8

임베디드 시스템에서의 오픈소스 활용 현황과 이슈

LG 전자 ▌임 효 준

1. 서 론

디지털 가전기기의 기능이 풍부해지고, 새로운 형태의 제품들이 등장하면서 가전기기에 사용되는 소프트웨어의 중요성이 강조되고 있다. 실제로 DTV, 휴대폰 등 복잡한 기능을 탑재한 가전제품에서는 소프트웨어 개발이 제품 개발의 상당 부분을 차지하고 있다.

서버용 컴퓨터와 데스크톱 컴퓨터 분야보다는 적용시기가 늦었으나, 요즘에는 임베디드 시스템(Embedded System)에도 많은 오픈소스 소프트웨어가 사용되고 있다. 특히 디지털 가전제품의 convergence 경향과 네트워크 기능의 탑재에 따라 앞으로 Linux를 위시한 오픈소스 소프트웨어의 임베디드 시스템 적용은 더욱 증가할 것으로 예상된다. 따라서 이제는 임베디드 시스템 개발자들도 오픈소스 소프트웨어에 대해 잘 이해하고 사용하여야 효율적으로 제품을 개발할 수 있다.

임베디드 시스템은 PC나 서버 컴퓨터와는 다른 여러 특징들이 존재하기 때문에, 일반적으로 오픈소스 소프트웨어를 사용할 때와는 다른 점들을 고려하여야 한다. 또한 임베디드 시스템에서 많이 사용하는 오픈소스 소프트웨어들도 일반 PC나 서버 컴퓨터의 경우와는 다른 경우가 많다.

본 글에서는 먼저 임베디드 시스템에서 오픈소스 소프트웨어를 사용할 때의 고려사항을 살펴본 후 자주 사용하는 오픈소스 소프트웨어 및 개발도구와 오픈 플랫폼에 대해 살펴보도록 하겠다.

2. 임베디드 시스템에서의 오픈소스 사용시 고려사항

2.1 경량화

임베디드 시스템은 재료비를 절약하기 위해 제품 기능을 수용할 수 있는 최소한의 메모리 용량과 최저의 CPU 성능을 가진다. 따라서 PC나 서버에서 사용하던 오픈소스 소프트웨어를 그대로 사용할 경우 메모리가 부족하거나 CPU 연산 능력이 떨어져 제대로 동작하지 않는 경우가 많다.

또한 임베디드 시스템에서는 소프트웨어를 저장하기 위해 보통 flash memory나 ROM 등을 사용하게 된다. 이러한 저장 장치들은 용량이 커질 경우 비용이 올라가기 때문에 제품 동작을 위한 최소 용량을 사용하게 된다. 따라서 오픈소스 소프트웨어도 이러한 적은 용량의 저장장치에 적재될 수 있도록 크기를 줄여야 한다. 어떤 경우는 오픈소스 소프트웨어의 기능 중 일부 기능을 과감히 없애는 일이 필요할 수도 있다.

예를 들어 서버용 컴퓨터에 웹 서버를 설치하는 경우, 가장 널리 사용되는 Apache web server[1]를 사용한다면 큰 문제가 없을 것이다. 그러나 임베디드 시스템에서 Apache web server를 그대로 이식해 사용한다면 대부분의 경우 부족한 메모리 및 CPU 연산 능력으로 인해 제대로 동작하지 않게 된다. 따라서 Apache web server를 경량화하거나 임베디드 시스템에 특화된 boa[2] 등과 같은 다른 오픈소스 소프트웨어의 사용을 고려하여야 한다.

그러나 임베디드 시스템에 특화된 소프트웨어는 필연적으로 일부 기능을 희생시킬 수밖에 없으므로 제품에서 필요한 기능을 해당 소프트웨어가 지원하는지를 확인하는 과정이 필요하다.

2.2 H/W 종속성

잘 만들어진 오픈소스 소프트웨어라면 H/W 종속적인 코드를 최소화하여 어떤 H/W에도 이식이 쉬워야 할 것이다. 그러나 현실은 그렇지 않다. 많은 오픈소스 소프트웨어는 x86 기반의 PC나 서버에서만 테스트가 된 경우가 많아 ARM이나 Mips, PPC 등의 architecture를 가지는 임베디드 시스템에서는 잘 동작하지 않는 경우가 많다.

따라서 오픈소스 소프트웨어를 사용할 때에는 해당 소프트웨어가 적용하고자 하는 H/W에 이식되어 테스트된 적이 있는지를 미리 살펴보는 것이 좋다. 또한 이

식하여야 하는 오픈소스 소프트웨어가 많고 이식의 난이도가 큰 경우에는 뒤에서 소개할 scratchbox 등의 tool 사용을 고려해 보는 것이 좋을 것이다.

2.3 안정성

앞에서 살펴보았듯이 대부분의 오픈소스 소프트웨어는 x86 시스템에서 개발되고 테스트되었다. 따라서 ARM이나 Mips CPU에 이식해 사용하는 경우 x86 시스템에서는 숨어 있던 버그가 발현되는 경우가 있다. 또한 memory와 CPU 연산 능력이 부족하기 때문에 새로이 등장하는 문제들도 있다. 따라서 검증된 오픈소스 소프트웨어라고 하더라도 임베디드 시스템에서 사용하는 경우는 다시 한 번 안정성을 점검해 보는 것이 좋다.

2.4 유지 보수

상용 소프트웨어와 달리 오픈소스 소프트웨어는 제품에 대한 지원을 요청할 곳이 따로 존재하지 않는다. 대신 개발자 커뮤니티를 통해서 지원을 받게 된다. 따라서 오픈소스 소프트웨어를 선정할 때에는 개발자 커뮤니티에서 활발하게 버그 수정 및 기능 개선이 이루어지고 있는지도 잘 살펴볼 필요가 있다.

2.5 오픈소스 라이센스

임베디드 시스템은 일단 시중에 유통이 되고 나면 소프트웨어나 매뉴얼을 수정하는 데에 많은 비용이 들어가게 된다. 또한 라이센스 위반으로 인해 제품 배포가 중지되는 경우 많은 영업 손실이 발생할 수 있으므로 개발 초기부터 오픈소스 라이센스를 고려하는 것이 중요하다. 임베디드 시스템에 오픈소스를 사용할 경우에는 특히 다음과 같은 사항을 유의하여야 할 것이다.

- 제품 매뉴얼에 반드시 오픈소스 관련 안내문을 삽입하여야 한다. 오픈소스 라이센스 위반으로 인한 법정 소송은 결국 매뉴얼에 이러한 안내문이 없는 경우에 발생하는 경우가 많으므로 회사의 제품 개발 체계를 정비하여 반드시 매뉴얼에 사용된 오픈소스들을 명시하고 필요한 경우 소스 코드 제공 방법을 기재하여야 한다.
- 다른 사람들이 노력과 정성을 들여 개발한 오픈소스를 사용하는 임베디드 시스템 개발자들은 추가된 기능에 대해 오픈소스 커뮤니티에 기여할 도덕적 의무가 있다. 그러나 현실적으로는 다른 상용 소프트웨어 업체와의 계약 관계 및 사내 IPR 보호 등의 이유로 보호되어야 할 소프트웨어 코드도 존재한다. 개발 초기부터 소프트웨어 설계를 잘 하여 공개/기여할 코드 부분과 보호할 코드 부분을 분리하여 라이센스로 인한 문제가 생기지 않도록 대비하여야 할 것이다.
- 외주 업체의 소프트웨어를 공급받아 제품에 탑재하는 경우도 최종 법적 책임은 제조사에게 있다. 따라서 외주 업체에서 공급받은 소프트웨어에 대해서도 라이센스 이슈가 없는지 확인하는 과정이 필요하다. 계약서에 이러한 사항을 명시하거나 관련 보고서를 제출하게 하는 등의 방법을 사용할 수 있을 것이다.

이와 같은 오픈소스 라이센스 준수를 위한 제반 작업이 귀찮게 생각될 수 있으나, 개발 초기부터 잘 대응한다면 충분히 준수해 나갈 수 있는 수준이다. 따라서 오픈소스 소프트웨어를 사용할 때에는 반드시 라이센스를 준수한다는 생각을 가지고 대응하여야 할 것이다.

3. 임베디드 시스템에서 많이 사용되는 오픈소스 소프트웨어

본 장에서는 임베디드 시스템 개발 시에 많이 사용되고 있는 오픈소스 소프트웨어들을 살펴 볼 것이다. 임베디드 시스템을 개발하는 업체나 개발자들이 제품에 사용할 오픈소스를 선정하는 데에 도움이 될 것이다.

3.1 리눅스[3]

1991년에 Linus Torvalds에 의해 개발된 리눅스는 이후 많은 기능이 개선되어 2008년 5월 현재 2.6.25 버전이 공개되어 있다.

처음에 리눅스는 PC와 서버용으로 주로 사용되어 왔으나, 최근에는 임베디드 시스템에서도 많이 사용되고 있다. 리눅스가 최근 임베디드 시스템에서 많이 사용되는 이유로는 다음과 같은 것들을 들 수 있다.

- 리눅스는 다양한 장치에 대한 드라이버 소프트웨어를 지원한다. 임베디드 시스템에 새로운 장치를 장착할 경우 device driver 소프트웨어를 개발하여야 하는데, 임베디드 리눅스의 경우는 이미 PC용 Linux에서 개발되어 있는 드라이버 소프트웨어를 재활용할 수 있는 경우가 많아, 개발 노력이 줄어든다.
- 리눅스는 오픈소스 소프트웨어로 사용료가 공짜이므로 칩셋 업체 등에서 새로운 칩의 기능을 테스트하기에 적합한 OS이다.
- 리눅스의 커널 소스는 모두 공개되어 있으므로 제품 동작에 필요한 새로운 기능을 추가하고 cus-

tomize하기에 용이하다.

- 리눅스 커널 위에는 PC 및 서버용 Linux에서 사용하던 많은 오픈소스 소프트웨어들이 있다. 이러한 오픈소스 소프트웨어를 활용할 수 있다는 것이 임베디드 시스템 개발에 큰 장점이 될 수 있다.
- 시중의 대다수 네트워크 장비들은 리눅스를 채택하고 있다. 따라서 리눅스의 네트워크 스택은 다른 어떤 OS보다 안정성과 호환성에서 높다고 볼 수 있다. 앞으로 네트워크 기능의 임베디드 제품 탑재가 늘어나면서 이러한 리눅스의 장점이 더욱 부각될 것이다.

최근에는 이러한 리눅스의 장점 때문에 칩셋 업체들이 리눅스만을 지원하는 경우가 많아 제조사들도 자연스럽게 리눅스를 쓰는 경우가 늘고 있다.

리눅스는 애초에 PC나 서버를 염두에 두고 개발되었다. 그러나 최근에 임베디드 시스템에서의 사용이 늘어나면서 임베디드 기기에서 사용되는 대부분의 CPU architecture를 지원하고 있다. 또한 임베디드 시스템을 위해 다음과 같은 추가 기능들이 이미 리눅스에 탑재되어 있다.

- uCLinux[4]: MMU(Memory Management Unit)가 없는 저사양 CPU에서도 리눅스를 사용할 수 있도록 만든 리눅스의 변종이다.
- Linux-tiny[5]: 임베디드 디바이스에 적용할 때 필요 없는 구성요소를 제거하여 리눅스 커널의 크기를 최소화함으로써 flash memory 및 memory를 절약할 수 있게 해 주는 기능
- rt-preempt[6], O(1) scheduler 등 real-time 기능 개선: 리눅스의 실시간성을 개선하여 실시간성이 요구되는 임베디드 시스템에 사용할 수 있게 해 준다.
- XIP(eXecute In Place): 실행 코드를 메모리에 적재하지 않고 flash memory에서 직접 수행할 수 있게 하여 부팅 시간을 단축하고 memory 사용량을 줄여 준다.
- IDE no probe[7]: IDE device에 대한 probing 과정을 생략하여 부팅 시간을 단축할 수 있게 하여 준다.
- preset-lpj[8]: 시스템 구동시 lpj(Loops Per Jiffy) 값을 계산하는 부분을 제거하여 부팅시간을 단축할 수 있게 해 준다.

3.2 glibc[9], uClibc[10]

Linux kernel 위에서 application을 구동하기 위해서는 기본적인 시스템 library가 필요하다. 대부분의 Linux 시스템은 시스템 library로 GNU glibc를 사용하고 있다. 그러나 일부 임베디드 시스템에서는 glibc보다 더 크기가 작은 경량의 library가 필요한 경우가 있다. 이러한 경우를 위해 glibc에서 많이 쓰지 않는 기능을 제거하고 작은 크기로 만든 library가 바로 uClibc이다. uClibc는 glibc에 비해 크기가 훨씬 작지만 glibc에서 제공하는 일부 API를 제공하지 않으므로 uClibc에서 제공하는 기능만으로 제품 기능 구현에 문제가 없는지 확인 후 사용하여야 한다.

3.3 busybox[11]

Linux를 사용해 임베디드 시스템을 구성하면 시스템의 기본적인 동작을 수행할 수 있는 command-line utility들을 탑재하여야 한다. 예를 들어 쉘 스크립트를 수행하기 위한 쉘, 파일의 생성/삭제/이름 변경을 위한 tool, network 등의 설정을 위한 utility 등이 모두 필요하게 된다. 기존에 PC나 서버에서 사용되던 command-line utility들은 너무 크기가 커서 임베디드 시스템에서 사용하기가 어렵다. 따라서 이러한 command-line utility에서 꼭 필요한 기능만을 추출해 경량화해 만든 것이 바로 busybox이다.

busybox는 컴파일 과정에서 필요한 기능만을 포함하도록 설정이 가능하므로 본인이 원하는 기능만을 포함시켜 작은 크기로 구성할 수 있다. 그림 1은 busybox의 설정 과정을 보여 주고 있다.

3.4 그래픽 라이브러리

PC나 서버용 Linux에서는 GUI application 구동을 위해 일반적으로 X11[12] 그래픽 라이브러리를 사용한다. 그러나 X11은 크기가 너무 커서 일반적으로 임베디드 시스템에 적용하기에는 적합하지 않다. 또한 서

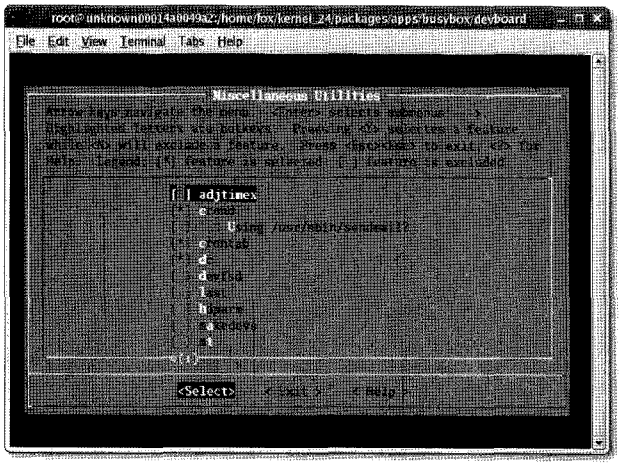


그림 1 Busybox의 설정 과정

버-클라이언트 구조로 인해 임베디드 시스템에서 성능이 크게 떨어질 수 있다. 따라서 다음과 같은 경량화된 그래픽 라이브러리를 사용하게 된다.

- Frame buffer: Graphic hardware 위에 기본적인 API만을 제공하는 디바이스 드라이버를 이식해 사용하는 방법이다. 아주 간단한 GUI가 필요한 저사양 임베디드 시스템에서 주로 사용된다.
- TinyX: X11을 경량화한 것으로 기본적으로 X11에서 지원하는 API를 모두 지원하기 때문에 호환성이 좋다. 그러나 DirectFB나 Nano-X에 비해서는 크기가 큰 편이다.
- DirectFB[13]: 기본 Linux frame buffer 위에 그래픽 가속 기능, 윈도우 관리 기능 등의 임베디드 시스템 개발에 꼭 필요한 필수 기능만을 추가하여 만든 경량의 graphic system이다.
- Nano-X[14]: X11과 유사한 API set을 제공하는 경량 graphic library로서 X11 application을 손쉽게 이식할 수 있게 한다는 목표로 개발된 graphics library이다. 현재는 community에서 유지 보수가 되지 않고 있다는 단점이 있다.

3.5 오픈소스 라이브러리

glibc나 uClibc에서 제공되지 않는 추가적인 기능들을 구현한 오픈소스 라이브러리도 임베디드 시스템에서 많이 사용되고 있다. 임베디드 시스템에서 많이 사용되는 라이브러리로는 다음과 같은 것들이 있다.

- Freetype[15]: Truetype font의 rendering engine
- IJG JPEG library[16]: JPEG 그림 파일의 decoding 기능
- libpng[17]: PNG 형식의 그림 파일 decoding 기능
- cdrtools[18]: CD 미디어에 데이터를 기록하는 기능
- openssl[19]: SSL(Secure Socket Layer)
- iconv[20]: Characterset conversion
- libxml2[21]: XML parsing
- zlib[22]: 데이터의 압축/해제

4. 오픈소스 개발도구

본 장에서는 오픈소스를 사용해 임베디드 제품을 개발할 때에 유용한 오픈소스 개발 도구들을 살펴보도록 하겠다.

4.1 Toolchain

임베디드 시스템 개발을 시작하기 위해 가장 먼저 준비해야 하는 것이 toolchain이다. Toolchain은 C 등의 언어로 작성된 언어를 기계가 이해할 수 있는 형태로 컴파일하고 링크를 하는 데에 필요한 tool들의 모음을 의미한다. 요즘은 대부분 GNU의 gcc compiler와 linker 및 make 등의 tool을 포함하는 GNU toolchain을 사용하고 있다.

임베디드 시스템은 일반적으로 PC에 비해 CPU 성능이 떨어지고 메모리도 충분하지 않다. 따라서 임베디드 시스템에서 직접 컴파일을 수행하게 되면 오랜 시간이 걸리게 된다. 그러므로 임베디드 시스템에서는 PC에서 임베디드 시스템용 실행 파일을 생성할 수 있는 cross-toolchain을 이용해 개발을 하는 것이 일반적이다.

임베디드 시스템은 어떤 CPU와 hardware 설정을 이용해 제품을 구성하느냐에 따라 다른 toolchain을 사용하여야 한다. 특히 시스템이 big endian인지 little endian인지, 그리고 FPU(Floating Point Unit)가 있다고 가정하고 만들어진 toolchain인지 아닌지 등의 요소를 잘 고려하여 toolchain을 준비하여야 한다. 또한 처음에는 잘 동작하는 것처럼 보였던 toolchain이 제품의 완성 단계에서 성능 및 안정성 문제가 있는 것으로 밝혀져 제품 개발 기간을 지연시키는 경우가 많다. 따라서 개발 초기에 toolchain에 대한 안정성/성능을 충분히 테스트해 보아야 제품 개발 기간의 지연을 막을 수 있다.

4.2 Eclipse[23]

Eclipse는 IDE(Integrated Development Environment) 개발을 위한 open platform이다. Eclipse는 Java로 구현이 되어 있으므로, Java 가상 머신이 설치되어 있는 컴퓨터에서는 어디든지 수행이 가능하다는 장점을 가지고 있다. 그림 2는 Eclipse platform의 software architecture를 보여 주고 있다.

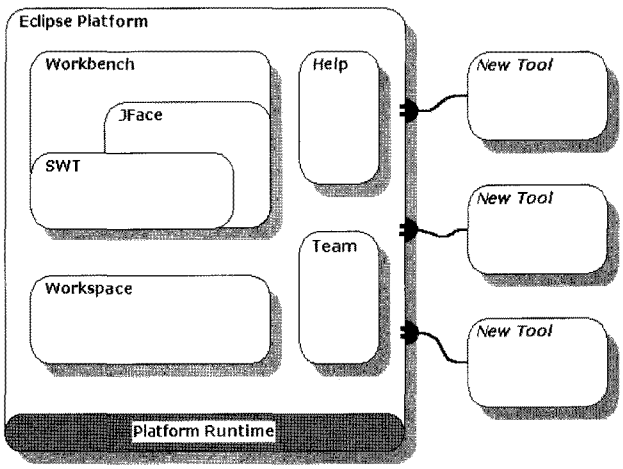


그림 2 Eclipse platform의 software architecture

Eclipse framework는 새로운 기능을 plug-in 형태로 개발하여 손쉽게 추가할 수 있게 되어 있어 최근에 개발되는 많은 IDE tool들은 Eclipse framework을 채택하고 있다. 또한 Eclipse framework에 기반한 1000개 이상의 open source project가 진행되고 있다.

4.3 scratchbox[24]

일반적으로 임베디드 시스템에서 직접 컴파일을 수행할 경우 속도가 느리기 때문에 별도의 기계에서 타겟 디바이스용 컴파일을 대신 수행하는 cross compiler를 사용하게 된다. 그러나 cross compiler를 사용할 경우 실제 디바이스와 컴파일 환경이 동일하지 않기 때문에 오픈소스 소프트웨어를 컴파일하면 잘 동작하지 않는 경우가 많다. scratchbox는 타겟 디바이스의 환경 변수 등 컴파일 환경을 에뮬레이트하여 makefile이나 소스 코드의 수정 없이 오픈소스 소프트웨어를 임베디드 디바이스에 쉽게 이식할 수 있게 하여 준다.

4.4 gdb(GNU debugger)[25]

gdb는 소스 코드 레벨에서 breakpoint를 설정하여 특정 부분까지만 실행을 시키면서 변수의 값을 보거나 조정하면서 디버깅을 수행할 수 있는 single step debugger이다.

임베디드 시스템을 개발할 때에도 gdb를 유용하게 사용할 수 있으나, gdb도 한계가 있음을 인식할 필요가 있다. gdb는 소프트웨어로 동작하는 디버거이기 때문에 시스템에 부하를 주어 디버깅되는 프로그램의 동작을 변경시킬 수도 있다. 따라서 정상적으로 동작하던 프로그램이 gdb로 디버깅할 때에는 문제를 일으키거나, 프로그램에 존재하던 버그가 gdb로 디버깅할 때에는 숨어 버리는 경우가 종종 생긴다. 그러나 단위 모듈의 테스트나 core 파일을 분석하는 용도로 매우 유용하게 사용할 수 있다.

임베디드 시스템에서 gdb를 사용하는 경우에는 시스템에 부하를 많이 주어 제대로 수행이 되지 않는 경우가 많으므로 직접 gdb를 사용하기보다는 gdbserver를 통해 gdb를 사용하는 경우가 대부분이다. 즉, 타겟 보드에는 gdbserver만을 구동시키고 실제 gdb는 PC와 같은 호스트 머신에서 구동함으로써 타겟 보드의 부담을 최소화하는 것이다.

4.5 Profiling tools

임베디드 시스템은 CPU 연산 능력이 충분치 않은 경우가 많아 소프트웨어의 성능을 최적화하는 작업이 필요한 경우가 많다. 소프트웨어에서 가장 많은 연산을 수행하는 부분을 찾아내어 최적화할 수 있게 도와주는 tool이 profiling tool이다. Profiling을 위한 tool로는 gprof[26]와 oprofile[27]이 가장 잘 알려져 있다. gprof는 compile 단계에서 profiling이 가능한 code를 삽입하는 방식이며, oprofile은 주기적으로 수행 중인 코드 지점을 확인하여 대략적인 CPU 사용률을 측정하는 방식으로 동작한다.

4.6 Memory tools

임베디드 소프트웨어를 개발할 때에 가장 잡기 어려운 버그는 아마도 메모리 관련 버그일 것이다. 특히 버그의 발생 지점과 발현 지점이 다른 경우는 정말 디버깅에 어려움을 겪게 된다. 이러한 메모리 관련 버그를 해결하는 데에 도움을 줄 수 있는 다양한 오픈소스 tool들이 존재한다.

- Valgrind[28]: Valgrind는 동적으로 할당한 메모리를 해제하지 않음으로써 생기는 memory leak 문제를 찾아내는 데에 유용한 툴이다. 또한 미리 정의한 메모리 영역을 벗어나 다른 메모리 영역을 침범하게 되는 buffer overrun/underrun 문제를 찾아내는 데에도 유용하다. 재컴파일 과정 없이도 바로 메모리 관련 분석 작업을 할 수 있어 사용이 매우 간편하다. 그러나 overhead가 큰 편이며, ARM이나 Mips에서는 잘 동작하지 않는 경우가 많아 임베디드 시스템에서 널리 사용되는 편은 아니다.
- mpatrol[29]: mpatrol은 memory leak을 찾아내는 데에 유용한 tool이다. 특히 임베디드 시스템에서는 프로그램이 종료되지 않고 제품이 켜져 있는 동안 계속 실행되는 경우가 많다. 이러한 경우 프로그램의 끝 지점에서 해제되지 않은 메모리들을 검출하는 전통적인 기법으로는 memory leak을 찾아내기가 힘들다. mpatrol은 코드 중간 중간에 할당되어 있는 메모리들을 검사하기 때문에 임베디드 시스템에서 유용하게 활용할 수 있다.
- Electric fence[30]: Electric fence는 모든 메모리 할당시 guard page를 함께 할당하여 buffer overrun이나 underrun 발생시 프로그램이 비정상 종료되도록 하여 준다. Electric fence는 buffer overrun/underrun이 발생한 시점을 정확히 찾아 주기 때문에 디버깅이 힘든 메모리 관련 버그 검출에 유용하다.

4.7 Tracing tools

시스템의 동작 상황을 실시간으로 모니터링할 목적으로 사용하는 것이 tracing tool이다. 유용한 오픈

소스 tracing tool로는 다음과 같은 것들이 있다.

- ltrace[31]: ltrace는 system library의 호출을 모두 기록하여 프로그램이 어떠한 library 함수를 사용하여 동작하고 있는지를 알려주는 tool이다.
- strace[32]: strace는 모든 system call 호출을 기록하여 프로그램이 어떠한 system call을 호출하였는지를 알려주는 tool이다.
- LTT(Linux Trace Toolkit)[33]: LTT는 process간의 전환, system call 등의 주요 event를 모두 기록하여 주는 tool이다. LTT는 이렇게 기록된 정보를 시각적으로 보여 주는 graphic tool도 함께 제공한다.

5. 임베디드 오픈 플랫폼

최근에는 오픈소스 소프트웨어를 사용해 플랫폼과 SDK를 개발하여 오픈소스 개발자들이 상위 application을 개발할 수 있도록 하는 오픈 플랫폼을 탑재한 제품들이 나타나고 있다. 본 장에서는 많이 알려진 오픈 플랫폼들을 살펴보도록 하겠다.

5.1 Nokia Maemo[34]

Nokia는 2005년에 WiFi를 통해 web browsing이 가능한 Internet tablet이라는 새로운 개념의 제품을 발표하였다. 첫 번째 Internet tablet 제품인 N770(그림 3)을 위해 Nokia는 Linux 기반의 Maemo라는 오픈 플랫폼을 개발하였고, maemo.org라는 사이트를 통해 개발자들이 새로운 application을 개발할 수 있는 환경을 제공하였다.

Maemo는 기존의 PC Linux에서 많이 사용되던 Gtk widget framework[35]을 수용하고 mobile device에 적합한 Hildon이라는 application framework[36]을 개발하였다. 또 scratchbox 기반의 개발 환경과 emulation 환경을 제공하여 많은 개발자를 유인할 수 있었다. 현재 N770의 후속 모델로 N800 및 N810이 출시되었으며, Maemo 플랫폼 기반으로 수백 개의 application이 개발되거나 이식되어 있다. 그림 4는 Maemo platform의 software architecture를 보여 주고 있다.

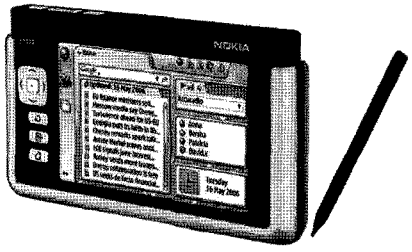


그림 3 Nokia N770

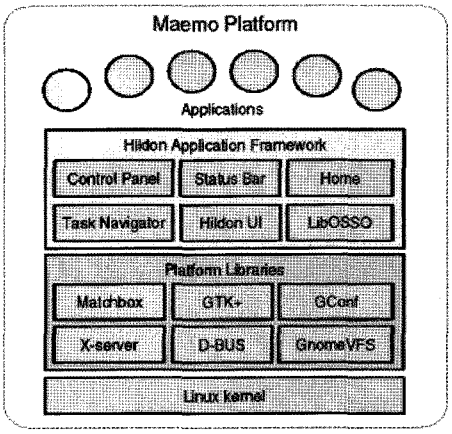


그림 4 Maemo의 software architecture

5.2 Android[37]

Google은 최근에 휴대폰에 적용할 수 있는 Linux 및 Java 기반의 오픈 플랫폼인 Android를 공개하였다. Android는 Linux 위에 Dalvik이라는 최적화된 Java virtual machine을 탑재하여 application들을 Java 기반으로 손쉽게 제작해 이식할 수 있게 해 주는 플랫폼이다. Android는 휴대폰에 필요한 각종 library와 application framework을 모두 제공하고 있어, 상위 application만 개발해 customize하면 휴대폰을 개발할 수 있는 환경을 제공하고 있다. 그림 5는 Android platform의 software architecture를 보여 주고 있다.

Google에서는 오픈소스 커뮤니티의 참여를 유도하기 위해 Android Developer Challenge[38]라는 이름으로 우수 application 개발자들에게 100만 달러의 상금을 주는 프로그램을 진행하고 있다.

Android와 유사한 휴대폰용 오픈 플랫폼으로 OpenMoko[39]라는 것도 있으나 Android만큼의 파급효과를 보여 주지는 못하였다. 또한 LiMo Foundation[40],

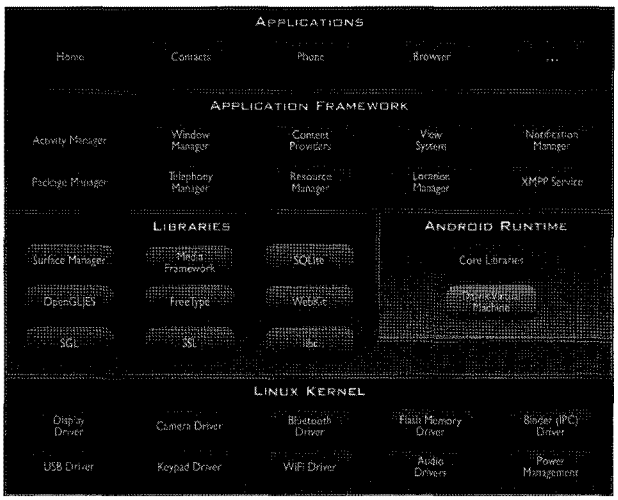


그림 5 Android의 software architecture

Lips Forum[41] 등의 단체에서도 휴대폰을 위한 platform을 제시하고 있으나 아직 개발 과정에 있다.

5.3 Ubuntu Mobile[42]

Ubuntu는 Debian[43] 기반의 Linux 배포판으로 빠른 업데이트와 높은 사용성으로 인해 최근에 많이 사용되고 있다. 이러한 Ubuntu Linux를 mobile device에 적용하기 위한 시도가 바로 Ubuntu Mobile이다. Ubuntu Mobile은 기존의 Desktop용 리눅스에 비해 다음과 같은 특징을 가지고 있다.

- 불필요한 복잡성을 가능하면 제거하여 사용자가 쉽게 사용할 수 있게 함
- 스타일러스 펜 없이 손가락으로 쉽게 제품을 사용할 수 있게 하는 데에 초점을 맞추어 UI를 개발함
- Intel의 MID(Multimedia Internet Device)[44]에 최적화된 기능을 제공
- 다양한 customized application을 개발할 수 있는 환경을 제공한다. HTML, Flash, Clutter, Python/Gtk, C/C++/Gtk, Java 등의 다양한 language와 window system을 사용해 application을 개발할 수 있다.

5.4 Rockbox[45]

Rockbox는 mp3 플레이어를 위한 오픈소스 펌웨어이다. Apple, Cowon, iRiver, Olumpus, SanDisk 등에서 출시한 mp3 플레이어들의 펌웨어를 Rockbox로 교체하면 Rockbox mp3 플레이어가 되는 것이다. Rockbox는 오픈 소스 커뮤니티에서 지속적으로 기능이 개선되고 있으며, 현재 다양한 sound codec, 게임, JPEG, text viewer 등의 기능을 포함하고 있다.

그림 6은 Apple의 iPod에서 rockbox를 구동시킨 예를 보여 주고 있다.

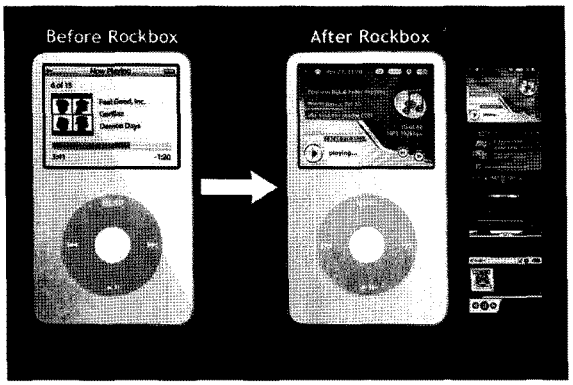


그림 6. Rockbox 구동 예

6. 결 론

이제는 임베디드 시스템에도 많은 오픈소스 소프트웨어가 사용되고 있다. 임베디드 시스템은 PC나 서버 컴퓨터와 다른 여러 가지 특징을 가지고 있기 때문에 그러한 특징을 잘 고려하여 오픈소스 소프트웨어를 현명하게 사용한다면 효율적으로 고품질의 임베디드 기기를 개발할 수 있을 것이다.

앞으로 오픈소스 소프트웨어가 점차 많이 보급되게 되면 임베디드 기기에서도 이러한 오픈소스 소프트웨어의 사용이 늘어나게 될 것이다. 따라서 오픈소스 커뮤니티의 우수한 개발 자원을 잘 활용하고 수많은 오픈소스 소프트웨어를 적재적소에 활용하는 것이 큰 경쟁 요소가 될 것이다. 이러한 점에서 국내의 오픈소스 커뮤니티 참여가 부진한 것은 장기적으로 국내 임베디드 산업에도 위험 요소가 될 수 있다. 현재의 국내 임베디드 산업의 경쟁력을 계속 유지하기 위해서는 많은 젊은 개발자들이 오픈소스 커뮤니티에 적극적으로 참여할 수 있는 문화적 제도적 뒷받침이 필요할 것이다.

참고문헌

[1] The Apache Software Foundation, http://www.apache.org/

[2] Boa Webserver, http://www.boa.org/

[3] The Linux Kernel Archives, http://kernel.org/

[4] Embedded Linux/Microcontroller Project, http://www.uclinux.org/

[5] Linux-Tiny, http://www.selenic.com/linux-tiny/

[6] Morten Mossige, Pradyumna Sampath, and Rachana G Rao, "Evaluation of Linux rt-preempt for embedded industrial devices for Automation and Power Technologies - A Case Study," 9th Real-Time Linux Workshop, Nov. 20, 2007

[7] IDE No Probe, http://elinux.org/IDE_No_Probe

[8] Preset LPJ, http://elinux.org/Preset_LPJ

[9] GNU C Library, http://www.gnu.org/software/libc/

[10] uClibc, http://www.uclibc.org/

[11] BUSYBOX, http://busybox.net/

[12] X.org Foundation, http://www.x.org/

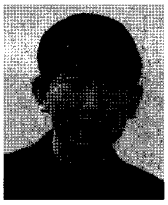
[13] directfb.org | Main, http://www.directfb.org/

[14] Greg Haerr's Nano-X Window System Page, http://www.microwindows.org/

[15] The FreeType Project, http://www.freetype.org/

[16] Independent JPEG Group, http://www.ijg.org/

[17] libpng, http://www.libpng.org/pub/png/libpng.html
[18] Cdrtools release information, http://cdrecord.berlios.de/
[19] OpenSSL, http://www.openssl.org
[20] libiconv, http://www.gnu.org/software/libiconv/
[21] The XML C parser and toolkit of Gnome, http://xmlsoft.org/
[22] zlib Home Site, http://www.zlib.net/
[23] Eclipse.org home, http://www.eclipse.org/
[24] Scratchbox, http://www.scratchbox.org/
[25] GDB: The GNU Project Debugger, http://www.gnu.org/software/gdb/
[26] GNU gprof, http://www.cs.utah.edu/dept/old/texinfo/as/gprof_toc.html
[27] OProfile, http://oprofile.sourceforge.net/news/
[28] Valgrind, http://valgrind.org/
[29] mpatrol, http://www.cbmamiga.demon.co.uk/mpatrol/
[30] Electric Fence, http://directory.fsf.org/project/Electric-Fence/
[31] Dynamic Library Call Tracer, http://sourceforge.net/projects/ltrace/
[32] strace, http://sourceforge.net/projects/strace/
[33] LTTng & LTTV homepage, http://ltt.polymtl.ca/
[34] maemo.org, http://maemo.org/
[35] The GTK+ Project, http://www.gtk.org/
[36] Hildon, http://live.gnome.org/Hildon
[37] Android - An Open Handset Alliance Project, http://code.google.com/android/
[38] Android Developer Challenge, http://code.google.com/android/adc.html
[39] OpenMoko: One More Key, http://www.openmoko.com/
[40] LiMo Foundation, http://www.limofoundation.org/
[41] Lips Forum, http://www.lipsforum.org/
[42] Ubuntu Mobile, http://www.ubuntu.com/products/mobile
[43] Debian, http://www.debian.org/
[44] Mobile Internet Device & Ultra Mobile PC, http://www.intel.com/products/mid/
[45] Rockbox - Open Source Jukebox Firmware, http://www.rockbox.org/



임 효 준

1992~1996 서울대학교 전산과학과(학부)
1996~1998 서울대학교 컴퓨터공학부(석사)
1998~2001 서울대학교 컴퓨터공학부(박사)
2001~2003 한국 채권연구원
2003~현재 LG 전자 책임연구원
E-mail : imhyo@lge.com