실시간 운영체제 상에서 에너지 절감을 위한 자바 API

Java API for Energy Saving on Real-Time Operating System

손필창, 전상호, 송예진, 조문행, 정명조, 이철훈
충남대학교 컴퓨터공학과

Pil-Chang Son(pcson@cnu.ac.kr), Shang-Ho Jeon(shjun@cnu.ac.kr),
Ye-Jin Song(yjsong@cnu.ac.kr), Moon-Haeng Cho(root4567@cnu.ac.kr),
Myoung-Jo Jung(mjjung@cnu.ac.kr), Cheol-Hoon Lee(clee@cnu.ac.kr)

요약

최근 들어 이동형 장치 및 휴대형 장치 같은 소형 임베디드 장치들이 빠르게 확산되고 있다. 이들 장치의 응용 프로그램들은 점차적으로 복잡하게 되어 그 처리 능력이 증대되었으며, 그에 따라 배터리 수명이 장치 사용에 있어 가장 심각한 제약사항이 되고 있다. 그래서 전력 소모를 줄이는 방법이 많이 연구되고 있으며, 전력 소모 감소 기능(저전력)이 지원되는 하드웨어, 소프트웨어가 탑재된 제품들도 생산되고 있다. 본 논문에서는 임베디드 자바플랫폼인 J2ME에 저전력 기법을 적용하기 위해 IBM과 MontaVista Software에서 제안한 동적 전원 관리(Dynamic Power Management : DPM) 기법을 탑재한 실시간 운영체제 UbiFOSTM를 기반으로 저전력 자바 API를 제안하고, 실험을 통해 30% 정도의 전력 감축이 됨을 보인다.

■ 중심어 : □저전력 자바 API□동적 전원 관리□실시간 운영체제□

Abstract

Recently, embedded systems like mobile and portable devices are quickly disseminated around the world. Since these devices need more computation power as the applications become gradually complicated, the battery lifetime becomes the most serious constraints. So research efforts have been focused on reducing the power consumption, resulting in producing devices with low-power H/W and S/W components. In this paper, we propose a low-power Java API set using the dynamic power management (DPM) scheme in the J2ME Java Platform on the real-time operating system UbiFOSTM, and show that we could save energy up to 30% through experiments using the API set.

■ keyword : □Low-Power Java API□Dynamic Power Management(DPM)□Real-Time Operating System□

I. 서 론

최근에 Cellular Phone, MP3 Player, Digital Camera, PMP와 같은 소형기기 에서부터 크게는 자동차, 비행기, 우주왕복선까지 각종 기기들을 위한 전자 제어 시스템 기능이 다양화 되고, 이를 위한 시스템 하드웨어의 성능이 높아지면서 시스템에서 소비하는 전력 역시 증가하게 되었다. 소비전력의 증가는 각종 기기의 사용시간 단축과 발열량 증가로 인한 기기 오동작 가능성의 증가를 초래하였다. 이를 해결하기 위해 하드웨어 측면에서의 배터리 성능향상을 위한 개발과 동시에 소프트웨어 측면에서도 정해진 배터리의 성능을 극대화 하여 사용할 수 있

* 본 연구는 정보통신부의 선도기반기술개발사업의 지원으로 수행되었습니다.

접수번호 : #061101-003
접수일자 : 2006년 11월 01일
심사완료일 : 2006년 12월 05일
교신저자 : 이철훈, e-mail : clee@cnu.ac.kr

는 기법의 개발이 필요하며, 이를 위하여 다양한 분야에서 많은 기법들이 연구되어 왔다.

자바는 플랫폼 독립성, 보안성, 네트워크 이동성, 실행 코드의 재사용성, 작은 실행 파일 크기 등의 장점을 가지고 있기 때문에 제한된 자원을 사용하는 임베디드 시스템에서 보편적인 플랫폼으로 사용되고 있다. 임베디드 시스템에서의 대표적인 자바 플랫폼은 SUN사에서 제공하는 J2ME로서 소형 기기들을 위한 제한된 자바 API를 제공한다. 이러한 자바 API는 네이티브(Native) 함수와 연계하여 동작하게 되는데 에너지 소모를 줄일 수 있는 기법을 네이티브 함수 부분에 적용하고, 이 네이티브 함수를 이용한 자바 API로 작성된 자바 애플리케이션은 동작 시 소모되는 에너지를 줄일 수 있게 된다.

본 논문에서는 임베디드 자바환경에서 소프트웨어적인 기법으로 실시간 운영체제 UbiFOS™ 상에서 저전력을 실현할 수 있는 저전력 자바 API를 설계 및 구현하였다.

II. 관련 연구

1. 실시간 운영체제 UbiFOS™

연성 실시간 운영체제인 UbiFOS™ 은 멀티태스킹 환경을 지원하고 256단계 우선순위 기반의 선점형 스케줄러를 제공하며, 동일한 우선순위에 대해 FIFO와 라운드 로빈 스케줄링을 제공한다. 또한 태스크에 관련된 태스크 관리기능과 메모리를 동적으로 관리하기 위한 메커니즘을 가지고 있으며, 태스크 간 동기화와 태스크 간 통신 기능을 제공하는 여러 구성요소를 가지고 있다[1].

태스크 관리 기능을 보면 우선 태스크의 생성 및 삭제가 가능하며, 태스크 생성 시에 필요한 스택은 전역변수를 통해 할당 받거나, 동적 메모리 관리를 위한 힙(Heap) 영역에서 할당 받을 수 있다. 또한 동적으로 태스크의 우선순위를 바꿀 수 있다[1][3].

UbiFOS™ 는 동적 메모리 관리를 위해 가변 크기의 메모리를 할당, 해제할 수 있는 힙 스토리지 매니저(Heap Storage Manager)와 고정 크기의 메모리를 할당 및 해제할 수 있는 메모리 풀(Memory Pool)을 제공하며, 태스크 간 동기화를 위해 세마포(Semaphore)와 이벤트 플래그(Event Flag)를 제공한다. 세마포는 바이너리 세마포와 카운팅 세마포가 있으며, 바이너리 세마포의 특수한 경우로, 우선순위 역전현상(Priority Inversion)을 해결하기 위한 우선순위 상속(Priority Inheritance), 삭제 안전장치(Deletion Safety), 재귀(Recursion)를 제공하는 상호 배제 세마포가 있다[1][3][5].

또한 독립적인 태스크들 사이에 정보를 주고받기 위해서 메시지 메일박스(Message MailBox)와 메시지 큐(Message Queue), 메시지 포트(Message Port), 태스크 포트(Task Port), 시그널(Signal)을 제공한다[3].

2. DPM(Dynamic Power Management: DPM) 기법

IBM 과 MobaVista Software에서 제안한 DPM 기법은 소비전력이 가장 큰 CPU의 frequency 뿐만 아니라 Bus Clock 까지 조절하여 전체 시스템의 소비전력을 감소시키는 전력 관리 기법으로, 시스템에 탑재되는 애플리케이션과 각 디바이스에서 요구하는 Bus Clock 까지 고려하여 소비전력을 감소시키는 융통성을 갖춘 전력 관리 기법이다[4].

2.1 DPM 구조모델

시스템 디자이너는 정책 관리자(Policy Manager)를 통해 시스템에서 제공하는 CPU Frequency와 Bus Clock에 맞춰 각 애플리케이션이 요구하는 CPU Frequency와 Bus Clock의 집합을 정의한 정책들(Policies)을 생성한다.

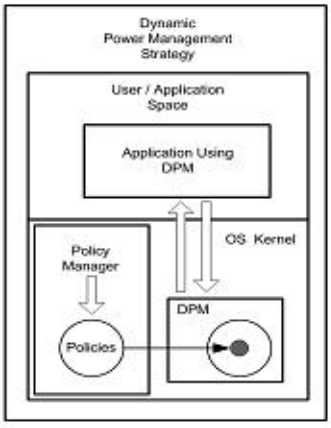


그림 1. DPM 구조모델

[그림 1]은 커널에서 제공하는 DPM 관련 API를 통해 정책을 결정하고 각 응용프로그램에 미리 정의된 정책을 할당하여 수행하는 DPM 구조모델이다[4].

2.2 DPM 구동정책(Operating Policies)

IBM 과 MontaVista Software에서는 IBM PowerPC 405LP 보드를 사용하였는데, 최대 266MHz에서 33MHz까지 지원하는 하드웨어를 통해 5단계의 구동정책을 제안하였다.

[그림 2]를 보면, 5단계(Task, Task+, Task-, Idle, IdleTask)로 구분하여, 각 단계별로 CPU Frequency, Bus Clock과 이에 따른 전압 레벨을 집합으로 하는 구동정책을 정의하였다[4].

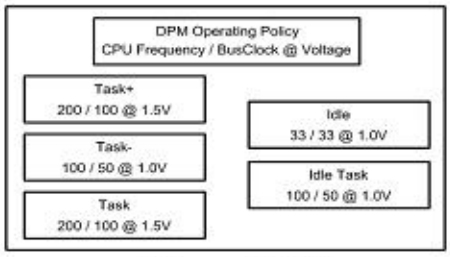


그림 2. DPM 구동정책

2.3 DPM 동작상태(Operating States)

[그림 3]은 DPM 동작 상태로, 각 태스크는 자신에게 할당된 정책에 따라 동작하며, DPM 스케줄링 API에 의해 할당된 정책을 변화시킬 수 도 있다[4].

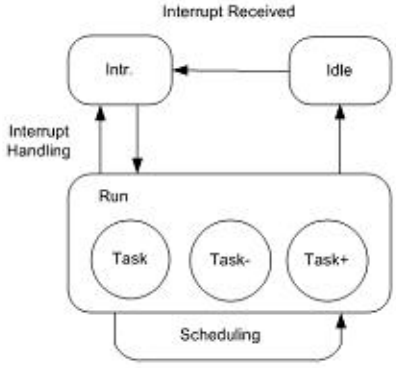


그림 3. DPM 동작 상태

3. J2ME 플랫폼

SUN사에서 제공하는 임베디드 자바 환경을 위한 플랫폼인 J2ME는 CLDC(Connected Limited Device Configuration : CLDC)와 MIDP(Mobile Information Device Profile : MIDP)로 구분되어 진다. [그림 4]는 J2ME 플랫폼을 나타낸 것이다.

본 논문에서는 [그림 4]의 J2ME 플랫폼에서 OEM-Specific Classes 영역에 저전력에 관한 클래스를 추가하여 네이티브 소프트웨어 플랫폼(Native Software Platform)과 연계하여 저전력을 위한 자바 API를 구현하였다. 저전력을 위한 DPM 기법은 네이티브 소프트웨어 플랫폼에 적용하였다[6].

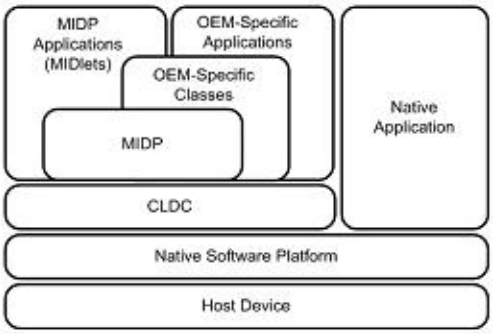


그림 4. J2ME 플랫폼

III. 임베디드 환경에서 저전력을 위한 자바 API 제안

1. 실시간 운영체제 상에서 DPM 기법 구현

저전력 기법으로는 DVS(Dynamic Voltage Scaling)와 DPM(Dynamic Power Management)이 있다. DVS는 DPM과 달리 태스크의 종료시점(deadline)을 정확히 알 수 있는 경성 실시간 운영체제(hard real-time operating system)에서 주로 사용되는 EDF(Earliest Deadline First), RM(Rate Monotonic) 스케줄링 기법으로 유휴시간을 계산하여 종료시점까지 수행시간을 늘려 전력은 전압의 제곱에 비례한다는 원리에 따라 공급 전압을 낮추는 저전력 기법을 말한다. [그림 5]는 DVS의 동작원리를 그림으로 나타낸 것이다.

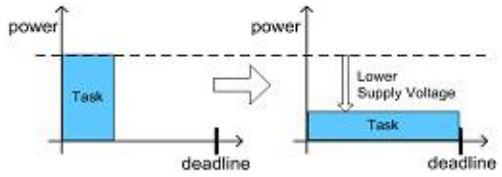


그림 5. 동적 전압 조절(DVS)

본 논문에서 사용한 실시간 운영체제인 UbiFOS™는 우선순위기반의 스케줄링 기법을 사용하는 연성 실시간 운영체제(soft real-time operating system)이며, DVS를 적용하기 위해서는 스케줄링 기법의 변경이 필요하다. 또한 본 논문에서 사용한 하드웨어 플랫폼(MBA 2440 보드)은 전압 조절 기능이 지원 되지 않는 플랫폼이다. 하지만 이 플랫폼은 공급 주파수를 조정하는 기법인 DFS(Dynamic Frequency Scaling : DFS)는 가능하며, DPM은 기존 UbiFOS™ 의 커널수정 없이 저전력 구현이 가능하기 때문에 이 기법을 사용하였다.

본 논문의 하드웨어 플랫폼은 초기에 399MHz Frequency로 동작하며, Frequency를 설정하는 레지스터의 값에 따라 CPU Frequency를 조절할 수 있어 DFS가 가능하다.

표 1. DPM 구동 정책

Policy	Task+	Task	Task-	Idle
Default	399	399	399	399
LS399	399	266	266	96
LS266	399	266	96	96
LS96	399	96	96	96

DPM 기법을 적용한 실시간 운영체제는 기존 UbiFOS™ 에 하드웨어 플랫폼에서 제공하는 Frequency 레벨을 기준으로 구동정책을 구성하였으며, 본 논문에서는 총 4가지의 정책을 정의하여 구현하였다. 각 태스크는 실행 중에 DPM 스케줄링 API에 의해서 구동정책을 동적으로 변경할 수 있다.

[표 1]은 본 논문에서 구현한 DPM 구동정책으로 4단계로 이루어져 있으며, 애플리케이션에 따라 동적으로 CPU Frequency의 변경이 가능하도록 구성하였다. Task+, Task, Task-, Idle 은 각 DPM 정책 적용 시 동적으로 변경할 수 있는 CPU Frequency 모드로 기본은 Task 모드이다. 즉 각 DPM 정책에서 애플리케이션 수행 시 CPU Frequency를 높이려면 Task+ 모드, 낮추려면 Task- 모드로 Frequency를 동적으로 변경할 수 있다. 애플리케이션 수행 중 태스크가 유휴(Idle) 상태가 되면 Idle 모드로 자동 변경 된다.

본 논문에서는 MBA2440에서 CPU Frequency의 변경 가능한 범위를 고려하여 DPM 정책을 Default, LS399, LS266, LS96 4단계로 나누어 저전력 자바 API를 구현하였다.

표 2. DPM 관련 API

함 수	설 명
LP_Set_DPM_Policies()	구동정책 설정 함수
LP_MBA2440_Fscaler()	구동정책에 따른 레지스터 설정 함수
LP_Cal_Frequency()	CPU 주파수 계산 함수

[표 2]는 본 논문에서 구현한 DPM 기법의 저전력 관련 API로 LP_Set_DPM_Policies()함수는 구동정책을 인자로 CPU Frequency를 설정하는 함수이고, LP_MBA2440_Fscaler()는 구동정책에 따라 Frequen cy 값을 변경하기 위해 관련 레지스터의 값을 변경하는 함수이며, LP_Cal_Frequency()는 현재 태스크에 할당된 CPU Frequency를 계산하기 위한 함수이다.

2. DPM 기법을 적용한 저전력 자바 API 설계 및 구현

저전력 자바 API를 구현하기 위해서는 OEM-Specific Classes 영역에 저전력을 위한 자바 클래스와 네이티브 소프트웨어 플랫폼에 네이티브 함수를 구현하여 상호 동작이 되도록 구현하여야 한다.

[그림 6]은 저전력을 위한 자바 API의 동작 메커니즘을 보여주는 그림으로써 저전력 자바 애플리케이션을 위한 저전력 자바 API와 DPM 기법이 적용된 저전력 네이티브 함수와의 상호 동작을 보여주고 있다. 저전력 자바 API와 네이티브 함수의 연결은 KNI (Kilobyte

Virtual-Machine Native Interface : KNI)를 이용하여 상호 동작이 가능하도록 구현하였다[7].

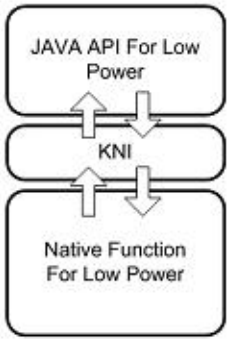


그림 6. 저전력 자바 API의 동작 메커니즘

저전력 자바 API는 저전력 자바 클래스의 메소드로 그 기능이 제공된다. 이 메소드는 자바 메소드와 자바 네이티브 메소드로 구분되어 지는데, 자바 메소드는 클래스 내에서 그 기능이 자바 프로그램 언어를 사용하여 정의된 메소드를 말하며, 자바 네이티브 메소드는 클래스의 시스템 의존적인 부분을 처리하기 위해 필요한 메소드를 말하며 native 키워드로 선언이 되고, 시스템을 구성하는 언어로 구현된 네이티브 함수에 의해 그 기능이 정의가 된다.

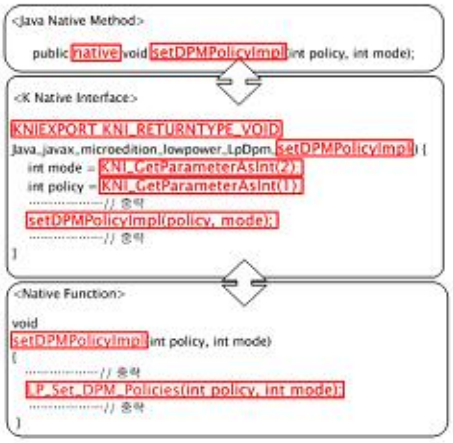


그림 7. 자바 네이티브 메소드의 구현모습

[그림 7]은 자바 네이티브 메소드가 네이티브 함수에 의해 기능이 정의되는 모습을 보여준다. 네이티브 함수는 시스템을 구성하는 언어로 구현된 함수를 말하며, 본 논문에서는 C 프로그램 언어를 사용하였다.

[그림 7]은 KNI를 이용하여 자바 네이티브 메소드와 네이티브 함수를 연결한 모습도 보여준다.

2.1 저전력 자바 API 구현

본 논문에서는 J2ME 플랫폼에 OEM-Specific Class로 javax.microedition.lowpower 패키지를 추가함으로써 저전력 자바 API를 구현하였다.

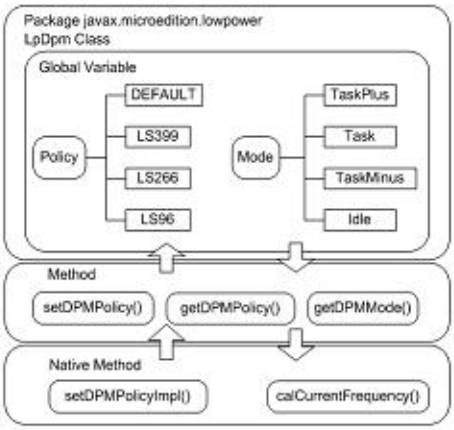


그림 8. 저전력 자바 API를 위한 자바 클래스

[그림 8]은 저전력 자바를 위한 LpDpm 클래스 의 구성을 보여주고 있다. 본 논문에서 제시한 4개의 DPM 정책은 각각 DEFAULT, LS399, LS266, LS96라는 명칭의 전역변수로 선언하였고, 이에 상응하는 모드를 각각 TaskPlus, Task, TaskMinus, Idle로 선언하였다.

[표 3]은 본 논문에서 구현한 저전력 자바 API로서 자바 애플리케이션 프로그래머가 setDPMPolicy()메소드를 이용해 DPM 정책을 설정할 수 있게 하였고, getDPMPolicy(), getDPMMode()메소드를 통해 정책과 모드를 가져올 수 있도록 구현하였다. setDPMPolicyImpl()메소드와 calCurrentFrequency()메소드는 자바 네이티브 메소드로 setDPMPolicy(), getDPMPolicy(), getDPMMode()메소드의 내부에서 사용되는 메소드이다.

표 3. 저전력 자바 API

저전력 자바 API	내 용
setDPMPolicy()	DPM 정책 설정 함수
getDPMPolicy()	DPM 정책 얻어오는 함수
getDPMMode()	DPM 모드 얻어오는 함수
setDPMPolicyImpl()	DPM 정책 설정하는 네이티브 함수
calCurrentFrequency()	현재 CPU Frequency를 계산하는 네이티브 함수

본 논문에서는 저전력 자바 API를 사용하는 프로그래머가 자바 애플리케이션에서 저전력을 요구하는 부분에 원하는 저전력 정책과 모드를 설정하여 전력관리를 동적으로 수행할 수 있도록 저전력 자바 API를 설계 및 구현하였다. 이 때 저전력 정책과 모드는 자바 애플리케이션의 성능에 영향을 미치지 않는 범위 내에서 설정하여야 하며, 자바 애플리케이션 개발자는 사전에 개발 플랫폼에서 정의하고 있는 정책과 모드에 대한 정보를 알고, 저전력 자바 API를 사용하여야 한다.

2.2 저전력 자바 API를 위한 네이티브 함수 구현

[그림 8]의 LpDpm 클래스에서 setDPMPolicyImpl() 메소드와 calCurrentFrequency() 메소드는 자바 네이티브 메소드로 정의했기 때문에 KNI를 통하여 자바 네이티브 메소드 부분을 네이티브 함수를 이용하여 구현해 주어야 한다.

[그림 9]은 LpDpm 클래스에서 자바 네이티브 메소드로 정의한 setDPMPolicyImpl() 메소드와 calCurrent Frequency() 메소드를 KNI를 통해 운영체제의 DPM API들을 사용하여 구현한 내용이다.

[그림 9]에서 보는 바와 같이 setDPMPolicyImpl() 메소드는 KNI를 통해 적용하고자 하는 DPM 기법의 모드와 정책을 LpDpm 클래스로부터 인자로 넘겨받아 DPM 관련 API인 LP_Set_DPM_Policies() 를 이용하여 동작하도록 네이티브 함수를 구현하였고, calCurrent Frequency() 메소드는 DPM 관련 API인 LP_Cal_Frequency()함수를 이용하여 CPU Frequency를 계산한 결과 값을 LpDpm 클래스로 반환하도록 네이티브 함수를 구현하였다.

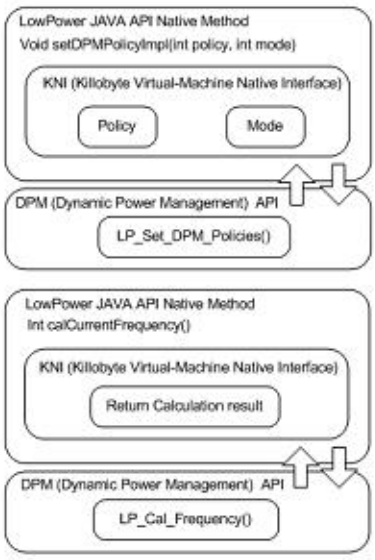


그림 9. 저전력 자바 API를 위한 네이티브 함수

본 논문에서 구현한 저전력 자바 API는 실시간 운영체제 UbiFOS™ 에서 DPM 기법을 적용하여 구현하였고, 기존의 DPM 관련 API를 이용하여 다양한 저전력 자바 API를 쉽게 추가 구현 할 수 있도록 설계하였으며, 운영체제에서 DPM의 기능이 추가된 경우에도 저전력 자바 API에 쉽게 적용 가능하도록 구성하였다.

Ⅳ. 실험 결과

본 논문에서 구현한 저전력 자바 API는 ARM920T 기반의 MBA2440 보드에서 실시간 운영체제 UbiFOS™를 사용하여 개발하였다. 개발도구는 ARM Developer Suit v1.2를 사용하였고, 디버그를 위해 OPENice-A1000 Emulator를 사용하였다.

[그림 10]은 저전력 자바 API를 개발하기 위한 테스트 환경이다. 전력은 전류에 비례하기 때문에(W=VI) 본 논문에서는 5V의 고정전압을 갖고 있는 MBA2440보드 상에서 전류 측정을 통하여 자바 애플리케이션이 수행하는 동안 전력이 얼마나 감소하였는지 측정하였다.

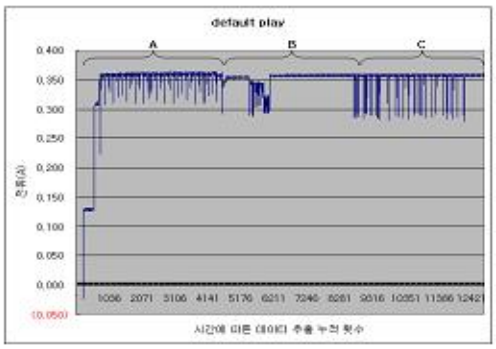
그림 10. 테스트 환경

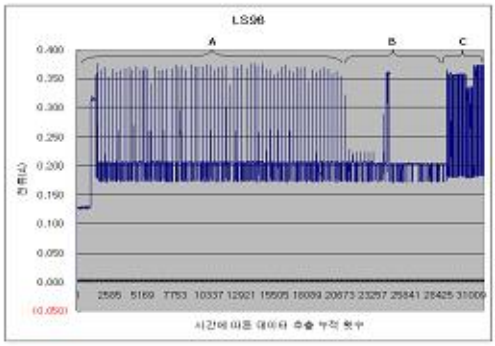
본 논문에서는 웨이브(Wav) 파일을 재생하는 자바 애플리케이션과 푸쉬푸쉬(PushPush) 게임 자바 애플리케이션을 테스트 애플리케이션으로 사용하였으며, 3장에서 제시한 4단계의 DPM 정책의 Task 모드로만 동작하도록 자바 애플리케이션 동작 초기에 저전력 자바 API를 사용하여 설정하였다. 또한 UbiFOS™의 유휴(Idle) 태스크가 동작할 경우 각 정책의 Idle 모드로 동작하도록 설정하였다.

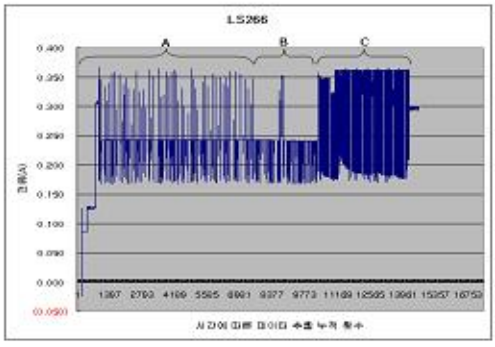
[그림 11]은 본 논문에서 구현한 저전력 자바 API를 이용하여 자바 테스트 애플리케이션을 테스트한 결과이다. 각 그래프에서 구간 A는 웨이브 파일 재생할 때의 전류를 나타내며, 구간 B는 웨이브 파일 재생이 끝나고 푸쉬푸쉬 게임이 시작되기 전의 전류를, 구간 C에서는 푸쉬푸쉬 게임이 동작할 때의 전류를 나타낸다.

그래프의 결과 값이 일정하지 못하고 상, 하로 자주 이동한 이유는 저전력 자바 API외의 외부적인 요인(오디오 입출력 장치, LCD)에 의한 것이다. 웨이브 파일 재생의 경우 웨이브 파일에 대한 접근은 DMA(Direct Memory Access)에 의해 이루어지는데, 이 기간 동안 UbiFOS™는 유휴 태스크를 수행하기 때문에 이 기간에는 각 정책의 Idle 모드로 동작을 하게 된다. 웨이브 파일을 재생할 때는 오디오 출력 장치를 사용하기 때문에 전류의 양이 늘어난다. 푸쉬푸쉬 게임의 경우 MBA2440 보드는 전압이 공급되는 한 LCD를 계속 refresh를 하기 때문에 LCD를 사용하는 동안 전류의 양이 늘어나게 된다. 또한 웨이브 파일 재생과 마찬가지로 UbiFOS™가 유휴 태스크를 수행할 때는 각 정책의 Idle 모드로 동작한다. 이러한 외부적인 요인은 저전력 자바 API에 의해 발생한 것이 아니며, 따라서 본 논문에서는 각 결과 그래프의 평균 측정치를 이용하여 전류를 측정하였다.

그 결과 [그림 11]에서 보는 바와 같이 Default 정책을 적용하였을 경우 DPM 기법이 전혀 적용되지 않은 상태이기 때문에 전류의 양이 약0.36A인 것을 확인 할 수 있었다. LS96 정책을 적용하였을 때는 CPU Frequency를 96MHz로 동작 시킨 것으로 Default 정책을 적용하였을 때보다 적은 약 0.21A로 측정되었고, LS266과 LS399 정책은 각 정책의 Task 모드가 266MHz로 CPU Frequency가 동일하기 때문에 약 0.24A로 두 정책간의 전류 측정량은 큰 차이를 보이지 않았다.







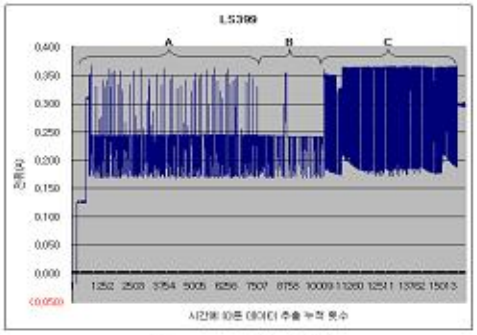


그림 11. 테스트 결과

저전력 자바 API를 사용할 경우, 즉 CPU Frequency를 낮춰 자바 애플리케이션을 실행하면, 그렇지 않을 때보다 소비 전류가 약 0.14A가 줄어든다. 순간 전력은 그 순간 전류 값에 비례하므로, 어떤 시간 구간에서의 에너지 소비는 그 구간에서의 평균 전류 값에 비례한다. 따라서 전류측정을 통해 웨이브 파일 재생, 푸쉬푸쉬 게임 자바 예제 애플리케이션의 경우 저전력 자바 API를 사용하여 전체 에너지 소비를 30% 정도 줄일 수 있음을 확인할 수 있다.

V. 결론 및 향후 연구 과제

본 논문에서는 실시간 운영체제 UbiFOS™ 상에서 DPM 기법을 구현하여 저전력 자바 API 구현의 기반을 만들었고, 자바 API 부분에서는 저전력을 위해 javax.microedition.lowpower 패키지를 생성하고 저전력을 위한 LpDpm 클래스를 구현해서 저전력 자바 API를 구현하였다. 또한 이 클래스와 상호 동작이 가능한 네이티브 함수를 운영체제의 DPM 관련 API를 사용하여 구현함으로써 자바에서도 저전력 기법을 적용할 수 있도록 설계 및 구현하였다.

향후 연구 과제는 실시간 운영체제 UbiFOS™ 에서 DPM 기능을 추가로 확장하고, DVS의 기능까지 구현하여 다양한 저전력 기능을 사용할 수 있는 저전력 자바 API를 구현하는 것이다. 이는 DVS 가 가능한 타겟 임베디드 플랫폼에서 개발하여야 할 것이며 자바에서 전력을 효율적으로 관리 할 수 있도록 그에 맞는 자바 API를 설계 및 구현하여야 한다.

참 고 문 헌

[1] http://www.aijisystem.com
[2] M. T. Schmitz, B. M. Alhashimi, and P. Eles, *System-Level Design Techniques for Energy-Efficient Embedded Systems*, Kluwer Academic Publishers, Boston, 2004.
[3] 박희상, 정명조, 조희남, 이철훈, "Design of Open-Architecture Real-Time OS Kernel", 한국정보과학회, 제31권, 제1호, pp.163-165, 2002(4).
[4] H. Blanchard, B. Brock, M. Locke, M. Orvek, R. Paulsen, and K. Rajamani, *Dynamic Power Management for Embedded Systems*, IBM and MontaVista Software, Version1.1, Nov. 2002.
[5] 조문행, 정명조, 이철훈, "The Design and Implemetation for Preventing A memory leakage of Memory Pool on Memory Management of Real-Time Operating Systems", 한국정보과학회, 제31권, 제1호, pp.163-165, 2005(4).
[6] http://java.sun.com/javame/ index.jsp
[7] 손필창, 강희성, 정명조, 이철훈, "The Design and Implementation of GUI in KVM on Real-Time Operating System, UbiFOSTM", 한국정보과학회, 제33권, 제1호, pp.358-360, 2006(6).

저 자 소 개

손 필 창(Pil-Chang Son) 준회원

▪2005년 2월 : 충남대학교 컴퓨터공학과 (공학사)
▪2005년 3월 ~ 현재 : 충남대학교 컴퓨터공학과 공학석사 재학
<관심분야> : 실시간 운영체제, 자바가상머신

전 상 호(Shang-Ho Jeon) 준회원

- 2006년 2월 : 충남대학교 컴퓨터공학과 (공학사)
- 2006년 3월 ~ 현재 : 충남대학교 컴퓨터공학과 공학석사 재학

<관심분야> : 실시간 운영체제, 자바가상머신

송 예 진(Ye-Jin Song) 준회원

- 2006년 2월 : 충남대학교 컴퓨터공학과 (공학사)
- 2006년 3월 ~ 현재 : 충남대학교 컴퓨터공학과 (공학석사)

<관심분야> : 실시간 운영체제, 자바가상머신

조 문 행(Moon-Haeng Cho) 정회원

- 2004년 2월 : 충남대학교 컴퓨터공학과 (공학사)
- 2006년 2월 : 충남대학교 컴퓨터공학과 (공학석사)
- 2006년 3월 ~ 현재 : 충남대학교 컴퓨터공학과 공학박사 재학

<관심분야> : 실시간 컴퓨팅, 실시간 운영체제, 초소형 초절전 실시간 운영체제

정 명 조(Myoung-Jo Jung) 정회원

- 2003년 2월 : 충남대학교 컴퓨터공학과 (공학석사)
- 2003년 3월 ~ 현재 : 충남대학교 컴퓨터공학과 박사과정 재학

<관심분야> : 실시간 컴퓨팅, 실시간 운영체제, 초소형 초절전 실시간 운영체제, 자바가상머신

이 철 훈(Cheol-Hoon Lee) 정회원

- 1983년 2월 : 서울대학교 전자공학과 (공학사)
- 1988년 2월 : 한국과학기술원 전기전자공학과 (공학석사)
- 1992년 2월 : 한국과학기술원 전기전자공학과 (공학박사)
- 1983년 3월 ~ 1986년 2월 : 삼성전자 컴퓨터사업부 연구원
- 1992년 3월 ~ 1994년 2월 : 삼성전자 컴퓨터사업부 선임연구원
- 1994년 2월 ~ 1995년 2월 : Univ. of Michigan 객원연구원
- 1995년 2월 ~ 현재 : 충남대학교 컴퓨터공학과 교수
- 2004년 2월 ~ 2005년 2월 : Univ. of Michigan 초빙연구원

<관심분야> : 실시간시스템, 운영체제, 고장허용 컴퓨팅