USN 소프트웨어 개발 도구 동향

A Trend of USN Software Development Tool

임베디드 S/W 기술 동향 특집

백광진 (K.J. Paek) S/W개발도구연구팀 연구원
전인걸 (I.G. Chun) S/W개발도구연구팀 선임연구원
우덕균 (D.K. Woo) S/W개발도구연구팀 팀장

목 차

임베디드 시스템을 위한 응용 프로그램 개발 도구로서 통합개발환경을 이용하는 것은 소프트웨어 개발의 생산성과 코드의 완성도를 향상시킬 수 있다는 점에서 매우 중요하게 인식되고 있다. 최근에 USN에 대한 관심이 높아지면서 이를 위한 여러 가지 응용 소프트웨어들이 개발되고 있으나, 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 고에서는 이와 같은 문제점을 해결하기 위하여 USN 응용 소프트웨어를 빠르고 편리하게 개발할 수 있는 통합개발환경의 동향을 살펴보고 ETRI의 본 연구팀에서 개발한 USN 소프트웨어 통합개발 도구인 “NanoEsto”를 기술하고 상용 제품과의 비교를 수행하였다.

I. 서론

임베디드 시스템을 위한 응용 프로그램 개발 도구로서 통합개발환경을 이용하는 것은 소프트웨어 개발의 생산성과 코드의 완성도를 향상시킬 수 있다는 점에서 매우 중요하게 인식되고 있다[1],[2]. 최근에 USN에 대한 관심이 높아지면서 이를 위한 여러 가지 응용 소프트웨어들이 개발되고 있으나, 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 원고에서는 이와 같은 문제점을 해결하기 위하여 USN 응용 소프트웨어를 빠르고 편리하게 개발할 수 있는 통합개발환경의 동향을 살펴보고 ETRI에서 개발한 Nano-Esto와 상용 제품과의 비교를 수행한다.

NanoEsto는 소스 편집, 컴파일, 이미지 적재 및 실행 기능과 소스 수준의 디버깅 기능을 비롯해 USN 응용 소프트웨어의 개발에 필요한 기능들을 제공한다. 또한 센서 노드의 자원이 충분하지 않음을 고려하여 응용의 성격에 따라서 최적의 커널을 구성할 수 있는 기능을 제공한다.

본 원고의 Ⅱ장에서는 USN 소프트웨어 개발 도구에서 필요로 하는 요소 기술에 관해 살펴보고, Ⅲ장에서는 현재 상용 제품을 살펴본다. Ⅳ장에서는 ETRI의 본 연구팀에서 개발한 USN 소프트웨어 통합개발 도구인 “NanoEsto”를 기술한다. Ⅴ장에서 상용 제품과 NanoEsto의 기능을 비교해보고, Ⅵ장에서 결론을 맺는다.

Ⅱ. 요소 기술

1. 센서 네트워크 OS

현재 개발되어 있는 센서 노드를 위한 초소형 운영체제는 크게 두 가지로 분류될 수 있다. 하나는 이벤트 기반 모델로 TinyOS와 SOS, Contiki 등이 여기에 속하고 이벤트 기반의 모델은 하나의 프로그램 흐름을 가지며 각각 이벤트에 따라 별도의 핸들러를 가지고 처리하게 된다. 나머지 다른 부류는 멀티 스레드 환경을 제공하는 운영 체제로 Nano-Qplus, MANTIS, Nano-RT, RETOS 등이 여기에 속한다[3].

전통적으로 성능상의 이점과 제어 흐름 상태 관리가 쉽다는 이유로 이벤트 기반의 프로그래밍 모델이 선호되어 왔다. 하지만 멀티 스레드 기반의 운영체제는 여러 일을 효과적으로 처리하고 개발의 복잡도를 낮출 수 있어 보다 손쉽게 개발할 수 있다는 장점이 있다.

센서 노드는 극히 적은 리소스를 가지고 있기 때문에 센서 네트워크 OS 개발은 많은 제약을 가지고 있다. 먼저, 커널의 크기가 제한된다. 제한된 상황에서 센서 노드에 맞는 적합한 구조의 커널이 필요하다. 다음으로 상대적으로 컴퓨팅 파워가 낮은 MCU를 사용하기 때문에 최대한 복잡한 연산을 줄이고 코드를 최적화하여 MCU의 부하를 줄여야 한다. 또한 센서 네트워크가 최대한 라이프 타임을 보장하기 위해 전력 보존 문제 또한 매우 중요하다. 하나의 노드뿐만 아니라 네트워크 전체의 지속 시간과 효율성을 고려한 전력관리 기술이 필요하다.

2. 교차 개발 환경(Cross Development Environment)

임베디드 시스템 개발을 위하여 호스트 컴퓨터가 필요한데 그 이유는 임베디드 시스템 자체가 개발 중인 경우가 대부분이고 컴파일러까지 설치하기에는 메모리 용량이나 CPU 속도가 감당할 수 없기 때문이다. 호스트 컴퓨터에서는 타깃 시스템용 실행파일을 생성하는 컴파일러가 필요하며 이를 크로스컴파일러라 한다. 보통 툴 체인은 크로스 컴파일러뿐만 아니라 실행파일 변환기 등 개발환경에 필요한 소프트웨어의 집합을 의미한다. 호스트 컴퓨터는 x86 계열의 데스크톱 컴퓨터가 대부분이므로 x86 계열의 컴퓨터에서 타깃 보드의 CPU가 이해하는

기계어를 생성해주는 X86-to-target 크로스 컴파일러가 필요하다.

툴 체인의 핵심은 크로스 컴파일러이지만 때로는 ELF-32 포맷의 실행 파일로부터 순수 바이너리만 추출해 줄 수 있는 목적 파일 포맷 변환기 등도 포함되어 있으며, 그 외 makefile이나 링커 스크립트, 스타트업 코드까지 포함하기도 한다. 이러한 목적 파일을 만드는 도구 이외에 목적 파일을 타깃보드로 다운로드 하는 도구 또는 롬 에뮬레이터 등도 포함된다.

3. 통합 개발 환경

앞서 언급한 도구들이 모두 다른 환경에서 동작하거나 사용법이 제 각각이라면 사용자는 혼란을 겪게 될 것이고 개발기간이 늘어나는 결과를 초래할 것이다. 이를 방지하기 위해 일관되고 사용하기 쉬운 통합개발환경을 사용자에게 제공해야 한다.

최근 많은 도구 회사들이 IDE의 기반 플랫폼으로 Eclipse를 선택하고 있다. Eclipse는 공개 플랫폼이며 누구나 플러그인을 개발하여 Eclipse의 기능을 확장할 수 있다. 많은 도구들은 내장 에디터가 성능이 나쁘다고 해서 바꿀 수 없으며 언어가 바뀜에 따라서 계속해서 문법뿐만 아니라 도구의 사용법을 배워야 했다. 이러한 단점을 효과적으로 해결한 것이 Eclipse이다. 원래 IBM에서 Eclipse 프로젝트를 시작하였으며 현재는 세계의 많은 개발자들에 의해 수정되고 배포되고 있다. ETRI에서도 Eclipse를 기반으로 하여 Esto와 NanoEsto라고 하는 임베디드 시스템 통합개발환경을 작성하였다.

4. 디버깅

디버깅이란 소프트웨어 안에 있는 버그를 찾아서 고치는 행동을 의미한다. 버그란 단순히 문법적 오류에서부터 실행중에도 찾아내기 어려운 오류까지 다양한 형태로 존재한다.

디버거는 주로 응용 프로그램에서 제공하는 문제 판별기능(오류 로그 파일, 오류 메시지 등)이나 시스템에서 제공하는 문제 판별 도구로 해결할 수 없는 응용 프로그램 문제에 사용한다. 디버거는 소스코드가 있을 경우 특히 유용하다.

임베디드 시스템의 디버깅은 타깃시스템과 호스트 컴퓨터로 이루어진 원격 개발환경을 이용하는 방법이 주로 사용되고 있다. 개발하고자 하는 임베디드 시스템인 타깃시스템에서 개발한 프로그램을 실행시킨다. 개발용 호스트에서 디버거를 이용해 타깃시스템과 특정 인터페이스를 통하여 연결되며 소스코드를 참조하면서 타깃시스템에서 실행되는 프로그램을 제어한다.

Ⅲ. 제품 동향

1. MS사의 .NET Micro Framework

마이크로소프트 .NET Micro Framework SDK는 .NET 라이브러리의 서브셋을 이용하여 C# 언어로 임베디드 응용 프로그램의 개발을 지원한다[4].

.NET Micro Framework 구조는 소형 디바이스에 최적화되어 있으며 Microsoft Visual Studio 2005와 C#을 이용하여 임베디드 시스템 프로그램을 가능하도록 한다. 다음과 같은 디바이스가 포함된다.

- 센서 네트워크
- 로보틱스
- 웨어러블 디바이스
- 의료 장비
- 산업용 자동화 디바이스
- 효과적이고 적은 자원을 사용하는 .NET 클라이언트를 요구하는 기타 디바이스들

(그림 1)은 Visual Studio 2005에서 .NET Micro Framework의 예제를 실행한 화면이다. .NET Micro Framework SDK는 디바이스 에뮬레이터를 포함하고 있다.

(그림 2)는 .NET Micro Framework 계층 구조를 나타낸다. 각 계층 구조를 살펴보면 다음과 같다.

- Hardware Layer: 마이크로 프로세서와 기타 회로를 포함한다. .NET Micro Framework는 하드웨어 플랫폼이나 OS 위에서 실행될 수 있도록 포팅이 가능하다.
- RCL: RCL은 세 가지 요소인 CLR, HAL 그리고 PAL로 구성된다. .NET Micro Framework CLR은 .NET Framework CLR의 서브셋으로 .NET Framework에 의해 제공되는 런타임 환경이다. HAL과 PAL은 하부의 시스템 하드웨어를 제어하며 CLR에 의해 호출되는 C++ 함수들의 집합이다.
- Class Library Layer: 재사용 가능한 객체 지향 타입들의 집합으로서 C# 라이브러리이다.

(그림 1) .NET Micro Framework 실행 화면

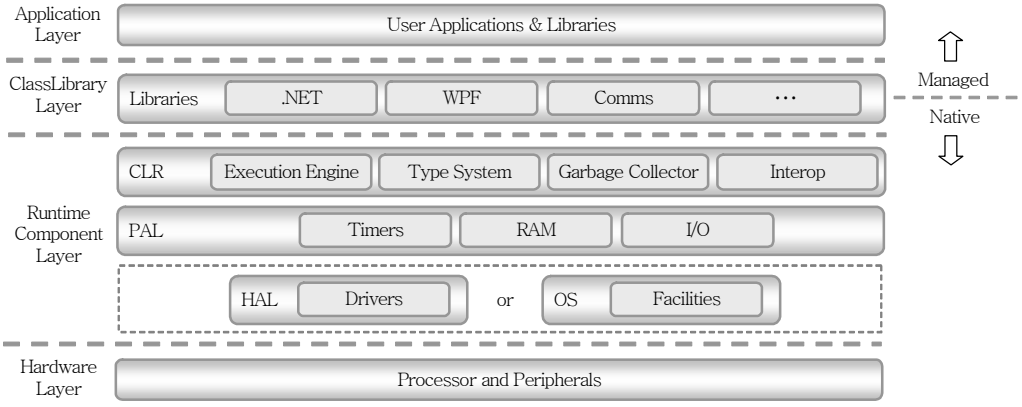
- Application Layer: 관리되는(managed) 응용 프로그램을 포함한다. C# 언어만을 지원하고 있다.

2. Crossbow사의 MoteWorks

MoteWorks는 무선 센서 네트워크 응용 프로그램을 위한 완전한 소프트웨어 개발환경을 제공하며 세 가지 소프트웨어 계층으로 구성된다[5].

- Mote Tier: XMesh 소프트웨어가 실행되며 센서 노드들을 통해 메시 네트워크(mesh network)를 형성하여 신뢰성 있는 통신 백본(backbone)을 제공한다.
- Server Tier: 무선 센서 네트워크로부터의 데이터의 버퍼링과 변환을 담당한다. Xserve와 XOtap은 서버 계층 응용 프로그램으로서 PC 또는 Stargate에서 실행될 수 있다.
- Client Tier: 센서 네트워크 관리를 위한 시각적 소프트웨어와 그래픽 인터페이스를 사용자에게 제공하는 MoteView라는 클라이언트 소프트웨어를 제공한다. (그림 3), (그림 4)에서 세 계층과 네트워크 구성도를 살펴볼 수 있다.

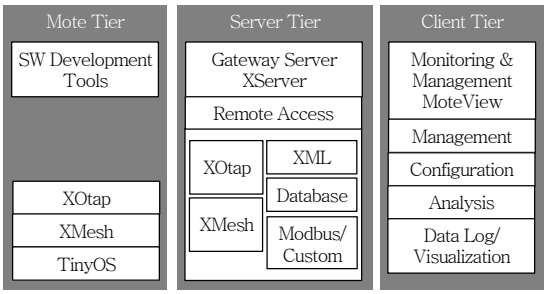
MoteWorks는 TinyOS를 포함하고 있으며 TinyOS는 버클리대학에서 처음 개발된 공개 소스 운영체제이다. 무선 센서 네트워크에서 가장 널리 분포되어 있는 운영 체제이며 적은 메모리 사이즈를 요



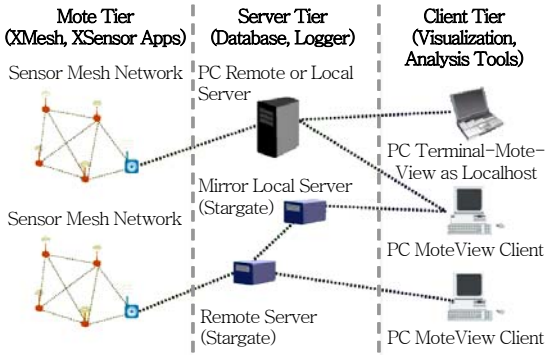
(그림 2) .NET Micro Framework 계층 구조

구하는 저전력 디바이스를 위해 설계된 컴포넌트 기반의 이벤트-드리븐(event-driven) 방식의 운영 체제이다.

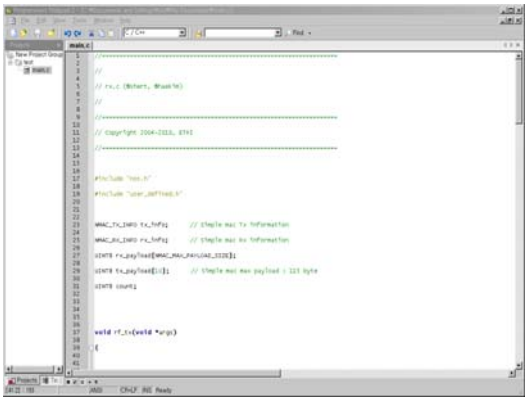
MoteWorks는 타깃노드를 위한 크로스 컴파일러와 TinyOS 프로그램 개발을 위한 진보된 에디터를 제공한다. (그림 5)는 MoteWorks의 Programmers Notepad 2를 실행한 화면이다.



(그림 3) MoteWorks의 3 계층



(그림 4) MoteWorks의 네트워크 구성



(그림 5) MoteWorks의 PN2 실행 화면

MoteWorks가 제공하는 소프트웨어 패키지는 다음과 같다.

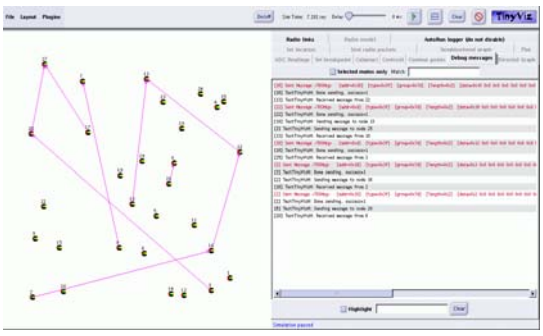
- TinyOS와 MoteWorks Tools
- nesC Compiler
- Cygwin
- AVR Tools
- Programmer's Notepad
- XSniffer
- MoteConfig
- Graphviz
- PuTTY와 TortoiseCVS

MoteWorks가 제공하는 개발 환경은 TinyOS의 개발 환경을 포함하며 GUI과 커맨드라인 환경을 병행하여 사용한다.

3. TinyOS

TinyOS는 임베디드 센서 노드를 사용하기 위해 설계된 이벤트 기반의 운영 체제이다. 특히, 최소한의 하드웨어 사양의 센서 노드에서 실행될 수 있도록 설계되었다. TinyOS의 프로그래밍 언어는 확장된 C언어인 "nesC"를 사용하며 버클리대학에서 최초로 개발되었다. 리눅스와 윈도 환경을 지원한다[6].

TOSSIM은 TinyOS 센서 네트워크를 위한 이산 이벤트 시뮬레이터이며 TinyOS 코드로부터 직접 컴파일 된다. 프로그래머는 타깃노드에 퓨징할 이미지를 컴파일하는 대신 TOSSIM 플랫폼에서 컴파일하



(그림 6) TinyViz

여 호스트 컴퓨터에서 실행할 수 있다. TOSSIM은 프로그래머가 작성한 프로그램을 PC의 시뮬레이션 환경에서 디버그, 테스트 및 분석을 가능하게 한다.

(그림 6)은 TOSSIM의 시뮬레이션 결과를 시각적으로 볼 수 있는 TinyViz의 실행 화면이다.

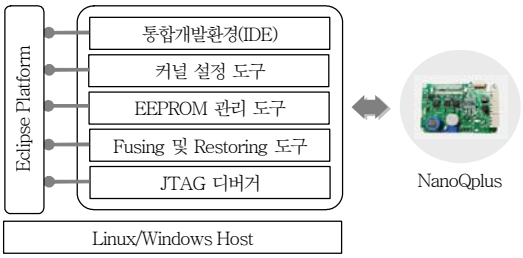
Ⅳ. NanoEsto 소개

USN을 구성하는 센서 노드는 제한된 메모리와 MCU 원을 갖기 때문에 응용 프로그램 개발 환경을 직접 구동할 수 없다. 따라서 USN 응용 소프트웨어의 개발은 데스크톱 PC와 같은 고사양의 호스트 컴퓨터에서 수행되고, 테스팅, 디버깅을 위하여 완성된 이미지를 타깃센서 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 생산성 저하를 초래한다.

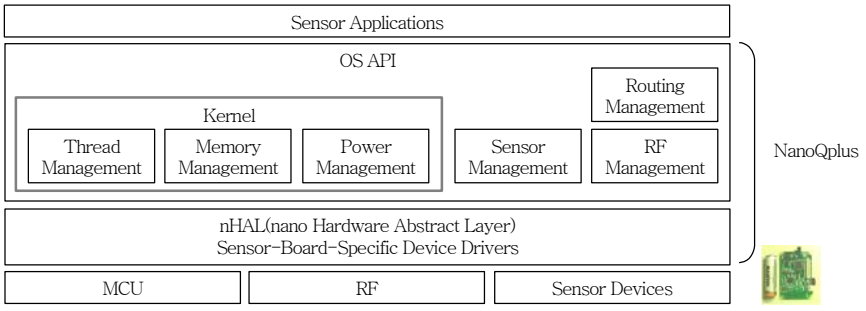
(그림 7)은 NanoEsto의 구조도를 나타낸다. NanoEsto에서는 하드웨어 플랫폼으로 ATMEL사의 AVR MCU를 탑재한 센서 노드를 지원한다[3]. 타깃센서 노드는 이미지 적재를 위한 USBasp와 하드웨어 디버깅을 위한 AVR JTAGICE mkII를 포함하고 있으며, 각각 USB 인터페이스에 의해 호스트 컴퓨터와 연결된다.

NanoEsto는 NanoQplus에서 실행되는 유비쿼터스 센서 네트워크를 위한 응용 프로그램 개발을 지원하는 진보된 소프트웨어 개발환경이다. NanoEsto는 USN 응용 프로그램 작성과 실행을 위한 포인트 & 클릭(point-and-click) 프로그래밍 환경을 제공하고 수작업에 의한 개발 시간을 줄임으로써 개발자의 생산성을 향상시킨다. NanoEsto는 Windows 뿐만 아니라 Linux에서 Eclipse GUI 스타일과 동일한 인터페이스를 제공한다.

NanoQplus는 ETRI에서 개발한 센서 네트워크 운영체제이다. NanoQplus는 센서 네트워크 운영체제의 요구조건을 최대한 고려하여 제작되었다. 이 운영체제의 가장 큰 특징은 친숙한 C 스타일의 쉬운 프로그래밍 환경을 제공하기 위해서 멀티 스레딩을 지원한다는 점이다. 센서노드용 초소형 운영체제 기술(NanoQplus)은 의료, 환경 방재, 스마트 홈, 국방, 산업 등의 다양한 응용 분야에 사용될 수 있으며, 미래 첨단 정보 서비스 제공에 필요한 다양한 임베디드 시스템에 공통으로 탑재되어, 응용 프로그램이 최적의 성능 및 기능을 발휘할 수 있게 하는 핵심 임베디드 소프트웨어 기술로 그 중요성 및 시장성은 더욱 확대되고 있다[7].



(그림 7) NanoEsto 구조도

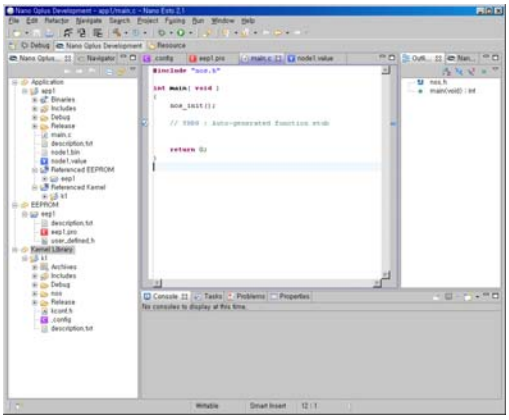


(그림 8) NanoQplus OS 구조도

NanoQplus의 경우 센서 및 구동장치의 종류에 따라 운영체제를 최적화하여 재구성하는 기능, 다양한 스케줄러 및 무선통신 방식의 지원, 모니터링 소프트웨어, 원격 업그레이드 등의 특징을 갖고 있어 미래 사회에서 요구되는 다양한 센서 네트워크 응용 환경에서 활용 가능하다. (그림 8)은 NanoQplus의 구조도이다.

1. NanoEsto IDE

NanoEsto IDE는 크로스 플랫폼 개발 환경에서 타깃시스템에서 수행되는 USN 응용 소프트웨어를 개발하기 위한 목적으로 개발되었으며, 프로젝트 관리자, 소스 편집기, 크로스 컴파일 툴체인과 실행 이미지 적재를 위한 퓨징 도구, 커널 설정 도구, EEPROM의 데이터 구조를 관리할 수 있는 EEPROM 관리 도구, EEPROM의 데이터를 관리할 수 있는 리스토어링 도구 등을 포함한다. 소스 편집기와 프로젝트 관리자의 경우 NanoQplus 기반의 C 언어에 특화되어 있으며 Eclipse 플랫폼이 기본적으로 지원하는 텍스트 편집기와 프로젝트 관리자를 확장하여 USN 응용 프로그램 개발에 적합한 기능을 수행하도록 구현하였다. (그림 9)는 NanoEsto의 IDE 실행 화면을 나타낸다.

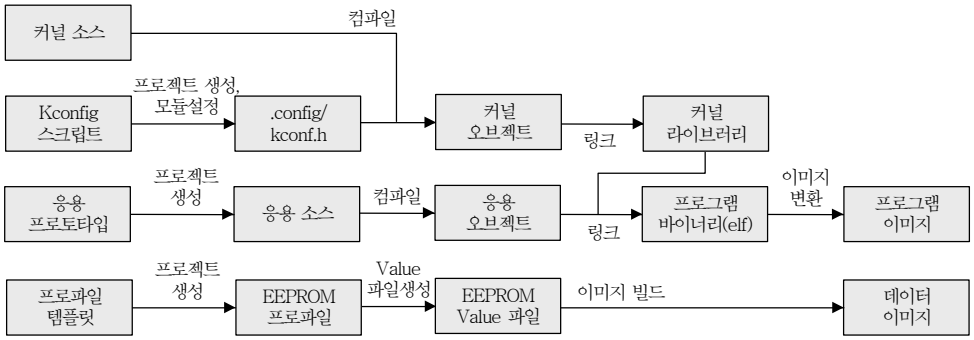


(그림 9) NanoEsto IDE 화면

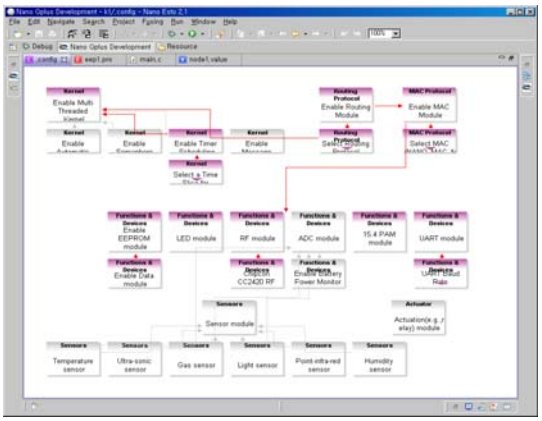
(그림 10)은 NanoEsto에서 최종 결과물인 프로그램 이미지와 데이터 이미지가 생성되는 과정을 보여주고 있다.

2. 커널 설정 도구

일반적으로 USN을 구성하는 타깃센서 노드의 프로그램 메모리 사이즈는 8~128kB로 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 NanoEsto는 GUI 기반의 커널 설정 도구를 포함한다. 특히 ETRI에서 개발된 USN용 초소형 운영체제인 NanoQplus를 적용한 타깃설정도구를 지원하고 있으며, 이는 각 모듈들의 의존성을 자동으로 검사하고 원격 개발 IDE와 연계하여 선택된 모듈에 대한 기본적인 스켈레톤 코드를 생성한다. (그림 11)은 커널 설정 도구를 실행한 화면이다.



(그림 10) 프로그램 및 데이터 이미지 생성 과정



(그림 11) 커널 설정 도구 화면

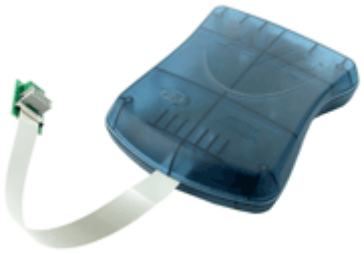
3. JTAG 기반 디버거

NanoEsto는 JTAG 기반 디버거를 지원한다. 응용 프로그램이 직접 타깃센서 노드에서 동작하는 환경에서 디버깅을 하며 개발자의 요구에 따라 변수, 레지스터, 메모리, 스택 등의 정보를 제공한다.

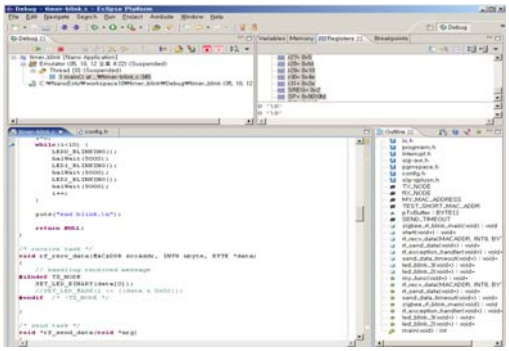
JTAG 기반 디버거는 IEEE1149.1 표준에 정의되어 있는 JTAG 포트를 이용하여 실제 타깃의 센서 노드에서 동작하는 NanoQplus 응용 프로그램의 디버깅을 지원한다. JTAG 기반 디버거는 Eclipse 사용자 인터페이스, 디버깅 엔진, ICE 장비로 구성되어 있으며 그 구조는 다음과 같다.

JTAG 기반 디버거는 IEEE1149.1 표준에 정의되어 있는 JTAG 포트를 이용하여 실제 타깃의 센서 노드에서 동작하는 NanoQplus 응용 프로그램의 디버깅을 지원한다. JTAG 기반 디버거는 Eclipse 사용자 인터페이스, 디버깅 엔진, ICE 장비로 구성되어 있으며 그 구조는 다음과 같다.

- 이클립스 사용자 인터페이스: 사용자 인터페이스는 개발자에게 편리한 환경을 제공하고 다른 도구와의 통합을 위해 이클립스 플러그인으로 개발되었으며 이클립스 플랫폼에서 기본적으로 제공하는 환경과 동일한 환경을 제공한다.
- 디버깅 엔진: 디버깅 엔진은 GNU GDB를 기반으로 개발되었다. 사용자 인터페이스를 통해 입력된 명령을 수행하며 AVR JTAGICE mkII를 통해 타깃시스템을 제어하고 필요한 정보를 가져온다.
- AVR JTAGICE mkII: AVR JTAGICE mkII는 Atmel사에서 제작한 on-chip 디버깅을 위한 강력한 개발 도구이다. IEEE1149.1 인터페이스가 있는 모든 AVR 8비트 RISC MCU와 AVR32 32비트 DSP/MCU를 지원한다.



(그림 12) Atmel사의 AVR JTAGICE mkII 장비

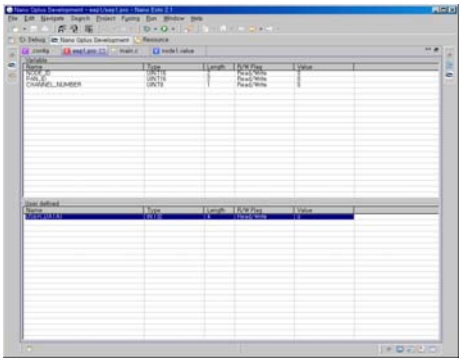


(그림 13) NanoEsto의 JTAG 기반 디버깅 화면

(그림 12)는 Atml사의 JTAGICE 장비를 보여주고 있으며, (그림 13)은 NanoEsto에서 JTAG 디버거를 이용하여 C 소스코드를 디버깅하는 화면을 보여주고 있다.

4. EEPROM 관리 도구

EEPROM 관리 도구에는 (그림 14)와 같이 시스템 변수와 사용자 변수가 있으며 사용자 변수만 추가 및 데이터 타입 수정이 가능하다. NanoQplus에서 정의된 데이터 타입 선택이 가능하며 R/W 플래그는 Read, Read/Write, N/A 중에서 선택할 수 있다.



(그림 14) EEPROM 관리 도구 실행 화면

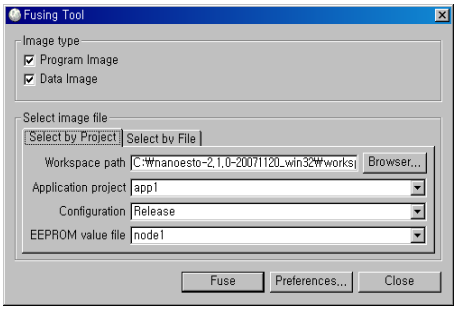
데이터 이미지 생성시 EEPROM 상의 해당 데이터에 접근하기 위한 정보를 헤더 파일로 생성한다. 사용자 정의 변수에 대한 위치 정보를 담은 user_defined.h 파일이 생성된다. R/W 플래그가 N/A일 경우, EEPROM에 기록되지 않고 헤더파일에만 정의된다.

5. 퓨징 및 리스토어링 도구

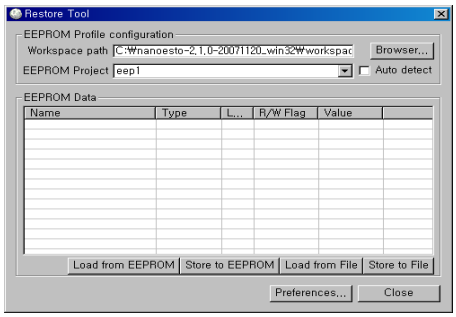
(그림 15)는 프로그램 및 데이터 이미지를 퓨징하기 위한 퓨징 도구의 실행 화면이다. 프로젝트를 통해 선택하거나 직접 파일을 선택할 수 있다.

(그림 16)은 리스토어링 도구의 실행 화면이다. 현재 연결된 타깃노드의 EEPROM으로부터 데이터 이미지를 로드하거나 호스트 컴퓨터의 파일로부터 로드가 가능하다. 읽어온 데이터 이미지는 값을 변경하여 다시 타깃노드에 저장하거나 파일로 호스트 컴퓨터에 저장할 수 있다.

USBasp는 Atmel AVR 컨트롤러를 위한 회로가



(그림 15) NanoEsto의 퓨징 도구 창



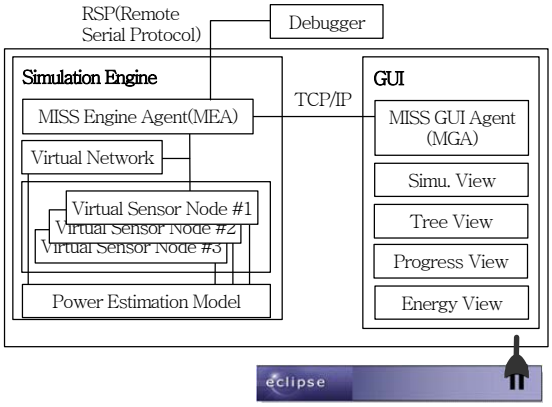
(그림 16) NanoEsto의 리스토어링 도구 창

(그림 17) USBasp 장비

내장된 USB 프로그램 도구이며 ATMega48이나 ATMega8과 몇 개의 컴포넌트들로 구성된다. 프로그램을 위해서는 단지 USB 드라이버만 사용된다. (그림 17)에서 USBasp 장비를 볼 수 있다.

6. 기계 명령어 기반 센서 네트워크 시뮬레이터

기계 명령어 기반 센서 네트워크 시뮬레이터(MISS)는 센서 네트워크 환경에서 센서 노드 개발 이전에 USN 응용 프로그램을 검증할 수 있는 도구로서 기계어 수준의 명령어 시뮬레이션 방법을 사용하여 실제 타깃센서 노드의 동작을 시뮬레이션 한다. 이를 통하여 센서 노드에 탑재되는 응용 소프트웨어 및 운영 체제의 동작과 네트워크의 안정성을 검증할 수 있으며, 예로 센서 노드의 위치 및 통신 환경을 고려한 네트워크 시뮬레이션이 가능하다. MISS는 (그림 18)과 같이 구성이 되며 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 출력된



(그림 18) MISS 구조도

다. 뿐만 아니라 센서 하드웨어에 부착된 LED 출력 표시 기능을 통해서 USN 응용 프로그램을 보다 현실적으로 검증할 수 있도록 구현하였다.

MEA 모듈은 디버거나 MGA로부터 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 통신에 대하여 전송 에러율과 전파 수신 거리를 적용하여 실제 환경에서의 RF 통신과 같은 시뮬레이션을 제공한다.

Virtual sensor node 모듈은 타깃센서 하드웨어를 소프트웨어로 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 도구와의 통합 및 사용자에게 편리한 환경 제공을 위하여 이클립스 프러그인 형태로 구축된다.

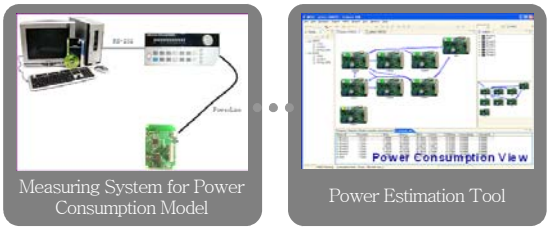
7. 전력 분석 도구

NanoEsto의 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.

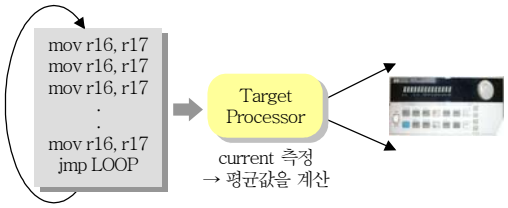
시뮬레이터와 연동하여 전력 소모량 분석 및 결과 출력은 다음과 같이 진행된다.

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 19) 우측 하단)를 통하여 출력

전력 분석 모델은 (그림 19) 좌측의 전력 측정 시스템을 통하여 개발되었다. 전력 측정 시스템은 좌측의 호스트 시스템, 우측의 멀티미터, 하단의 센서 노드로 구성된다. 멀티미터는 센서 노드의 파워 라



(그림 19) 전력 측정 시스템 및 분석 결과



(그림 20) mov 명령어의 소모 전류 측정 과정

인과 연결되어 센서 노드에서 소비되는 전류를 측정하고, 측정된 전류 값은 호스트 시스템에 전달된다.

센서 노드의 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 20)과 같이 mov 명령을 반복적으로 수행시키며 멀티미터를 사용하여 mov 명령에 대한 전류 값을 측정한다.

Ⅴ. 관련 도구 비교

<표 1>은 NanoEsto와 기타 개발 도구와 주요 항목을 비교한 표이다. 앞서 살펴본 .NET Micro Framework와 MoteWorks를 NanoEsto와 여러 가지 측면에서 비교를 해보면 우선 실행환경 면에서 NanoEsto는 Windows와 Linux 환경을 모두 지원함으로써 사용자에게 높은 융통성을 제공하며 개발에 관련된 모든 작업을 커맨드라인을 사용하지 않고 IDE에서 사용자에게 편리한 GUI를 통해 수행할 수 있다. 특히 JTAG 디버깅 시 디버깅과 소스 수정을 IDE에서 동시에 수행할 수 있으므로 사용자의 생산성과 작업 능률을 높일 수 있다.

NanoEsto의 EEPROM은 사용자가 직접 코딩을 하지 않아도 IDE를 통해 EEPROM에 데이터를 저장할 수 있게 해주며 이 기능은 프로그램과 데이터를 분리해서 응용 프로그램을 작성 가능하게 하므로 하나의 프로그램 이미지를 사용하고 각기 다른 데이터 이미지를 사용함으로써 각각의 노드를 위한 프로그램 이미지를 작성하는 수고를 덜 수 있다.

NanoEsto의 전력분석 도구는 에너지 효율성이 중요한 센서 네트워크의 특성을 만족시켜 줄 수 있는 도구로서 응용 프로그램의 에너지 소비량을 프로그램 코드를 실행하면서 측정하여 에너지 효율이 높은 센서 노드용 응용 프로그램을 작성할 수 있도록 해준다.

〈표 1〉 NanoEsto와 다른 개발 도구와의 비교

	.NET Micro Framework	MoteWorks	NanoEsto
사용 언어	C#	nesC	C
지원 OS	.NET Micro Framework	TinyOS	NanoQplus
실행 환경	Windows	Windows	Windows, Linux
퓨징 도구	없음	커맨드 라인	IDE 포함
EEPROM 관리 도구	없음	없음	IDE 포함
센서 네트워크 시뮬레이터	없음	지원	지원
원격 업그레이드	없음	지원	지원
전력 분석	없음	없음	지원
멀티 스레딩	지원	없음	지원
JTAG 디버깅	없음	별도 지원	IDE 포함

이상의 비교로 NanoEsto는 다른 도구와 비교했을 때 우수한 USN 소프트웨어 개발 도구임을 판단할 수 있다.

Ⅵ. 결론

NanoEsto는 USN 응용 소프트웨어의 개발에 필요한 기능들을 Eclipse 사용자 인터페이스를 통해

● 용 어 해 설 ●

JTAG(Joint Test Access Group): 1980년대 후반에 연구중이던 Boundary-Scan 설계를 IEEE에서 1990년에 표준화하였고 IEEE std 1149.1이 제정되었다. Boundary-Scan Cell이 핵심이며 임베디드 시스템 개발시 디버깅하기 위한 장비를 칭하기도 함

EEPROM(Electrically Erasable Programmable Read-Only Memory): 사용자가 메모리 내의 내용을 수정할 수 있는 롬으로써, 정상보다 더 높은 전압을 이용하여 반복적으로 지우거나, 다시 기록할 수 있음

서 제공하고 있으며, 센서 노드의 자원 제약적인 특성을 고려하여 응용의 성격에 따라 최적의 커널을 설정할 수 있는 기능과 퓨징 및 EEPROM 관리 기능 등을 제공한다.

현재 상용 도구인 MS사의 .NET Micro Framework, Crossbow사의 Moteworks와 ETRI의 NanoEsto를 살펴 보았으며, 이를 통해 도구들 간의 기능을 비교하였다. 이 비교를 통하여 NanoEsto가 더 풍부한 기능을 제공함으로써 USN 응용 프로그램 개발의 생산성을 높일 수 있는 도구임을 판단할 수 있었다. 향후 NanoEsto가 계속적으로 발전되어 관련 기술에서 비교 우위를 차지할 것으로 기대된다.

약 어 정 리

DSP	Digital Signal Processor
EEPROM	Electrically Erasable Programmable Read-Only Memory
IDE	Integrated Development Environment
JTAG	Joint Test Action Group
MCU	Micro Controller Unit
MEA	MISS Engine Agent
MGA	MISS GUI Agent
MISS	Machine Instruction-level Sensor network Simulator
SDK	Software Development Kit
USB	Universal Serial Bus
USN	Ubiquitous Sensor Network

참 고 문 헌

[1] P. Varhol, “Integrated Software Tools Improve Productivity and Code Quality,” Electronic Design, Oct. 1999, pp.62-70.

[2] S.V. Tyle, “Engineering Software Tools Meet Demands,” Electronic Design, June 1994, pp.71-80.

[3] 박창순, 이광용, 이형석 정호영 외, “생활 속의 임베디드 소프트웨어,” U-Book, 2007.

[4] http://www.xbow.com

[5] Donald Thompson and Colin Miller, “.NET Micro-Framework White Paper,” Nov. 14, 2006.

[6] http://www.qplus.or.kr/

[7] http://www.tinyos.net/