실시간 운영체제 *i*RTOS 상에서의 KVM기반 자바 응용프로그램 설계 및 구현

The Design and Implementation of Java Application in KVM on Real-Time Operating System, *i*RTOS

이진욱*, 김종진**, 이철훈*
충남대학교 컴퓨터공학과*, (주)넥스원**

Lee jin-wook*, Kim jong-jin**, Lee cheol-hoon*
Dept. of Computer Engineering, Chungnam National Univ*, LIG nex1 Corp.**

요약

최근 PDA나 스마트폰 같은 휴대용 장치의 사용이 비약적으로 증가함에 따라, 이 기종의 하드웨어 플랫폼 상에서 플랫폼 독립성을 제공하는 자바 기술이 임베디드 소프트웨어 분야에 핵심 플랫폼이 되어가고 있다. 특히, 휴대용 장치에서는 자바의 여러 스펙중 KVM(Kilobyte Virtual Machine)을 사용하고 있다. 본 논문에서는 상기 구현한 KVM에 대한 검증을 위해 Java Game 응용 프로그램을 설계 및 구현 하였으며, 실시간 운영체제인 iRTOS상에서 검증하였다.

Abstract

Recently the use of the portable device like a PDA or a Smart-Phone increases to follow, Java technology that support Platform Independence on Hardware Flatform has become core Flatform in Software sphere. Especially Portable device use a KVM(Kilobyte Virtual Machine) of Java's various specification. In this paper, we design and implement Java Game Application for verification of KVM and verify on Real-Time Operating System iRTOS.

Ⅰ. 서론

최근 IT 산업의 발전과 더불어, 자바의 VM(Virtual Machine)을 통한 임베디드 시스템과 정보 가전제품에 플랫폼 독립성, 보안성, 네트워크 이동성 등을 보장하기 위해 자바를 많이 사용한다. 특히 핸드폰이나 PMP등의 휴대용 장치 같은 크기가 작고 제한된 메모리나 CPU 처리능력을 가지는 장치에서는 보다 경량화 되고 그에 맞게 최적화 된 KVM(Kilobyte Virtual Machine)을 사용하여야 한다.

KVM은 플랫폼 독립성, 실행코드의 재사용성, 작은 실행 파일 크기, 동적 적응성, 이식성, 개발의 용이성 등의 장점이 있으며 많은 휴대용 장치에 탑재가 증가하는 추세에 있다. 이러한 KVM에서 자바 API를 통해 그래픽 윈도우 시스템과 Touch Screen 기능을 제공하기 위해서는 시스템에 의존적인 부분을 따로 구현해야 하는데, 이는 네이티브(Native) 함수로 구현할 수 있다. Sun사는 편리한 그래픽 환경을 위해 J2ME의 프로파일인 MIDP를 제공한다. MIDP는 자바의 GUI를 위한 표준 API를 javax.micro edition.lcdui 패키지로 명세하고 있는데, 이는 운영체제의 그래픽 윈도우 시스템과 연계되어 동작하게 된다.

본 논문에서는 KVM에 대한 검증을 위해 실시간 운영체제 *i*RTOS의 그래픽 윈도우 시스템과 GUI API와의 상호 동작을 위한 네이티브 함수(기본 도형 그리기, 색 지정, 텍스트 처리), 그리고 이벤트 처리를 이용하여 Java Game 응용 프로그램을 설계 및 구현하고 이를 실시간 운영체제인 *i*RTOS상에서 검증하였다.

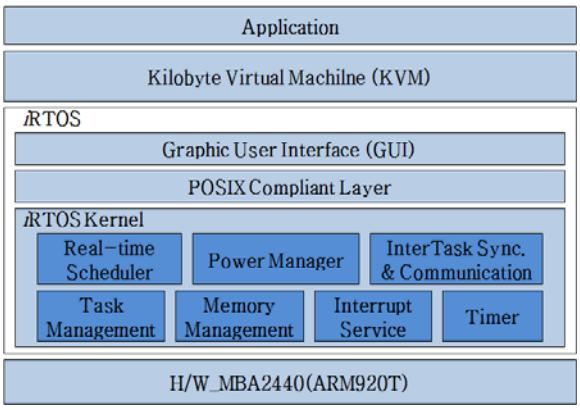
본 논문의 2 장에서는 실시간 운영체제 *i*RTOS와 KVM의 역할, J2ME플랫폼의 CLDC(Connected

Limited Device Configuration), MIDP(Mobile Information Device Profile)에 대한 관련 연구에 대해 기술한다. 3장에서는 iRTOS 기반에서 KVM을 이용한 자바 응용프로그램의 설계 및 구현을, 4장에서는 실험 환경 및 결과에 대해 설명한 후, 마지막 5장에서는 결론 및 향후 연구 과제에 대해 기술한다.

Ⅱ. 관련 연구

1. iRTOS

본 논문에서 KVM을 구현하기 위해 기반으로 사용한 iRTOS는 우선순위 기반 선점형 멀티 쓰레드 실시간 운영체제이다. 즉 실시간 운영체제 커널과 응용 프로그램이 통합되어 하나의 큰 프로그램으로 동작하는 구조로써 공통의 메모리 영역을 자유롭게 접근할 수 있고 크기는 약 24KByte 정도이다.



▶▶ 그림 1. iRTOS의 기능 블록 다이어그램

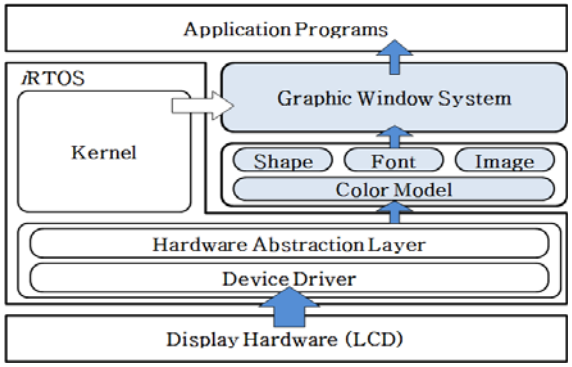
그림 1은 iRTOS의 기능을 블록 다이어그램으로 나타낸 것으로, 운영체제에서 제공되는 기능은 태스크 관리, 태스크간 동기화/통신, 동적 메모리 관리 등이 있다.

태스크 관리 기능에는 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.

iRTOS는 동적 메모리 관리를 위해 가변 크기의 메모리를 할당 및 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 세마포와 이벤트 플래그를 제공한다.

1.2. iRTOS의 그래픽 윈도우 시스템

그림 2는 현재 iRTOS 상에서 구현된 그래픽 윈도우 시스템으로 타겟 플랫폼이 바뀌었을 경우 쉽게 이식시키기 위해 Layered Architecture Design 방식으로 설계되어 있다[1].



▶▶ 그림 2. iRTOS 의 그래픽 윈도우 시스템

2. KVM

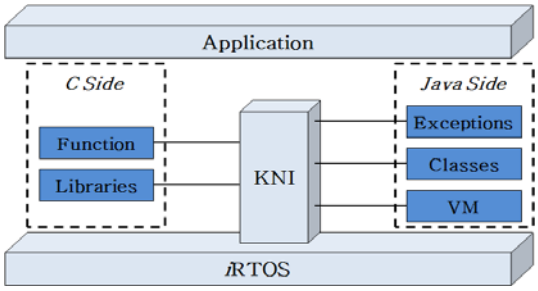
KVM은 기존 JVM의 서브셋으로 적은 메모리의 용량과 CPU 처리 능력에 맞게 설계되었다. 휴대폰, 호출기 및 PDA같은 휴대용 장치에서는 주로 KVM을 사용하며 일반적인 컴퓨팅 기능을 사용할 수 있다. KVM은 모바일 장치에서 Java 소프트웨어를 실행하기 위해 사용할 수 있는 CLDC의 유일한 구현이다. KVM 은 일반적으로 class file을 사용하고, Standard Java class loading을 지원한다. 기본 Word는 32bit 의 Java bytecode full set을 지원하며, 멀티쓰레드와 가비지 컬렉션을 지원한다. 그리고 NullPointException과 IndexOutofBoundsException의 Runtime Exception은 제거되었다[2][3].

2.1. CLDC(Connected Limited Device Configurations)

CLDC는 성능이 제한된 CPU나 메모리가 한정적인 시스템을 대상으로 하는 스펙이다. J2ME Configuration에는 자바 가상머신과 최소한의 자바 클래스 라이브러리들로 정의되어 있으며, 클래스 라이브러리들은 각각의 특정 장치들에 사용 가능하도록 수평적인 특성의 공통 분모를 정의하고 있다. 현재 J2ME는 메모리나 소비전력과 같은 자원의 제약 사항에 따라 CDC(Connected Device Configuration)와 CLDC로 구분된다.

2.2 KNI(K Native Interface)

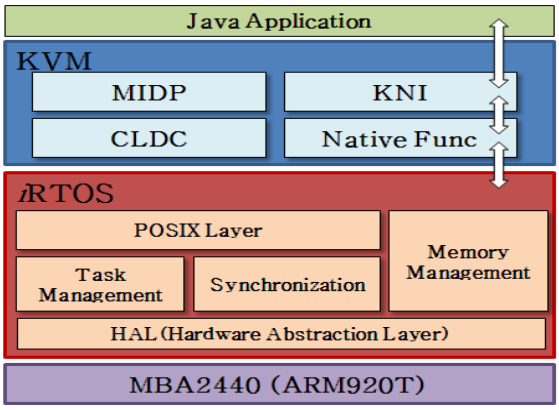
KNI는 그림 3에서 보는 바와 같이 자바와 자바 이외의 언어로 만들어진 애플리케이션이나 라이브러리가 상호 작동할 수 있도록 연결 시켜주는 인터페이스로 자바에서 지원하지 못하는 플랫폼 종속적인 기능들과 이미 다른 언어로 만들어진 애플리케이션이나 라이브러리를 사용할 수 있도록 해주는 임베디드 시스템을 위한 인터페이스이다. 네이티브 언어를 이용한 자바 프로그램에서 자바로 된 부분은 여전히 플랫폼 독립적이지만 네이티브 언어로 된 부분은 호스트 환경에 맞게 다시 컴파일 되어야 한다[4].



▶▶ 그림 3. KNI의 구조

Ⅲ. KVM 기반 자바 응용프로그램 설계 및 구현

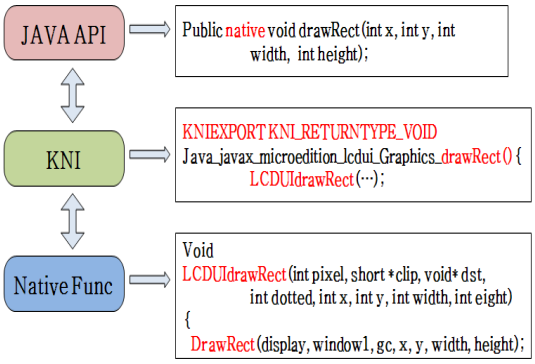
1. GUI의 네이티브(Native) 함수 구현



▶▶ 그림 4. iRTOS와 KVM의 구조

본 논문에서는 Java Game 응용 프로그램을 설계 및 구현 하였으며 이것의 기반으로 사용될 iRTOS가 이종 언어로 설계되었기 때문에 이들의 동작을 연동시켜줄 네이티브 함수 구현을 통해 운영체제에서 제공하는 기능을 응용 프로그램이 사용할 수 있도록 하였다. iRTOS와 KVM의 구조는 그림 4와 같다.

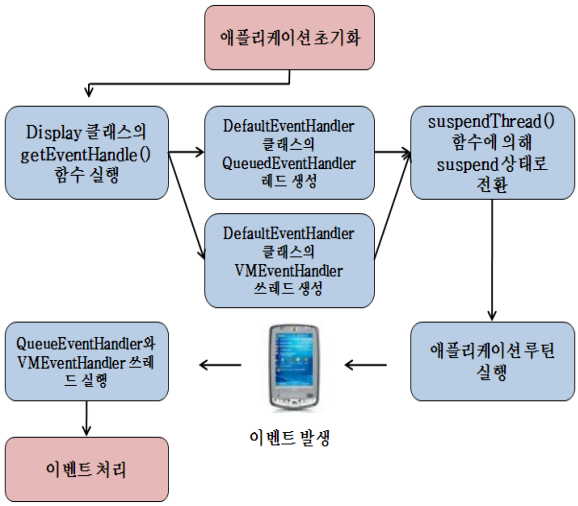
본 논문에서는 iRTOS의 그래픽 윈도우 시스템과 lcdui API의 상호 동작을 위한 네이티브 함수를 구현하였다.



▶▶ 그림 5. KVM의 GUI 동작 메커니즘

그림 5는 자바 API와 네이티브 함수, 즉 lcdui API부분과 iRTOS상에서 이벤트 발생 시에 KNI를 통해서 동작할 수 있도록 구현한 네이티브 함수와의 동작 메커니즘을 나타낸 그림이다. 네이티브 함수는 iRTOS의 그래픽 윈도우 시스템을 이용하여 구현하였고, lcdui API와 네이티브 함수의 연결은 KNI를 이용하여 lcdui API와 상호 동작이 될 수 있도록 구현하였다.

2. GUI의 이벤트 처리 구현



▶▶ 그림 6. 이벤트 처리 동작 과정

이벤트는 마우스, 키보드 상태의 비동기적인 변화를 KVM GUI에 전달하는 방법이다. iRTOS에서는 Touch Screen 이벤트를 지원하며, 이는 인터럽트 방식으로 처리한다. 본 논문에서 구현한 이벤트 처리는 그림 6과 같다.

MIDP에서는 애플리케이션 초기화 시 Display 클래스에서 DefaultEventHandler 클래스의 QueuedEventHandler 쓰레드와 VMEventHandler 쓰레드를 생성해서 실행시킨다. 그 쓰레드들은 이벤트가 발생하기 전까지 Suspend 상태로 있다가 이벤트 발생하게 되면 다시 깨어나 동작하게 된다.

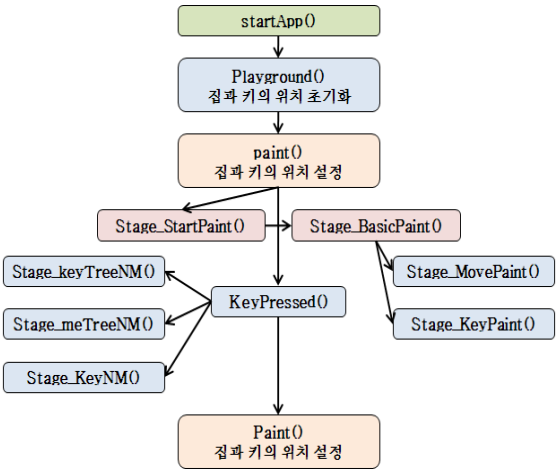
네이티브 함수에서는 이벤트가 발생했을 때 이벤트 발생과 발생한 이벤트의 종류를 알리는 flag 변수를 셋팅 함으로써 이벤트 동작을 실행하게 된다. 발생한 이벤트의 정보는 StoreKVMEvent() 함수 안에서 StoreMIDPEvent() 를 호출함으로써 저장된다.

표 1은 발생한 이벤트를 저장하는 StoreKVMEvent() 함수를 나타내고 있다. StoreMIDPEvent() 함수에 의해 저장된 이벤트는 Default EventHandler 클래스의 쓰레드들에 의해 읽혀지게 되고 이벤트처리 루틴을 실행하게 된다.

표 1. StoreKVMEvent() 함수

```
void StoreKVMEvent(int eventOccurr)
{
        Event event;
        KVMEventType evt;
        event.type = 0;

        while(eventOccurr > 0){
          if(eventOccurr    && KeyPress){
              event.type    = KeyPress;
          }else if(eventOccurr && TouchScreen){
              event.type = TouchScreen;
<생략>
  if(event.type == KeyPress) {
              evt.type = keyDownKVMEvent;
              evt.chr = translateKey();
              StoreMIDPEvent(&evt);
elseif(event.type == TouchScreen){
<생략>
     eventOccurr=0;
```



▶▶ 그림 7. Pushpush 게임의 동작 과정

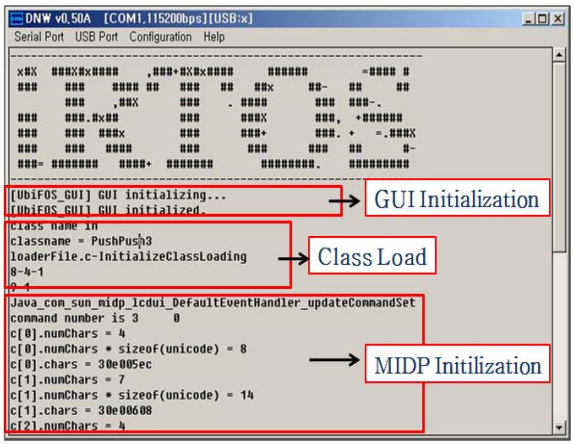
3. 자바 응용 프로그램 구현

본 논문에서 구현한 KVM 기반의 모바일 게임 프로그램 Pushpush에 대한 구현함수 및 동작과정을 그림 7에 나타내었다. 구현된 Pushpush 게임은 startApp() 함수로 게임이 시작되어 Playground() 함수로 집과 키의 위치를 초기화 시켜준다. 이 초기화가 이루어지고 난 후 GUI 네이티브 함수와 연동하는 paint() 함수 내에서 게임 초기화면에 대한 이미지 처리를 위해 Stage_StartPaint() 함수를 호출한다. Stage_StartPaint() 함수는 내부적으로 Stage_BasicPaint() 함수를 호출하여 실제 LCD 화면에 집, 나무, 키 등 게임을 위한 이미지를 그려준다. 이러한 준비 과정을 마치면 사용자의 입력 이벤트를 기다리

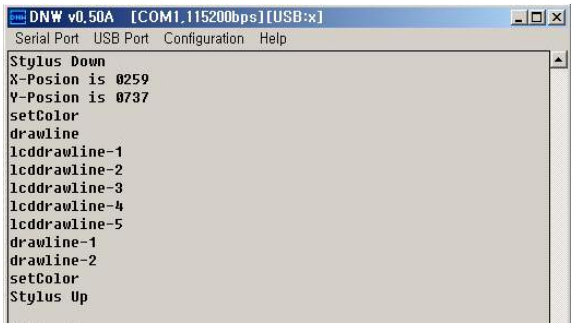
게 되고, Touch Screen 이벤트가 발생하면 GUI 이벤트 처리 루틴과 연동하는 KeyPressed() 함수를 통해 키를 이동 시킨다. 스테이지 클리어 조건에 만족하면 스테이지 넘버를 저장하고, 다음 스테이지를 위해 집과 키의 위치를 설정한다.

Ⅳ. 테스트 환경 및 결과

본 논문의 테스트 환경은 MBA2440 보드에서 실시간 운영체제로 *i*RTOS를 사용하였고, 개발도구는 ARM SDT v2.51을 사용하였다. 그리고 디버그를 위해서 OPENice-A1000 Emulator를 사용하였다. 본 논문에서 구현한 그래픽 윈도우 시스템과 KVM의 GUI API와의 상호동작을 위한 네이티브 함수를 테스트하기 위해 모바일 게임 프로그램인 Pushpush를 수행함으로써 Touch Screen 이벤트를 테스트 하였다.



▶▶ 그림 8. Initialization 과 Class Load



▶▶ 그림 9. 이벤트 발생 시 동작하는 화면

▶▶ 그림 10. Pushpush 게임 동작 화면

그 결과 그림 8, 9처럼 GUI, MIDP의 초기화와 이벤트 발생 시 네이티브 함수를 통해 동작하는 것을 확인하였고, 그림 10처럼 Pushpush 게임이 동작하는 것을 통해 KVM에 대한 검증을 수행하였다.

Ⅴ. 결론 및 향후 연구과제

본 논문에서는 리소스가 제한된 디바이스에 자바 응용 프로그램을 실행시키기 위해서 실시간 운영체제인 *i*RTOS의 그래픽 윈도우 시스템과 KVM의 GUI API 상호 동작을 위한 네이티브 함수와 이벤트 처리에 관련된 API를 구현하였다. 또한 구현된 네이티브 함수를 이용한 자바 응용 프로그램 Pushpush 게임을 설계 및 구현하였으며, 이를 통해 GUI API를 위한 네이티브 함수와 이벤트 처리에 대한 올바른 동작을 확인할 수 있었다.

향후 연구과제로는 KVM의 다양한 기능을 지원하기 위해 애플릿(Applet)과 고급 GUI 기능을 지원하는 PP(Personal Profile)를 구현함으로써 실시간 운영체제 *i*RTOS에서 다양한 자바 기능을 지원하는 노력을 기울여야 할 것이다.

■ 참 고 문 헌 ■

[1] 윤기현, 김용희, 박희상, 이철훈, "Design of Graphic User Interface for the Real Time

Operating System”,한국정보과학회, Vol.29, No2(I), pp400-402, 2002.

[2] 전상호 정근재 이정원 이철훈, “임베디드 시스템을 위한 KVM 네트워크 설계 및 구현” 한국컴퓨터종합학술대회 논문집(A), pp.349-351, Jun 2006, 한국정보과학회.

[3] 전상호, 이철훈, “임베디드 시스템을 위한 K가상머신 사운드 API 설계 및 구현”, 한국정보과학논문지, Vol. 33, No. 2(A), pp.404-408, Oct 2006, 한국정보과학회.

[4] Sun Microsystems, “K Native Interface Specification “, 2002..

[5] Scott Oaks & Henry Wrong, “JAVA Threads, 2nd Edition Java™ 2”, 1999.

[6] Sun Microsystems, "Inside the JAVA2 Virtual Machine second edition", Jan 2000.

[7] Sun Microsystems, “Mobile Information Device Profile(JSR-37)”, 2000.