 검증할 수 있다. 뿐만 아니라 센서 노드들의 자원 제약적인 특성을 고려하여, 응용의 성격에 따라 최적의 커널을 구성할 수 있는 기능을 제공한다. 이러한 기능은 기존의 다른 개발 도구에서는 찾아볼 수 없었던 기능으로 관련 기술에서의 비교 우위를 차지할 것으로 예상된다.

참고문헌

[1] P. Varhol, "Integrated software tools improve productivity and code quality", Electronic Design, pp. 62-70, October 1999.
[2] S.V. Tyle, "Engineering software tools meet demands", Electronic Design, pp. 71-80, June 1994.
[3] http://www.atmel.com
[4] http://www.eclipse.org
[5] P. Levis, N. Lee, M. Welsh, D. Culler, "TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications," Proc. of 1st ACM Conference on Embedded Networked Sensor Systems, pp.126-137, November 2003.
[6] I. Chun, C. Lim, NanoEsto Debugger : The Tiny Embedded System Debugger, ICACT, pp. 715-718, February 2006.
[7] http://savannah.nongnu.org/projects/avrdude

저 자 소 개

정창희

2003년 충남대학교 컴퓨터공학과 학사. 2005년 서울대학교 컴퓨터공학과 석사. 현재, 한국전자통신연구원 연구원. 관심분야: 컴파일러, 임베디드 소프트웨어, 병렬 처리.

Email: chjung@etri.re.kr

우덕균

1993년 홍익대학교 컴퓨터공학과 학사. 1995년 홍익대학교 전자계산학과 석사. 2001년 홍익대학교 전자계산학과 박사. 현재, 한국전자통신연구원 선임연구원. 관심분야: 컴파일러, 임베디드 소프트웨어 개발환경.

Email: dkwu@etri.re.kr

김용상

1996년 고려대학교 전산학과 학사. 현재, 한국전자통신연구원 연구원. 관심분야: 임베디드 소프트웨어, 센서 네트워크 시뮬레이션.

Email: yskim@etri.re.kr

전인걸

1996년 성균관대학교 정보공학과 학사. 1998년 성균관대학교 정보공학과 석사. 현재, 한국전자통신연구원 선임연구원. 관심분야: 임베디드 소프트웨어, 소프트웨어 개발도구, 소프트웨어 품질.

Email: igchun@etri.re.kr

임채덕

전남대학교 전산학 학사. 충남대학교 전산학 석·박사. 현재 한국전자통신연구원 책임연구원, 임베디드SW개발도구연구팀장. 관심분야: 임베디드 소프트웨어, 분산실시간컴퓨팅, 임베디드 시스템 개발 도구

Email: cdlim@etri.re.kr