특집

임베디드 소프트웨어 플랫폼 표준화 동향

김 두 현* 김 정 국**

목 차

1. 서 론

임베디드 소프트웨어는 PC용 소프트웨어와 달리 실시간성, 고신뢰성 및 저전력을 요구하는 산업용·군사용 제어기기, 디지털 정보기기, 자동 입출력 센서 장비 등의 마이크로프로세서 위에 탑재되는 소프트웨어로 예를 들어, 인터넷 셋탑박스에 내장된 인터넷 접속 기능, 멀티미디어 처리 기능, 전자상거래 기능 등을 제공하는 소프트웨어가 이에 해당한다.

임베디드 소프트웨어는 PC라는 비교적 넉넉한 환경에서 수행되는 소프트웨어와는 달리 제약된 환경 하에서 시스템이 요구하는 기능만을 최적으로 제공해야하는 것을 대전제로 하고 있다. 이러한 전제하에서 본다면 임베디드 소프트웨어는 다음과 같은 특징을 갖고 있다.

- 임베디드 소프트웨어가 실행되는 시스템의 용도에 따라 연성 혹은 경성 실시간 처리를 지원하여야 한다. 예를 들어, 무인항공기에 사용되는 비행제어시스템, 항법시스템 등에 내장되는 임베디드 소프트웨어는 무인항공기 제어에 허용되는 시간 내에 작업을 처리하여야 한다.
- 소프트웨어의 오동작 및 작동 중지가 허용되지 않는 임베디드 시스템에서 고도의 신뢰성을 제공하여야 한다. 예를 들어, 원자력 발전, 항공기 제어, 미사일 등과 같이 mission-critical한 임베디드 시스템에서는 소프트웨어의 오동작 또는 불시의 작동 중지 등은 심각한 결과를 초래하게 될 것이다.
- 임베디드 시스템은 크기, 가격 및 발열 등을 이유로 제한된 하드웨어 자원으로 구성되므로 임베디드 소프트웨어는 경량화, 저전력 지원, 자원의 효율적 관리 등의 하드웨어에 최적화되는 기술을 지원하여야 한다.
- 시장을 지배하기 위해서는 다양한 기종과 규격의 마이크로프로세서에 최적화된 별도의 솔루션이 동시에 제공되어야하며, 고난도의 임베디드 소프트웨어 응용을 빠르고 안정되게 개발하기 위해 개발 도구가 필수적이다.

이상과 같은 특징은 PC 환경에서의 소프트웨어 개발자가 자신이 개발한 프로그램을 임의의 임베디드 시스템에 손쉽게 올릴 수는 없다는 점을 암시하는 것으로 이를 위해서는 타켓팅하는 임베디드

* 건국대학교 인터넷미디어공학부 조교수
** 한국외국어대학교 컴퓨터 및 정보통신공학부 교수

시스템에서 수행되고 있는 운영체제 하에서 이와 같은 특징의 기능을 구현하여야 할 것이다. 그러나 이러한 기능을 개발자 각자가 알아서 구현한다면 그 개발자에게는 매우 어려운 작업일 뿐만 아니라 유사한 프로그램 간에는 매우 중복적인 일이 될 것이다. 또한 얼마 후 타 시스템에 이를 탑재하려 할 경우 그 시스템의 운영체제 환경이 변화되면 많은 것을 새로 시작해야하는 불행을 겪을 수도 있을 것이다.

이와 같은 이유에서 임베디드 소프트웨어 개발자 그룹들과 산업체에서는 일종의 계층구조로 볼 수 있는 API(Application Program Interface)의 추상화를 통하여 하부계층의 구현 내용이 상이하더라도 상위 계층의 구현물은 영향을 받지 않도록 함으로써 이식성과 호환성을 최대한 보장해야한다는 의식이 싹트게 되었고, 이를 시발점으로 하여 최근 들어서는 임베디드 시스템의 산업분야 즉, 가전분야를 비롯하여 이동통신, 텔레매틱스 등 다양한 제품군에 공통으로 적용되는 이른바 플랫폼 표준을 제정하는 단체 활동이 구체화되기에 이르렀다.

사실 이와 같은 플랫폼 표준은 이미 임베디드 소프트웨어가 주목을 받기 이전부터 ISO/IEC, IEEE가 주축이 되어 만든 POSIX(Portable Operating System Interface)[1] 표준안이 존재하였으며 지금도 이것은 Linux, UNIX, MS Windows에 많은 영향을 미치고 있는 것이 사실이다. 그러나 임베디드 소프트웨어에 있어서는 POSIX를 수용하면서도 앞에서 설명한 바와 같은 특별한 기능들을 플랫폼 차원에서 제공하여야 하므로, 산업체와 학계에서는 임베디드 소프트웨어로 특징지을 수 있는 플랫폼 표준을 별도로 제정하려는 시도가 지속되고 있다.

본 고에서는 이러한 연유에서 출발한 국내외 유력 단체인 KESIC, CELF, TRON, Eclipse 등의 활동 상황과 표준안의 특징을 짚어 봄으로써 임베디드 소프트웨어의 기술 개발 및 산업화에 있어서 플랫폼 표준과의 접점에 대한 이해를 높여보고자 한다.

2. 플랫폼의 정의

우선 소프트웨어 구조상에서 어느 범주까지를 플랫폼이라 할 수 있는지를 정의하여야 할 것이다. 물론 여기서 말하는 플랫폼이란 하드웨어가 아닌 소프트웨어 플랫폼을 이야기하는 것은 당연할 것이지만, 그 이상으로 구체적으로 정의하는 것은 매우 다양한 관점들이 교차하고 있기 때문에 쉽게 결론을 내리기는 어려운 점이 있다. 그러나 본 고를 전개해 나가기 위하여 굳이 그 범주를 정해본다면 플랫폼이란 운영체제와 미들웨어 그리고 개발도구 등을 포함하는 하드웨어와 응용 프로그램 사이의 영역에 존재하는 기반 소프트웨어라 볼 수 있겠다.

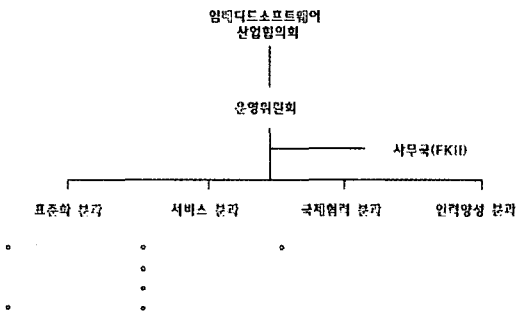
즉, 기존의 POSIX[1]는 단순히 시스템 콜들을 표준화했다면 플랫폼 표준은 시스템 콜의 확장 또는 축소와 아울러 각종 응용분야에 공통적으로 사용되는 핵심 기능들을 모아서 제공하는 미들웨어와 그의 API도 포함한다. 이러한 미들웨어는 각종 응용분야에 맞추어 관련 산업체가 새롭게 정의해야 한다는 점이 특이하다. 예를 들어 디지털 홈 서비스에 핵심인 홈 서버나 홈 게이트웨이에 탑재하는 OSGi[2] 및 각종 가전기기 제어 프로토콜인 HAVi[3], UPnP[4]는 홈 네트워크 분야에서 매우 필수적인 미들웨어가 될 것이다. 그러나 텔레매틱스 분야에서는 이와 달리 MOST(Media Oriented Systems Transport)나 AMI-C(Automotive Multimedia Interface Collaboration) 등의 API가 미들웨어 표준이 될 수 있을 것이다. 이러한 개념은 한국전자통신연구원에서 발표한 참고문헌 [5]을 참조하면 쉽게 이해될 수 있을 것이다.

3. 국내 표준화 동향

우리나라 임베디드 소프트웨어 플랫폼 표준화 활동은 지난 2003년 3월에 창립된 한국임베디드소

프트웨어산업협의회(KESIC, Korea Embedded Software Industry Council, http://www.kesic.or.kr)을 중심으로 이루어지고 있다. 특히 KESIC은 최근들어 한국오픈셋탑포럼(KOSF, Korea Open Set-top Forum)의 활동을 사실상 겸하면서 이의 결과물을 KESIC 표준안의 Best Practice로 만들기 위하여 구체적으로 노력 중에 있다.

KESIC은 현재 삼성전자, LG전자, 휴맥스 등 150여 회원사가 참여하고 있으며, KESIC의 조직은 (그림 1)의 조직도에서 보는 바와 같이 현재 4개의 분과로 이루어져 있다. 특히 본 고에서 관심을 갖고 있는 2004년 4월경에 발표된 KELPS(KESIC Embedded Linux Platform Specification) Ver. 1.0[6]은 표준화 분과 WG1 활동의 결과다.



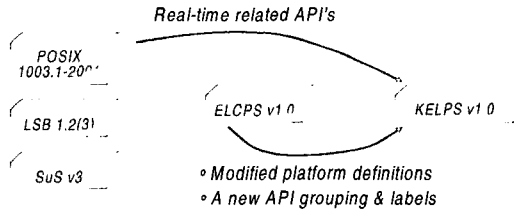
(그림 1) KESIC의 조직도(출처: http://www.kesic.or.kr)

<표 1> KELPS Ver. 1.0의 근간이 된 표준안

ISOC99	ISO/IEC 9899:1999, Programming Languages - C	
LSB1.3	Linux Standard Base	http://www.linuxbase.org/spec/
POSIX.1-2001	IEEE POSIX 1003.1-2001	http://www.ieee.org
SUSV3	Open Group Single UNIX Specification version 3	http://www.opengroup.org
ELCPS v1.0	Embedded Linux Consortium Platform Specification version 1.0	http://www.embedded-linux.org

KELPS Ver. 1.0은 <표 1>에 요약된 바와 같이 Embedded Linux Consortium의 ELCPSv1.0을 기반으로 이를 국내 산업의 요구를 반영하여 KESIC이 개정한 임베디드 리눅스 플랫폼 명세서로서 Linux Standards Base 1.3[7], IEEE POSIX 1003.1-2001 specification[1](IEEE POSIX 1996 버전을 대체하는 것으로서 Real-Time, 스레드, 네트워킹 등에 관한 update 사항을 포함), Single UNIX Specification V3.[8](UNIX98 standard를 대체하고, IEEE POSIX 1003.1-2001과 연동하도록 설계), ELCPS V1.0.[9] 등을 기반으로 작성되었다.

이러한 타 표준안을 기반으로 새로운 한국형 표준안인 KELPS V1.0을 제정하는 데에 있어서 근간이 된 개념은 (그림 2)와 같다. 즉 ELCPS V1.0에 반영되지 않았던 POSIX 1003.1-2001의 실시간 관련 API를 포함하면서 ELCPS V1.0의 플랫폼 타입을 수용하여 다양한 응용을 흡수할 수 있도록 하고 아울러 이에 알맞게 API 그룹핑을 재정립하였다.



(그림 2) KELPS V1.0과 타 표준안과의 관계도

ELCPS V1.0과 같이 가장 큰 특징이라 할 수 있는 KELPS V1.0의 플랫폼타입은 시스템 환경 집합을 정의하는 것으로 "minimal" 환경으로 시작하여, 이에 점진적으로 기능 그룹을 추가한 환경인 "intermediate"환경과 "full" 환경으로 구성된다. 이러한 환경의 설정은 ELCPS V1.0 표준과 IEEE POSIX 1003.13의 정의를 기반으로, 현 국내 임베디드 산업의 실정을 반영하여 설정한 것이다. 다음은 플랫폼타입에 대하여 KELPS V1.0 원문에 있

는 내용을 가능한 충실히 인용한 내용이다.

3.1 최소 시스템 환경(Minimal System Environment)

임베디드 되거나 네트워크로부터 분리되어 있는 한 개 또는 그 이상의 특수기기의 동작 환경을 말한다. Minimal 환경은 위와 같은 기기의 목적에 필요한 최소한의 환경을 말하며, 이러한 환경에서 동작하는 기기들은 사용자와의 대화 기능, 다른 기기와의 네트워크를 통한 협력 기능, 소규모 매체 기반의 파일 시스템 기능, 실시간 처리 기능 등을 선택적으로 제공할 수 있다. 이러한 환경에서 기기의 응용은 일반적으로 하나의 주소 공간을 사용하는 다중 리눅스 태스크 또는 POSIX 스레드들로 구성된다. (POSIX의 fork() API는 제공되지 않는다.)

3.2 중간 단위 시스템 환경 (Intermediate System Environment)

최소 시스템 환경에, 대용량 기억장치를 위한 기능 (파일, 파일 시스템, Linux Large File Support), 비동기(non-blocking) 입출력, 객체나 라이브러리의 동적 링킹 기능들을 추가로 제공하는 환경이다. 또한 이 환경은 다중 프로세스 및 다중 주소 공간을 위한 기능이 가능한 환경이다. 이 환경은 하드웨어적으로 실질적인 대용량 저장 장치를 가정하지는 않지만, RAM, ROM, Flash 메모리 기반의 파일 시스템과, 다중 프로세서를 사용하는 하드웨어적 환경까지도 그 기반으로 한다. 이 환경은 최소 시스템 환경에, 대용량 기억장치를 위한 기능 (파일, 파일 시스템, Linux Large File Support), 비동기(non-blocking) 입출력, 객체나 라이브러리의 동적 링킹 기능들을 추가로 제공하는 환경이며, 다중 프로세스 및 다중 주소 공간을 위한 기능이 가능한 환경이다.

3.3 최대 시스템 환경 (Full System Environment)

최소 및 중간 단위 환경에 추가하여 다목적 Linux 환경의 모든 기능을 포함하는 환경이다. 이 환경은 시스템 유틸리티들에 대한 기술이 없는 점을 제외하면(POSIX shell 은 popen()과 같은 함수를 통하여 최대 시스템 환경에 규정되어 있다.) LSB 1.3 시스템과 동일하다. 하드웨어 모델은 하나 또는 그 이상의 처리장치와 메모리 대용량 저장장치 네트워크 지원, 그리고 사용자 인터페이스/디스플레이 장치를 포함한다. 기본적으로 이 환경은 시스템 유틸리티들에 대한 기술이 없는 점을 제외하면(단 POSIX shell에 대한 기능 정의의 popen()과 같은 함수들을 통해 정의 되었다.) LSB 1.3 시스템과 동일하다.

이와 같은 환경에 대한 정의와 아울러 <표 2>에서 보는 바와 같이 KEL_ASYNCHRONOUS_IO 등 48개의 API 그룹이 정의되어 있으며, 각 그룹에마다 위의 3가지 시스템 환경별로 필수요구 사항이냐, 확장 시 우선적으로 고려할 사항이냐에 따라 R(Required)와 PE(Primary Extension)를 부여하였고, 아무것도 부여되지 않은 것은 옵션으로 명시하였다. 즉, 예를 들어 KEL_FILE_SYSTEM은 minimal에서는 PE, intermediate과 full에서는 R로 정하고 있다. 또한, ELCPS V1.0에 비하여 새로 추가된 KEL_THREADS와 KEL_THREADS_REALTIME은 모든 타입에서 R인 반면, KEL_THREADS_EXT는 minimal에서는 옵션으로, KEL_THREADS_REALTIME_EXT은 Minimal과 Intermediate에서 PE로 정의하고 있다. 이상은 KELPS V1.0의 원문을 참조한 것으로 보다 자세한 내용은 참조문헌 [4]를 참조하기 바란다.

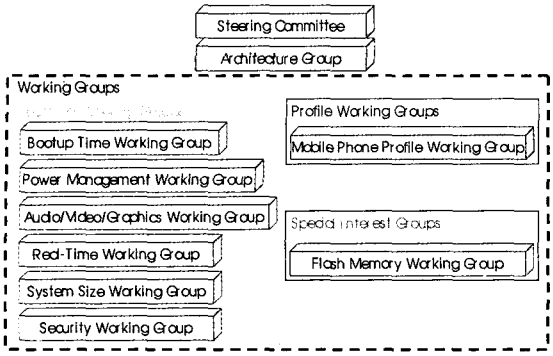
4. 국외 표준화 동향

4.1 CELF(Consumer Electronics Linux Forum)

CELF(http://www.celinuxforum.org)[10]는 풀어쓴 이름의 의미에서 알 수 있듯이 가전제품

<표 2> KELPS의 API 그룹과 플랫폼 타입별 지원 정도

API 그룹	Min.	Inter.	Full	내용
KEL_ASYNCHRONOUS_IO	PE	R	R	비동기 입출력 인터페이스
KEL_C_LANG_JUMP		R	R	ISO C 라이브러리 점프 인터페이스 함수
KEL_C_LANG_MATH		PE	R	수학 함수 인터페이스 그룹을 지원
KEL_C_LANG_SUPPORT	R	R	R	ISO C 라이브러리 인터페이스 그룹
KEL_C_LANG_SUPPORT_R	R	R	R	다중 쓰레드에 안전한(thread_safe) 일반 ISO C 라이브러리 인터페이스
KEL_C_LIB_EXT		R	R	일반 C 라이브러리 확장 인터페이스
KEL_DEVICE_IO	PE	R	R	장치 입 출력 인터페이스
KEL_DEVICE_SPECIFIC			R	일반 단말기 인터페이스
KEL_DEVICE_SPECIFIC_R			R	쓰레드-안전 일반 단말기 인터페이스
KEL_DYNAMIC_LINKING		R	R	동적 링킹 인터페이스
KEL_FD_MGMT		R	R	파일 표현자 관리 인터페이스
KEL_FIFO			R	FIFO 인터페이스
KEL_FILE_ATTRIBUTES			R	파일 속성 인터페이스
KEL_FILE_SYSTEM	PE	R	R	파일 시스템 인터페이스
KEL_FILE_SYSTEM_EXT			R	파일 시스템 확장 인터페이스
KEL_FILE_SYSTEM_R	PE	R	R	쓰레드-안전 파일 시스템 인터페이스
KEL_IPC		R	R	프로세스간 통신 인터페이스
KEL_JOB_CONTROL			R	작업 컨트롤 인터페이스
KEL_JUMP		R	R	확장 점프 인터페이스
KEL_LARGE_FILE		R	R	큰 파일 인터페이스
KEL_MEM_MGMT	PE	R	R	메모리 관리 인터페이스
KEL_MULTI_ADDR_SPACE		R	R	다중 주소 공간 인터페이스
KEL_MULTI_PROCESS		R	R	다중 처리 인터페이스
KEL_NETWORKING	PE	PE	R	네트워킹 인터페이스
KEL_NETWORKING_RPC			R	RPC 인터페이스
KEL_PIPE		R	R	파이프 인터페이스
KEL_REGEXP			R	정식 표현된 인터페이스
KEL_SCHEDULING	R	R	R	실시간 스케줄링 인터페이스
KEL_SEMAPHORES	R	R	R	세마포 인터페이스
KEL_SHELL_FUNC			R	쉘 과 유틸리티 인터페이스
KEL_SIGNALS	R	R	R	신호 인터페이스
KEL_SIGNAL_JUMP		R	R	단일 점프 인터페이스
KEL_SINGLE_PROCESS	R	R	R	단일 처리 인터페이스
KEL_SPAWN		PE	R	spawn 인터페이스
KEL_STDIO_LOCKING	R	R	R	쓰레드-안전 Stdio 잠금 인터페이스
KEL_SYMBOLIC_LINKS			R	심볼릭 링크 인터페이스
KEL_SYSTEM_DATABASE			R	시스템 데이터베이스 인터페이스
KEL_SYSTEM_DATABASE_R			R	다중 쓰래드에 안전한(Thread-safe) 시스템 데이터베이스 인터페이스
KEL_SYSTEM_LOGGING		PE	R	시스템 로깅 인터페이스
KEL_THREADS	R	R	R	POSIX 쓰레드 인터페이스
KEL_THREADS_EXT		R	R	확장된 POSIX 쓰레드 인터페이스
KEL_THREADS_REALTIME	R	R	R	실시간 스케줄링 POSIX 쓰레드 인터페이스
KEL_THREADS_REALTIME_EXT	PE	PE	R	실시간 확장된 POSIX 쓰레드 인터페이스
KEL_TIMERS	R	R	R	실시간 클럭과 타이머 인터페이스
KEL_USER_GROUPS			R	사용자와 그룹 인터페이스
KEL_USER_GROUPS_R			R	쓰레드-안전 사용자와 그룹 인터페이스
KEL_WIDE_CHAR		R	R	확장 문자 라이브러리 인터페이스
KEL_WIDE_CHAR_DEVICE_IO		R	R	확장 문자 장치 I/O 인터페이스



(그림 3) CELF 조직도

(Consumer Electronics)을 위하여 특화된 임베디드 리눅스 기반의 플랫폼을 제정하는 것이 목적이다. 가전제품이라는 도메인이라는 것은 일단 학구적이기 보다는 상용 제품에 적용하는 것을 전제로 하고 있다. 이는 CELF가 2003년 6월 일본의 대표적인 가전업체를 포함하는 소니, 파나소닉(마쓰시다전기), 히다찌, NEC, 필립스, 삼성전자, 샤프, 도시바 등 8개 업체가 손을 잡고 설립한 것을 보아서도 알 수 있다. 현재 우리나라의 주요 가전업체와 ETRI를 포함하여 전 세계 50여개 이상의 가전 전문 업체들이 멤버로 가입되어 있다.

CELF는 임베디드 리눅스 기반의 가전제품 플랫폼 표준을 만드는 것 뿐만 아니라 임베디드 리눅스 기술의 지속적인 개선과 보다 많은 활용을 촉진시키는 목적도 함께 갖고 있다. 이러한 목적과 궤도를 같이하여 그 성격 또한 ELC나 KESIC이 표준 제정에 무게를 두고 있는 것과는 달리 스펙 자체보다는 가능하다면 존재하는 오픈 소스 프로젝트 인프라로 이용하여 가전제품에 필요한 공통 기술 또는 솔루션을 구현하는 것에 중심을 두고 있다.

이러한 활동의 이면에는 GPL 중심의 공개소스를 개선하여 가전제품에 적용할 경우 그 개선부분을 다시 공개해야 하는 것이 기정사실라면 이와 같이 공개될 부분은 서로 협력하여 필요한 기술을 신속하게 개발하여 공유하고, 이를 응용하여 경쟁력 있는 제품을 개발하는 것은 회원사들 간의 자유 경쟁에 맡긴다는, 즉, 협력 속의 경쟁이라는 철학이 담겨있는 것으로 해석된다.

이와 같은 배경을 반영하듯 포럼의 조직도 (그림 3)에서 보는 바와 같이 Steering Committee와 Architecture Group이 전부이고, Architecture Group 속에 임베디드 리눅스가 진정으로 가전용 솔루션이 되기 위한 핵심 기술을 논의하고 구현하는 목적의 워킹그룹들이 설치되어 있다. 〈표 3〉은 참조문헌 [10]를 통하여 파악할 수 있는 각 워킹그룹의 활동 내용을 요약한 것이다. 현재 CELF의 스펙과 이에 상응하는 구현물이 공개되어 있다[10].

4.2 T-Engine 포럼 (http://www.t-engine.org, http://www.t-engine.co.kr)

이제까지 소개한 ELC, KESIC, CELF 등은 모두 공개소스인 임베디드 리눅스를 기반으로 하는 반면 T-엔진 포럼은 일본 동경대의 사카무라 켄 교수가 주축이 되어 1983년부터 추진되어 온 TRON(The Real-time Operating system Nucleus) 프로젝트 중 현재 임베디드 시스템을 위한 일본의 사실상 de facto인 μITRON이 모체가 출범된 포럼이다.

T-Engine은 T-Engine 하드웨어, 리얼타임 OS T-Kernel, 디버그 모니터 T-Monitor로 구성되고, 개발 환경이나 디바이스 드라이버나 미들웨어의 유통 기반 등도 포함한, 개방형 표준 개발 플랫폼으로 〈표 4〉와 같이 4종류의 T-Engine 플랫폼으로 나뉘어진다.

현재 T-Engine은 2004년 8월 현재 Macromedia사는 T-Engine에 Flash를, SUN은 Java를, Montavista는 Linux를, MS는 WinCE를 내장하는 연구를 하는 등 글로벌 S/W 기업들의 참여하고 있고, 유비쿼터스 사회 건설과 맥을 같이하여 응용

<표 3> CELF 워킹 그룹별 목적과 주요 활동 내용

WG명	목적	스펙 활동 내용
Bootup Time WG	- To Reduce the boot time of the Linux kernel(Cold-start in 1.5 sec, Kernel start in 0.5 sec)	- Calibrate_Delay() Avoidance - Supporting IDE NoProbe - Kernel Execute-In-Place (XIP)
Power Management WG	- To allow CE device manufactures to optimize the power use of their products	- platform suspend/resume - device power management - platform dynamic power management - variable scheduling timeouts
Audio/Video/Graphics WG	- To Meet AVG requirements for CE devices	- No single default/standard interfaces exist for AVG
Real-time WG	- To improve the real-time capabilities of the Linux to address the range of real-time requirements for CE devices	- Preemptible kernel - O(1) scheduler - Interrupt thread - Sift-IRQ threads - POSIX timer - POSIX message queues - Priority inheritance on user mutexes
System Size WG	- To minimize memory usage	- Kernel XIP - Compress FS(initrd) - Compress FS(Cramfs) - Compress FS(jffs2)
Security WG	- To make CE products more secure, both from the consumer's and the manufacturer's point of view	- Data preservation despite kernel bugs - Protection from malicious attacks - Protection from stack overrun attacks - Preservation of program code integrity - Network security Interfaces to security hardware - Investigate kernel interface to support security services

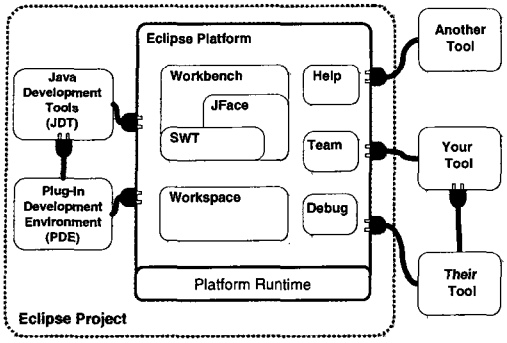
<표 4> T-Engine 플랫폼의 종류와 차이점

플랫폼 명	차이점
표준 T-Engine(표준 티·엔진)	휴대 정보 기기용의 비교적 고도의 유저 인터페이스를 가지는 기기를 위한 플랫폼
μT-Engine(마이크로·티·엔진)	가전 기기나 계장기기 등 비교적 유저 인터페이스가 적은 기기를 위한 플랫폼
nT-Engine(나노·티·엔진)	소형 가전 기기등에 적용 하기 위한, 동전 크기의 기기 플랫폼
pT-Engine(피코·티·엔진)	조명기구, 스윗치, 센서, 자물쇠, 밸브 등, 유비쿼터스·컴퓨팅 환경의 최소 단위에 적용하는 기기를 위한 플랫폼

분야도 확대되어 세계화되어 가고 있는 것은 사실이나 공개소스로서 존재 자체가 이미 세계화라고 볼 수 있는 임베디드 리눅스와의 경쟁이 불가피할 것이라 예측된다.

4.3 Eclipse(http://www.eclipse.org)

Eclipse 프로젝트는 IDE((Integrated Development Environment), 즉 통합 개발환경을 표준화하기 위한 프로젝트로서, IBM과 OTI에서 IBM VisualAge 시리즈 개발에 참여했던 인력을 중심으로 1999년에 시작된 Java 기반의 프로젝트다. Eclipse 프로젝트의 초기에는 개발자가 여러 종류의 IDE 사이를 번거롭게 오갈 필요 없이 이들을 조화롭게 사용할 수 있도록 해주는 프레임워크 개발을 목표로 하였으나, 2001년에 IBM이 이를 오픈소스 프로젝트로 만들었고, 이에 Borland, MERANT,



(그림 4) Eclipse의 플러그인 개념도(출처: http://www.eclipse.org)

QNX Software Systems, Rational Software, Red Hat, SuSE, TogetherSoft, Webgain이 합류함으로써 Eclipse.org 컨소시엄이 결성되게 되었다. 그 후에도 HP, Fujitsu, Sybase, Oracle, OMG, ETRI 등 많은 소프트웨어 벤더 및 연구소들이 컨소시엄에 참가하여 통합 개발 환경을 위한 산업 표준적인 플랫폼을 만들기 위해 노력하고 있다.

Eclipse 프로젝트는 특히 다양한 OS나 GUI와 비GUI를 모두 지원하는 등 어플리케이션 개발 툴을 위한 개방형 플랫폼 제공할 뿐만 아니라, HTML, Java, C, JSP, EJB, XML, GIF 등 컨텐츠의 형식을 제한하지 않고 언어 중립성을 추구하고 있다. 또한 이미 설치된 제품에 새로운 도구를 자유롭게 추가할 수 있게 하는 등 Eclipse.org는 플랫폼 개발에서 그치는 것이 아니라, 개발자들이 Eclipse 플랫폼에서 동작하는 도구를 만들 때 중복되는 노력을 피하고 성과를 최대한 공유하도록 하기 위하여 공통 컴포넌트를 개발하는 Eclipse 도구 프로젝트(The Eclipse Tools Project)도 진행하고 있다.

Eclipse의 기술적인 핵심은 (그림 4)에서 볼 수 있듯이 Eclipse의 모든 기능은 플러그인을 통해 제공한다는 점이다. 즉, 플러그인은 플랫폼이 제공하는 서비스를 이용하거나 또는 다른 플러그인과 연동/통합됨으로써 플랫폼의 확장성이나 통합성의 극대화를 추구하고 있다. 현재 Eclipse 플랫폼은 V3.0.1이 공개되어 있다.

5. 결 론

본 고에서는 Post-PC 시대로 접어들면서 반도체, 통신 등과 아울러 향후 모든 기기의 필수 요건으로 작용하게 될 임베디드 소프트웨어에 대하여 이의 플랫폼으로서의 의미와 표준화 동향에 대하여 간략히 살펴보았다.

이상에서 소개된 포럼차원의 이슈 이외에 응용프로그램의 호환성에서 가장 걸림돌이 되는 것 중 하나인 GUI 분야야 말로 간과할 수 없는 영역이라 할 수 있겠다. 현재 임베디드 리눅스의 경우 시스템 콜은 호환성이 유지된다 할지라도 거의 모든 응용프로그램이 필연적으로 사용해야 하는 GUI 라이브러리나 위젯은 GTK[13]와 Qt/Embedded [14] 등 그 솔루션이 다양하게 제공되고 있는 상황이다. 따라서 시스템 콜 차원의 API를 아무리 표준화 한다 하여도 이러한 GUI와 미들웨어 부분이 표준화되지 않으면 프로그램의 호환성은 보장되지 않을 것이기 때문에 이에 대한 대비책이 필요한 상황이다.

본 고에서 소개된 KESIC, CELF, T-Engine, Eclipse 등은 그 어느 표준 포럼들보다도 활발하게 움직이고 있는 포럼들로써 필자가 본 고의 내용을 적고 있는 순간에도 그 내용은 업데이트되고 기술의 수준은 향상되고 있다고 해도 과언은 아닌 것이다. 또한 본 고에서 다루지 못한 수많은 표준 포럼들이 존재하고 있는 것도 사실이다. 예를들어 홈네트워크 분야의 Digital Living Network Alliance (DLNA)[15] 등 각 분야 나름대로의 플랫폼 표준화 활동이 이루어지고 있으며, 앞으로 지능형 로봇, 텔레매틱스, 이동통신, 디지털 방송 등 차세대 성장동력 분야별로 이에 맞는 임베디드 소프트웨어 플랫폼 표준들이 제정될 전망이다. 따라서 이들의 공통분모를 신속히 찾아내어 이를 표준화함으로써 산업전체의 효율성을 증대시키고 분야간 시너지를

높이는 전략이 필요한 시점이라 하겠다. 아무쪼록 이렇게 급변하는 임베디드 소프트웨어 표준화 물결 속에 소외되지 않고 보다 적극적이고 주도적인 자세로 참여하여 임베디드 소프트웨어의 강자로 자리매김 하기를 바랄 뿐이다.

참고문헌

[1] IEEE POSIX 1003.1-2001, http://www.ieee.org

[2] OSGi, "Open Services Gateway Initiative (OSGi) Service Platform Specification Release 3", http://www.osgi.org, Mar 27, 2003

[3] HAVi, "Specification Of the Home Audio.Video Interoperability(HAVi) Architecture Version 1.1," HAVi Association, http://www.havi.org, May 15, 2001.

[4] UPnP(Universal Plug and Play),http://www.upnp.org/

[5] 임채덕 등, "임베디드 소프 트웨어 기술동향 및 산업 발전 전망," 정보통신연구진흥 제 4권 제 3 호, 정보통신연구진흥원, http://iita6.iita.re.kr:8888/korean/journal/13/focus_01.htm, 2002. 9.

[6] KESIC, "KELPS(Korea Embedded Linux Platform Specification) Version 1.0", http://www.kesic.or.kr

[7] LSB(Linux Standard Base), http://www.linuxbase.org/spec/

[8] Open Group Single UNIX Specification version 3(SUSV3), http://www.opengroup.org

[9] Embedded Linux Consortium, "ELCPS (Embedded Linux Consortium Platform Specification) Version 1.0," http://www.embedded-linux.org

[10] CELF, "CELF Specification Version 1.0 Release 2", http://ww.celinuxforum.org

[11] T-Engine Forum, http://www.t-engine.org, http://www.t-engine.co.kr

[12] Eclipse, http://www.eclipse.org

[13] GTK+, http://www.gtk.org

[14] Qt, http://www.trolltech.com/

[15] DLNA(Digital Living Network Alliance), http://www.dlna.org

저자약력

김 두 현

1985년 서울대학교 컴퓨터공학과 학사
1987년 한국과학기술원 전자계산학과 석사
1993년 한국과학기술원 전자계산학과 박사
1987년~2004년 한국전자통신연구원 책임연구원
2004년 - 현재 건국대학교 인터넷미디어공학부 조교수
관심 분야 : 객체지향 분산실시간시스템, 멀티미디어 시스템, 임베디드시스템

김 정 국

1977년 서울대학교 계산통계학과 학사
1979년 한국과학기술원 전자계산학과 석사
1986년 한국과학기술원 전자계산학과 박사
1984년 - 현재 한국외국어대학교 컴퓨터 및 정보통신공학부 교수
관심분야 : 객체지향 분산실시간시스템, 실시간 운영체제, 임베디드시스템