임베디드 소프트웨어 통합 개발 환경 기술

한국전자통신연구원 임채덕 · 김태호

1. 소 개

과거에는 주로 하드웨어로 개발되어지던 임베디드 시스템에서 소프트웨어가 차지하는 비중이 점차로 높아지면서, 소프트웨어의 개발이 심각한 문제로 대두되고 있다. 즉, 소프트웨어의 복잡도는 급속도로 높아지는 반면에 시장에서 요구되는 개발 기간은 더욱 짧아지고 있다. 따라서, 임베디드 시스템의 개발자는 개발을 편리하게 할 수 있는 도구를 요구하고 있다.

임베디드 소프트웨어가 장착되는 시스템은 일반적으로 프로세서의 처리 능력과 메모리 용량 등의 자원에 제약이 있기 때문에, 소프트웨어가 장착되는 시스템에서 개발을 수행하지 않는다. 따라서, 일반적으로 임베디드 소프트웨어의 개발은 상대적으로 자원에 제약이 적은 PC나 서버 환경에서 이루어진다. 이와 같이 컴파일 등이 이뤄지는 환경과 실제 프로그램이 운영되는 환경이 다르다는 점이 임베디드 소프트웨어의 개발 환경의 가장 큰 특징이며, 이를 크로스 개발 환경이라고 한다. 이때 프로그램의 개발이 이뤄지는 컴퓨터를 호스트 시스템이라고 하고, 개발될 소프트웨어가 장착되어 동작하는 컴퓨터(혹은 시스템)을 타겟 시스템이라고 한다. 이 두 종류의 시스템은 개발 과정 중에 네트워크 또는 시리얼 라인 등의 연결 장치를 통해 연결된다. 개발 환경은 호스트 시스템에서 컴파일된 바이너리 프로그램을 타겟 시스템으로 전송하고, 타겟 시스템에서 실행 시키면서, 타겟 시스템에서 동작하는 프로그램의 동작을 호스트 시스템에서 모니터링 하고 디버깅하는 등의 기능이 제공된다.

이러한 임베디드 소프트웨어 개발 과정의 특성을 지원하는 도구가 없다면, 개발 과정에서 반복적으로 발생하는 비생산적인 행위를 수동적인 방법 (예를 들어서, 호스트에서 컴파일 하고, ftp로 타겟 시스템에 연결하여 파일을 전송하고 일일이 수행시키는 전통적인 개발 과정)으로 수행해야 하며 이것은 개발자의 생산성에 저하를 가져올 것이다. 따라서, 임베디드 소프트웨어의 개발에 적절한 환경을 제공하는 것은 반드시 필요하다.

원칙적으로 소프트웨어 공학에서 의미하는 소프트웨어 개발 환경은 컴파일 환경과 디버깅 환경이 통합된 소스 코드 편집기 뿐 만 아니라, 모델 편집도구, 소스 저장소, 테스트 도구, 배포 도구, 문서화 도구, 소스 컨벤션 검사 도구, 관리 정책 검사 도구 및 프로세스 관리 도구 등 소프트웨어 개발의 시작부터 끝까지 필요한 모든 것들을 포함한다. 이중 가장 기본적인 환경은 컴파일 환경과 디버깅 환경을 중심으로 통합된 소스 코드 편집기 (IDE; Integrated Development Environment)라고 할 수 있다. **본 고에서는 임베디드 소프트웨어의 통합 개발 환경에서 갖춰야하는 기능을 정리하고, 기존에 도구에서는 어떤 방식으로 기능을 제공하고 있는지를 기술하였다. 또한, 개발 도구는 각각의 기능뿐만 아니라 그 기능을 어떻게 통합하는가도 중요한 기술이다. 따라서, 본 고는 이 두가지 측면을 함께 기술하였다.**

임베디드 소프트웨어를 위한 IDE가 갖추어야 하는 기능에는 다음과 같은 것들이 있다.

- 타겟 이미지 자동 생성
- 크로스 컴파일 및 원격 실행의 자동화
- 사용이 편리한 GUI 기반 소스 편집 기능
- 비주얼한 원격 디버깅
  - · 소스를 보면서 한 줄씩 실행하기
  - · 변수값 보기, Watch, 실시간 표현식 평가 기능
  - · 메쏘드 (함수) 호출 구조 (예: 스택 현황 출력)
- 기존에 존재하는 소스코드를 IDE 환경에 맞는 구조로 변경하는 프로젝트 관리 기능
- GUI 설계로부터 해당되는 소스 코드가 생성되는 기능
- 타겟 모니터링 기능
- 버전 관리 기능

2절에서는 임베디드 소프트웨어 통합개발환경의 전반적인 기술 요소와 동향을 살펴보고, 3절에서는 본

연구팀에서 개발한 임베디드 소프트웨어 통합개발환경의 사례인 Esto, nano Esto, QuickDriver를 소개한다. 마지막으로 4절에서는 개발 환경 지원 도구의 발전 방향과 본 글의 결론을 맺는다.

2. 임베디드 SW 통합개발환경 주요 구성 요소 및 Eclipse 소개

본 절에서는 제1절에서 소개한 임베디드 소프트웨어 통합개발환경의 주요 기능들을 살펴보고 Eclipse 프로젝트[1]를 소개한다.

2.1 타겟 이미지 생성 도구

타겟 이미지 생성 도구는 임베디드 시스템이 동작하는 데 필요한 커널, 라이브러리, 유틸리티, 응용 프로그램들 중에서, 타겟 시스템에 필요로 하는 부분을 개발자가 선택하면 최적화된 타겟 이미지를 생성하는 도구이다. 이 도구의 예로는 Microsoft의 Platform Builder[2]와 Windows Embedded Studio[3], QNX의 System Builder[4], 그리고 한국전자통신연구원에서 개발한 타겟 빌더[5]가 있다.

타겟 이미지 생성 도구는 각 컴포넌트들 간의 의존성을 관리하므로, 개발자가 어떤 컴포넌트를 선택하면 그 컴포넌트의 실행을 위해서 필요한 컴포넌트들을 타겟 이미지에 포함시킨다. 예전에는 커널 컴포넌트들 간의 의존성과 응용 프로그램들 간의 의존성이 따로 관리되었으나 근래에는 타겟 이미지를 구성하는 모든 컴포넌트들의 의존성이 통합적으로 관리된다. 또한, 타겟 이미지 생성 도구는 다양한 타겟 보드에 대한 BSP (Board Support Package)를 제공한다. 근래에는 BSP보다 더 상위 수준인 임베디드 시스템의 응용 분야를 선택하여 해당되는 부분을 포함하는 타겟 이미지를 생성할 수 있다. 예를 들면, 개발자가 자동차 분야를 선택하면 이 분야의 임베디드 시스템이 공통적으로 필요로 하는 라이브러리와 유틸리티(예를 들면, CAN 지원 기능)를 포함한 타겟 이미지를 생성한다.

2.2 프로젝트 관리 도구

프로젝트 관리 도구는 빌드를 위한 설정을 관리해 주어, 개발자가 일일이 Makefile을 만들고 유지 보수하는 수고를 덜어준다. 개발자가 일일이 타겟에 맞는 크로스 컴파일러와 링커의 위치를 지정하고 응용에 필요한 라이브러리를 설정할 필요 없이, 타겟만 선택하면 해당하는 크로스컴파일러와 링커를 자동으로 선택할 수 있거나 응용의 종류를 선택하면 필요한 라이브러리도 자동으로 선택된다. 임베디드 리눅스용 응용을 개발할 때에는 공개된 소스 코드를 이용하는 경우가 많은데 이들은 대부분 automake와 autoconf에 의해서 Makefile이 생성되어 있다. 이러한 Makefile을 IDE의 프로젝트 설정으로 개발자가 직접 변환하는 것은 매우 불편하다. 따라서, 프로젝트 관리 도구는 Makefile 기반의 공개 소스 코드를 가능하면 편리하게 프로젝트 설정으로 변환해주는 기능도 필요하다.

임베디드 소프트웨어를 위한 프로젝트 관리 도구는 빌드 설정 외에도 디버깅과 실행에 관련된 설정도 필요하다. 현재 개발 중인 프로그램을 어떤 타겟에 전송하여 실행시킬 것인지와, 타겟에서 프로그램을 실행시킬 때 필요한 옵션, 또는 디버깅할 때의 옵션 등을 설정한다. 이 프로그램의 실행에 필요한 라이브러리나 데이터 파일을 설정하면 프로그램과 함께 타겟으로 전송된다. 개발자는 한 번만 이와 같은 설정을 해두면, 원격 실행 버튼만 클릭함으로써 간단하게 호스트에서 개발 중인 프로그램을 타겟에 전송하고 실행시켜서 테스트해 볼 수 있다. 또한, 원격 디버깅 버튼만 클릭함으로써 타겟으로 프로그램을 전송하고 원격 디버깅을 시작할 수 있다. 임베디드 소프트웨어 개발 시에 원격 실행과 원격 디버깅은 매우 빈번하게 발생하는 작업이므로 개발자가 이를 편리하게 할 수 있어야 한다.

그림 1은 TimeSys사의 임베디드 소프트웨어 IDE인 TimeStorm [6]에 있는 Launch Configurations 윈도우인데, 타겟의 IP 주소, 포트 등의 정보를 설정한 것을 보여준다. 이 외에도 타겟에서 실행시킬 프로그램의 인자와 환경변수, 디버깅 옵션, 타겟에 전송할 파일들을 설정할 수 있다.

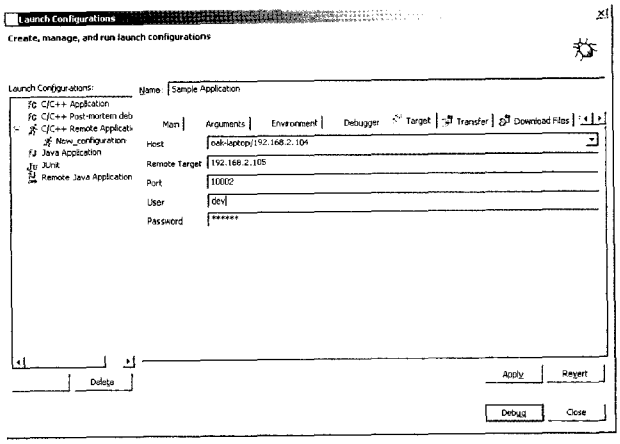

그림 1 TimeStorm의 Launch Configuration 화면

2.3 소스코드 편집 지원 기능

IDE는 소스 코드를 편리하게 네비게이션할 수 있도록 지원한다. 원하는 함수의 선언이나 정의로 빨리 찾아가거나, 클래스를 브라우징하는 것, 함수의 호출 관

계를 보여주는 것 등 소스코드에서 원하는 부분을 빨리 찾아갈 수 있도록 해주는 기능은 기존 IDE에 이미 있는 것들이지만, 임베디드 소프트웨어용 IDE에는 이런 기능들이 부족하였다. 최근 임베디드 소프트웨어용 IDE들은 점차 편리한 소스 네비게이션 기능을 지원하고 있다. IDE는 또한 버전 관리를 지원한다. 편집 환경과 버전 관리가 통합이 되어야 원활한 버전 관리가 될 수 있다. 임베디드 소프트웨어도 점점 복잡하고 커짐에 따라 소스 코드의 버전 관리가 필요하다.

2.4 원격 디버깅 도구

임베디드 소프트웨어 개발이란 응용 프로그램 개발뿐만 아니라 새로운 타겟 하드웨어로 커널 포팅, 커널에 새로운 기능 추가, 커널의 기능 수정, 그리고 디바이스 드라이버 개발도 포함한다. 이를 위하여 커널을 디버깅할 수 있는 디버거가 필요한데, 소프트웨어적인 방법과 하드웨어를 이용하는 방법이 있다. 예를 들어, kgdb는 리눅스 커널을 디버깅할 수 있게 해주는 프로그램으로 소프트웨어로서 구현되어있다.

한편, 커널조차 올바르게 동작하지 않으면 호스트의 디버거와 타겟 간의 통신 채널도 동작하지 않게 되어 원인을 파악하기 어렵다. 이런 경우에는 JTAG/BDM을 지원하는 하드웨어를 이용하여 디버깅한다. 하드웨어를 이용하면, 운영체제가 다운되더라도 어느 위치에서 어떤 상태로 다운되었는지를 조사할 수 있고, 가상 메모리와 물리 메모리의 상태를 알 수 있다. JTAG/BDM 하드웨어를 이용할 경우, 커널 디버깅 외에도 커널을 새로운 하드웨어로 포팅하는 데 필요한 몇 가지 중요한 작업을 할 수 있다. 타겟에 부트로더가 없는 경우에도 플래시 메모리 퓨징(fusing) 및 이미지 다운로딩을 할 수 있고, 부트로더를 포팅하거나 개발 및 디버깅하는데 JTAG/BDM 하드웨어가 이용된다. 대부분의 개발 도구 제공 업체는 JTAG/BDM 하드웨어 솔루션을 제공한다. 예를 들면, TimeSys는 자사의 TimeStorm 통합개발도구와 함께 사용할 수 있는 BDI2000이라는 JTAG 장비를 제공하고 Wind River는 Probe[7]를 제공한다.

2.5 원격 모니터링 및 프로파일링 도구

임베디드 소프트웨어의 경우에는 소프트웨어가 타겟 시스템에서 실행되면서 CPU나 메모리와 같은 자원을 얼마나 소비하고 있으며, 성능 요구 사항을 만족하는지 분석하는 것이 필요하다. 또한, 프로그램이 실행되면서 발생하는 상세한 커널 이벤트의 분석도 필요하다.

타겟 모니터는 타겟에서 실행되는 프로세스나 쓰레드의 리스트, 각 프로세스의 CPU, 메모리 (스택, 힙, 텍스트) 사용량, 타겟 전체의 메모리 사용량 등을 보여준다. 타겟 모니터는 이 정보들을 개발자가 한 눈에 파악하기 용이하도록 그래프로 표현하기도 한다.

타겟 프로파일러는 실시간으로 타겟에서 발생하는 이벤트를 보여주거나, 실행 시에는 이벤트를 로깅만 하고 나중에 이들에 대한 정보를 보여줄 수도 있다. 전자를 실시간 프로파일링이라 하고 후자를 사후 프로파일링이라고 한다. 이미 실행 중인 프로세스나 태스크에 프로파일러를 연결하여 프로파일링을 하기도 한다 (동적 프로파일링). 함수나 소스코드 라인 별로 CPU 사용시간을 측정하기도 하며, 각 태스크나 프로세스가 어떻게 스케줄링되고 있는지도 그래프로 보여준다. 프로파일링을 하는 대상이 너무 많으면 타겟 시스템에 부하를 줄 수도 있고, 정작 원하는 정보를 얻기 어렵기 때문에 이벤트 필터링 기능을 제공한다. 또한 개발자가 새로운 이벤트를 정의하여 시스템 이벤트들과 함께 프로파일링할 수도 있다.

타겟 모니터와 타겟 프로파일러도 통합개발도구 안에 통합되고 있다. 통합된 타겟 모니터는 디버거와 같은 도구와 연동되기도 한다. 예를 들면, 타겟 모니터의 프로세스 리스트에서 하나의 프로세스를 선택한 후 이 프로세스에게 원하는 시그널을 보내거나, 이 프로세스를 디버거에 연결시켜서 동적으로 디버깅을 시작할 수 있다. [그림 2]는 QNX Momentics [8]의 타겟 시스템 모니터 화면을 보여준다.

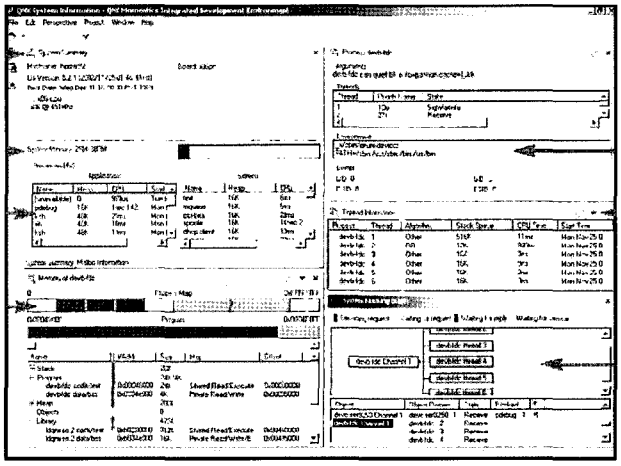

그림 2 QNX Momentics의 타겟 시스템 모니터링 화면

2.6 GUI 빌더

과거에는 특별한 화면 출력이 없는 임베디드 시스템이 주종을 이뤘으나, 최근에는 점차 화려하고 멋진 그래픽을 보여주는 디스플레이를 탑재하고 있는 임베디드 시스템이 많이 나타나고 있다. 요즘 생산되는 핸드폰들은 기본적으로 컬러 LCD를 탑재하고 있고, 320X240 이상의 해상도를 가진 컬러 LCD를 탑재한

PDA의 사용이 늘어나고 있다. 임베디드 GUI 응용 프로그램을 빨리 개발할 수 있도록 임베디드 소프트웨어용 통합개발환경도 GUI 빌더를 제공해야 한다. QNX는 Photon microGUI [9]라는 임베디드 소프트웨어용 그래픽 윈도 시스템을 제공하는데, 이를 위한 GUI 빌더로 PhAB(Photon Application Builder)를 제공한다.

2.7 Elipse 현황

Eclipse 프로젝트의 목표는 다양한 개발 도구들이 쉽게 통합될 수 있는 통합개발환경을 개발하는 것이다. Eclipse는 1999년에 IBM의 연구원들에 의해 개발되기 시작하였는데, 2001년에는 공개 소스 프로젝트가 되어 Eclipse 컨소시엄이 구성되었다. Elipse의 초기 멤버로는 Borland, MERANT, QNX, Rational, Red Hat, SuSE, TogetherSoft가 있고, 그 후에 Fujitsu, Sybase, Oracle, OMG, MontaVista, TimeSys, Intel, Ericsson 등의 유명한 업체와 협회가 참여하고 있다. 한국전자통신연구원도 2002년 8월부터 Eclipse 멤버로 활동하고 있다.

Eclipse가 지향하고 있는 개발 도구의 플랫폼에서는 다음과 같은 개발 환경의 통합이 가능해진다. 지금까지의 개발자는 프로젝트를 수행하면서 다양한 개발 도구를 사용하게 된다. 시스템을 모델링하는데 도구 A를 사용한 후, 이를 구현하기 위하여 Z 라는 프로그래밍 언어를 지원하는 통합개발환경으로 B 제품을 사용하고 있다. 뿐만 아니라, 프로그램의 모니터링과 프로파일링을 하기 위해서 C 라는 도구를, 테스팅을 위하여 D라는 도구를, 소스코드 버전 제어를 위하여 E 라는 도구를 사용한다. 개발을 마친 후에는 문서화를 위하여 도구 F를 사용한다. 이 개발자는 다른 프로젝트에서 Y라는 프로그래밍 언어를 사용해야 하기 때문에 또 다른 통합개발환경으로 G 제품을 사용한다. 이처럼 한 개발자는 하는 일에 따라서 별개의 도구를 사용하는 방법을 익혀야 하고, 이 도구들 간의 데이터가 서로 호환되지 않아서 개발자가 직접 변환해야 하는 경우도 있고, 도구의 장점을 살려서 사용하지 못하는 경우도 있다. 따라서, 이와 같은 다양한 도구들이 하나의 플랫폼에 통합한다면 개발 과정이 보다 자연스럽게 통합될 수 있다는 점이 Eclipse가 등장하게 된 배경이다.

Eclipse 플랫폼은 공개 소스 도구들을 유기적으로 통합하는 플랫폼으로써 개발자에게 확장성과 일관성을 제공한다. 개발자는 원하는 도구를 자신의 Eclipse 플랫폼에 플러그인하면 되므로 새로운 도구를 쉽게 추가하여 확장할 수 있다. Eclipse에 플러그인 되는 도구로 만들 수 있는 응용에도 제한이 없어서 개발자는 다양한 언어(C, C++, Java, Pascal, HTML, UML 등)를 지원하는 도구를 Eclipse에 플러그인하여 사용할 수 있다. 또한, 원하는 회사의 컴파일러나 디버거, 또는 웹에디터를 구입하거나 공개 소스를 가져와서 사용할 수 있다. Eclipse에 플러그인되는 도구는 일관된 모습을 제공하므로 플러그인 된 새로운 도구의 사용 방법을 쉽게 익힐 수 있다. Eclipse는 linux, unix, Windows, Mac OS X 등의 다양한 플랫폼에서 실행된다.

Eclipse는 공개 소스 커뮤니티이다. 따라서, 도구 회사 측면에서는 Eclipse 프로젝트에 참여함으로써 서로 기술을 공유할 수 있으며 자신만의 장점을 살려서 도구에서 유용한 기능 만을 집중하여 발전시킬 수 있다.

Eclipse 기반 임베디드 소프트웨어 통합개발환경으로는 현재 TimeSys의 TimeStorm과 Montavista의 DevRocket, 그리고 QNX의 Momentics가 대표적이다. TimeStorm과 DevRocket은 임베디드 리눅스용 통합개발환경이고, 후자는 QNX의 Neutrino 운영체제용 통합개발환경이다.

3. 임베디드 SW 통합 개발 환경 사례

본 절에서는 본 연구팀에서 개발한 임베디드 소프트웨어 통합 개발 환경인 Esto 와 nano Esto [5], 그리고 QuickDriver에 대하여 설명하겠다. Esto는 임베디드 리눅스 기반의 운영체제인 Qplus 상에서 동작하는 응용 프로그램을 개발하기 위한 환경이며, nano Esto는 센서 네트워크용 운영 체제인 nano Qplus 상에서 동작하는 응용프로그램을 개발하기 위한 환경이다. 그리고, QuickDriver는 하드웨어와 운영체제 및 응용 프로그램간의 연결 고리 역할을 하는 구성요소인 디바이스 드라이버 프로그램을 개발하기 위한 환경이다. 각각의 개발 환경은 유사한 기능도 있고, 모두 Eclipse를 기반으로 개발되었다. 하지만, 각각의 응용 분야에 따라 필요로 하는 기능에 차이가 있기 때문에 파생 도구로 개발되어 왔다. 즉, 단순한 편집기와 프로젝트 관리자 이상의 기능을 제공함으로써 더욱 편리하게 응용 분야의 소프트웨어의 개발에 사용할 수 있는 통합개발환경의 사례를 이들 도구를 통하여 살펴볼 수 있다.

3.1 응용 프로그램 통합개발환경 - Esto/nano Esto

그림 3은 Esto와 nano Esto의 기능을 요약하여 표시한 것이다. 이 도구는 Eclipse 플랫폼 위에서 플러그 인 형태로 개발되어 임베디드 응용 프로그램을

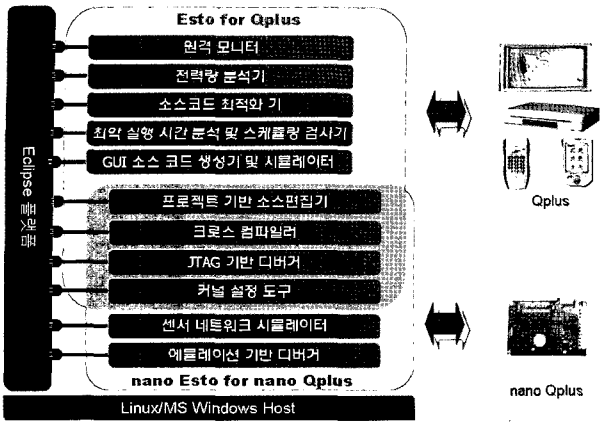

그림 3 Esto와 nano Esto의 기능 요약

개발할 수 있는 환경을 지원한다. 그림 3과 같이 Esto와 nano Esto는 공통적으로 프로젝트 기반의 소스 편집기, 크로스 컴파일러, 원격 실행기, 원격 디버깅과 커널 설정기를 제공한다. 일반적인 임베디드 소프트웨어 통합 환경과 마찬가지로 호스트 시스템에서 소프트웨어를 개발하여, 타겟 시스템에서 개발된 소프트웨어를 적재하여 운영할 수 있는 환경을 제공한다. 특히, 추적점을 이용한 프로그램 수행을 정지시키지 않고, 변수나 레지스터 값의 변화를 모니터링 할 수 있는 기능과 저렴한 비용으로 임베디드 시스템을 하드웨어적으로 디버깅할 수 있는 JTAG 기반 디버깅을 지원한다. 그리고, 임베디드 시스템에 최적화된 운영체제를 조합할 수 있도록 커널을 구성해주는 커널 설정도구인 타겟 빌더가 제공된다.

Esto는 위의 공통 기능 외에도 타겟 시스템의 상태와 시스템 콜과 라이브러리 함수의 호출 정보를 보여주는 원격 모니터링 기능, 프로그램 상에서 전력 소모량을 분석하여 그 결과를 통해 소스코드 최적화 방법을 적용시킬 수 있는 기능, 프로그램의 최악 실행 시간 및 스케쥴링 가능성을 검사하는 기능 등이 지원된다. 그리고, 임베디드용 GUI(Graphical User Interface)를 설계하면 GTK (GIMP ToolKit; Gnu Image Manipulation Program ToolKit) [10] 기반의 소스코드가 생성되는 기능과 작성된 GUI에 대한 호스트 시스템 상에서의 시뮬레이션 환경이 지원된다.

nano Esto는 공통 기능 외에도 실제 타겟 센서 노드 없이 호스트 상에서 디버깅할 수 있는 에뮬레이션 기반 디버깅, 명령어 수준 기반의 정확한 센서 네트워크 환경 시뮬레이션, 수 많은 센서 노드들이 존재하는 환경에서 센서 노드들의 전력 소모량을 분석하는 시뮬레이션 기반의 전력 분석 기능이 제공된다.

이상과 같이 Esto와 nano Esto는 임베디드 소프트웨어를 개발하기 위한 다양한 기능이 통합되어 있는 도구의 한 사례이다.

3.2 디바이스 드라이버 프로그램 통합 개발 환경 - QuickDriver

디바이스 드라이버는 하드웨어와 운영체제 및 응용 프로그램 사이의 연결고리 역할을 하면서, 응용 프로그램이 하드웨어에서 제공하는 기능을 사용할 수 있도록 제어 및 상호 동작을 위한 일관된 인터페이스를 제공하는 소프트웨어이다. 디바이스 드라이버는 하드웨어와 소프트웨어의 양쪽 측면에 모두 관련이 있기 때문에 개발 도구가 필요로 하는 기능에도 차이가 있다. 본 절에서는 디바이스 드라이버 프로그램 통합 개발환경인 QuickDriver에서 제공되는 기능과 일반적인 IDE와의 차이점을 위주로 설명하겠다. 그림 4에 의하면 QuickDriver 는 Eclipse 에 통합된 디바이스 드라이버 개발 환경으로, IDE, 코드 생성, 검증 및 테스팅, 개발 유틸리티 기능이 있다.

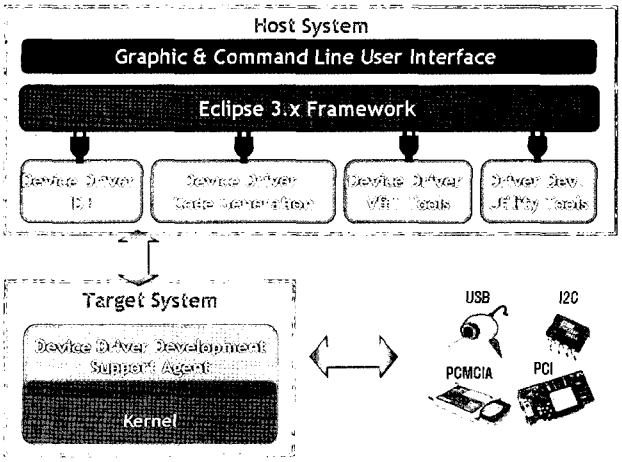

그림 4 QuickDriver의 구조

디바이스 드라이버의 개발이 하드웨어와 밀접한 관계가 있으므로 하드웨어 관련 진단 기능과 유틸리티 기능이 제공되어야 한다. 그림 5와 같이 기본 시스템 리소스에 대한 진단을 통해 드라이버 실행 시 접근할 수 있는 하드웨어 장치의 정보, 유효한 리소스 범위와 상태를 검사하고, 장치에 명령이나 값을 전송할 수 있는 기능인 디바이스 타입별 커널 자원 진단 도구가 있다. 그리고, 그림 6처럼 디바이스드라이버가 장착되는 커널의 API(Application Programming Interface)를 쉘에서 직접 수행시킬 수 있는 기능과 커널에서 발생되는 메시지를 로그를 볼 수 있는 기능, 커널 도움말 기능이 지원된다.

QuickDriver에서는 디바이스 드라이버의 구문이 PCI, USB, PCMCIA등의 버스 타입과 문자형, 블록형, 네트워크형의 디바이스 종류에 따라 유사한 패턴을

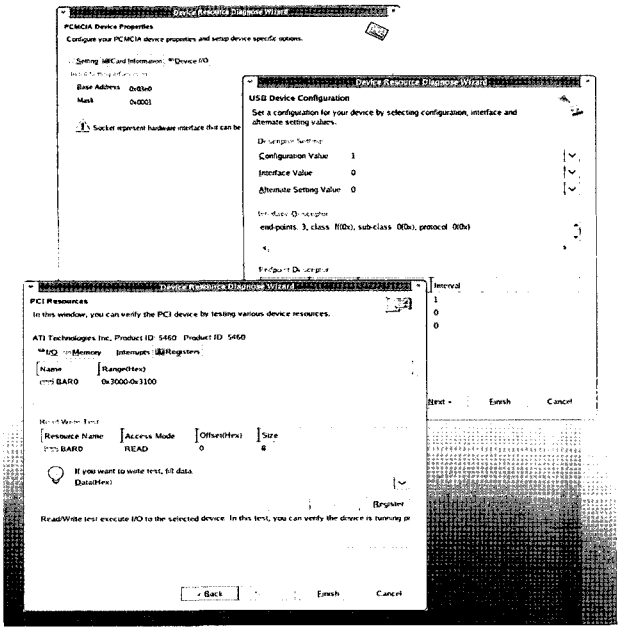

그림 5 디바이스 타입별 커널 자원 진단 도구

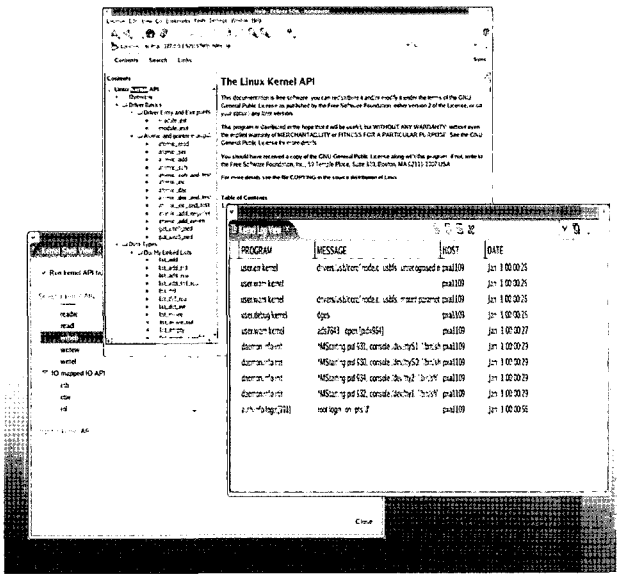

그림 6 커널 API 도움말, 커널 쉘, 커널 메시지 로그 뷰어

가지고 있다는 데에 착안하여, 디바이스 정보와 함께 인터럽트 및 커널 타이머, tasklet, workqueue, shared workqueue 등의 태스크 관련 정보, ioctl 및 파라미터 정보 등의 세부 정보 등을 입력하면, 디바이스 드라이버의 골격 소스를 생성하는 소스 코드 자동 생성 기능이 있다.

또한, QuickDriver에서는 디바이스 드라이버의 품질을 향상시키기 위해서 소프트웨어 테스팅과 정적 검증기능을 지원하고 있다. 테스팅은 임베디드 시스템 개발 환경에 맞는 기능을 제공한다. 즉, 호스트 시스템에서 테스트 코드를 컴파일 하고, 타겟 시스템으로 전송하고, 타겟 시스템에서 테스팅을 수행하고, 마지막으로 테스트 결과를 호스트 시스템으로 전송하여 보여주는 일련의 과정을 자동화되고 통합된 환경에서 수행된다. 이때 테스트 결과는 테스팅의 통과 여부 정보와 함께, 테스팅 시에 소스 코드의 어느 부분이 수행되었는지 하는 정보도 함께 출력된다.

테스팅은 오류를 발견하는 효율적인 방법이지만, 모든 소스 코드를 실행해 보는 것은 실질적으로 불가능하기 때문에 오류가 없다는 것을 보일 수 없는 한계가 있다. 따라서, 소스 코드를 실제 수행하지 않고, 소스 코드를 분석하여 오류가 없음을 수학적인 방법으로 증명하는 정형 검증 기능이 지원된다. 그림 9에 표시된 검증 도구는 검증 대상 소스와 검증 조건을 선택하면 대상 소스가 검증 조건에 맞게 작성되었는지 검증 결과를 출력하는데, 검증 조건에 맞지 않게 작성된 경우에는 오류에 이르는 경로 정보를 함께 출력하여 오류를 수정하는데 도움을 준다.

이외에도 전통적인 임베디드 소프트웨어 통합 개발 도구에서 요구되는 프로젝트 기반 컴파일 환경을 확장하여 디바이스 드라이버 개발을 위한 빌드 환경을 추가로 지원하고 있으며, 리눅스의 KBuild 시스템과 연동되어 디바이스 드라이버를 소스코드 및 기타 관련된 파일을 관리할 수 있는 프로젝트 관리기가 지원되며, 커널 디버거를 통해서 디버깅이 가능하다.

QuickDriver와 다른 상용 도구와 비교를 하면, 기존의 도구들 (Compuware의 DriverStudio [11], Jungo의 WinDriver/Kernel Driver [12,13], Microsoft의 DDK [14])은 일부의 기능만을 지원하고 있으나 QuickDriver는 개발 과정 중에 필요로 하는 다양한 기능을 지원하고 있다 [15].

4. 맺음말

본 원고에서는 임베디드 소프트웨어 개발에 사용되는 도구들의 현 주소를 알아보았다. 현재 우리가 사용하는 임베디드 소프트웨어 통합개발환경이 많은 기능을 제공하고 있지만, 여전히 기능의 확장과 통합이 필요하다. 이러한 소프트웨어 통합 개발 환경이 잘 갖춰져 있어야 개발자들은 반복적이고 소모적인 작업에 노력을 들이지 않고 개발에 집중할 수 있다. 이를 위해서 확장이 용이한 Eclipse와 같은 공통 플랫폼을 사용하면서, 그 위에 필요로 하는 기능을 통합해가는 것도 좋은 대안이 될 것이다.

나아가 임베디드 소프트웨어의 특성에 맞는 시스템 레벨 테스팅 및 검증, 전력 분석 및 최적화 도구 등도 통합개발환경 형태의 도구로 발전되어 갈 것으로 예상

된다. UML (Unified Modeling Language) [16] 등을 지원하는 소프트웨어의 모델링 도구와도 통합되어 개발과정의 전단계를 지원하는 도구가 될 것으로 예상된다.

참고문헌

[1] Eclipse foundation, Eclipse, http://www.eclipse.org
[2] Microsoft, Platform Builder, http://msdn.microsoft.com/embedded/usewinemb/ce/cedevtools/default.aspx
[3] Microsoft, Windows Embedded Studio, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/xpehelp/html/xeconAboutWindowsXPEmbeddedStudio.asp
[4] QNX, System Builder, http://www.qnx.com/products/development/target_tools.html
[5] ETRI, 타겟 빌더, http://www.qplus.or.kr
[6] TimeSys, TimeStorm, http://www.timesys.com
[7] Wind River, Probe, http://www.windriver.com/products/development_tools/
[8] QNX, Momentics, http://www.qnx.com/products/development/
[9] QNX, Graphics solutions, http://www.qnx.com/products/graphics/
[10] GTK, http://www.gtk.org
[11] Compuware, DriverStudio, http://www.compuware.com/products/driverstudio/default.htm
[12] Jungo, WinDriver, http://www.jungo.com/windriver.html
[13] Jungo, KernelDriver, http://www.jungo.com/kerneldriver.html
[14] Microsoft, Windows Driver Development Kit, http://www.microsoft.com/whdc/devtools/ddk/default.mspx
[15] 임채덕 외 5인, "디바이스 드라이버 개발 도구 동향", 전자통신동향분석, 제 21권 1호, 2006년 2월
[16] OMG, UML, http://www.uml.org

임 채 덕

1984. 2~1989. 2 전남대학교 전산학(학사)
1998. 2~2005. 8 충남대학교 전산학 석사, 박사
1989. 2~2006. 6 ETRI 책임연구원 임베디드SW개발도구팀장
2006. 7~2007. 7 UC Irvine 방문연구원 파견
관심분야 : 임베디드 SW, 개발 도구, 실시간 컴퓨팅
E-mail : cdlim@etri.re.kr

김 태 호

1991. 3~1995. 2 성균관대학교 정보공학과 (학사)
1995. 3~2005. 2 KAIST 전산학 석사, 박사
2001. 6~2002. 6 SRI International 방문연구원
2005. 3~현재 ETRI 임베디드SW연구단 선임연구원
관심분야 : 정형 명세 및 검증, 프로그램 정적 분석, 임베디드 소프트웨어
E-mail : taehokim@etri.re.kr