韓國 컴퓨터情報學會 2008 제38차 하계학술발표논문집
第16卷 第1號, 2008. 06.

임베디드 시스템을 위한 JAVA API 구현

노 시 영*, 신 성 윤*, 박 상 준*, 이 종 찬*

Implementation of Java API for Embedded System

Si-Young No*, Seong-Yoon Shin*, Sang-Joon Park*, Jong-Chan Lee*

요 약

임베디드 시스템을 지원하는 JAVA API를 개발하면 임베디드 시스템을 개발자가 JAVA를 사용하므로써 코드의 재사용, 객체지향 개념의 시스템 개발들을 가능하게 한다. JAVA API를 구현하는데 있어 시스템에 의존적인 부분들이 존재하게 되는데, 이는 native 함수에서 구현한다. 본 논문에서는 리눅스 기반의 임베디드 시스템 디바이스를 제어하기 위한 JAVA API를 구현하는데 있어 플랫폼 독립적인 자바 부분과 의존적인 native 부분으로 나누어 설계 및 구현하였고, 임베디스 시스템 디바이스의 JAVA API를 통한 제어에 초점을 두었다.

▸ Keyword : JAVA. API, Embedded System(임베디드시스템)

• 제1저자 : 노시영
* 군산대학교 컴퓨터정보공학과

Ⅰ. 서 론

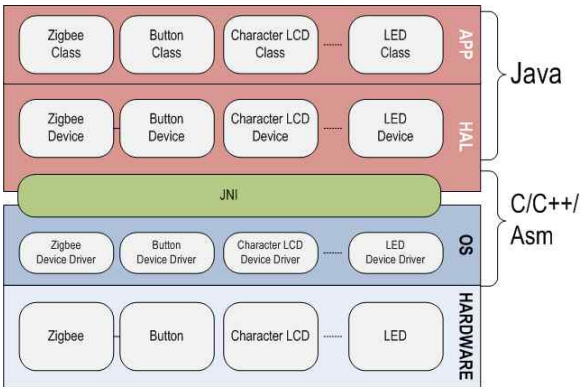
최근 들어 임베디드 시스템 기반의 다양한 어플리케이션이 개발되고 있다. 이러한 임베디드 시스템에는 하드웨어와 소프트웨어가 결합된 고기능의 전자제어시스템을 내장하고 있다. 즉, 단순한 논리회로로 구성된 하드웨어가 아니라, 마이크로프로세서 혹은 마이크로 컨트롤러를 내장하여 제작자가 의도한 다양한 기능을 수행하는 소프트웨어를 포함할 수 있는 시스템이다. 이렇게 임베디드 시스템이 소형화되고, 고성능으로 발전함에 따라 응용 범위도 넓어지고 있 고, 많은 연구들이 진행되고 있다. 이러한 하드웨어 적인 측면의 발전과 달리, 소프트웨어는 아직도 전문적인 기술과 하드웨어에 의존적인 프로그램을 벗어나지 못하고 있다.

본 논문에서는 리눅스 기반의 임베디드 시스템 디바이스를 제어하기 위한 JAVA API를 구현하는데 있어 플랫폼 독립적인 자바 부분과 의존적인 native 부분으로 나누어 설계 및 구현하였고, 임베디드 시스템의 디바이스를 JAVA API를 통해 제어하는데에 초점을 두었다. 플랫폼에 의존적인 native 부분은 JNI(Java Native Interface)를 통해 구현하였다. JNI란 자바와 자바 이외의 언어로 만들어진 애플리케이션이나 라이브러리가 상호 작동할 수 있도록 연결시켜주는 인터페이스로 자바에서 지원하지 못하는 플랫폼 종속적인 기능들과 이미 다른 언어로 만들어진 애플리케이션이나 라이브러리를 사용할 수 있게 한다. JNI를 통한 자바 네이티브 프로그램은 자바가 아닌 다른 프로그래밍 언어들로 자바 객체를 생성하고, 검사하고, 갱신하는 것과 다른 언어에서 자바 메소드 호출을 가능하게 하고 클래스를 로드하고 클래스 정보를 얻을 수 있도록 하는 것 그리고 런타임 타입 체크를 수행하는 것 등을 가능하게 해준다.

본 논문에서는 임베디드 시스템의 플랫폼에 상관없이 어플리케이션에서 이용할 수 있는 HAL 관련 JAVA API를 추가하였다.

Ⅱ. 자바 API 구현 및 시스템 실험

임베디드 시스템을 위한 자바 API와 고려사항으로 본 논문에서 제안한 자바 API의 구조는 두 부분으로 구성된다(<그림1> 참조). 하나는 API를 구성하는 인터페이스와 클래스들을 제공하는 자바 부분(*.java)이고 다른 하나는 각각의 디바이스들을 제어하기 위한 플랫폼 의존적 시스템 호출 위해 JNI 형식을 사용하여 C나 C++로 구현한 네이티브 라이브러리이다. HAL은 하위단인 디바이스와 상위단인 자바 어플리케이션의 인터페이스를 수행함으로서 상위 계층이 하위 계층과 독립적인 프로그래밍이 가능하도록 지원한다.



<그림1> JAVA API 구조

JNI와 디바이스 드라이버으로 리눅스 기반의 디바이스를 제어하기 위하여 디바이스를 커널에 포팅시키기 위한 드라이버를 제작해야 한다. 이를 위하여 디바이스 드라이버를 모듈 방식으로 제공함으로서 커널의 유연한 확장성을 제공할 필요가 있다.

HAL 관련 API의 구현은 어플리케이션 자바 API에서 하드웨어에 종속적인 JNI에 상관없이 JNI를 이용하기 위해 HAL 자바 API를 설계하였다.

- IDevice.class

디바이스 장치를 나타내는 클래스로 인터페이스로 구현하였다.

- AbstractCharacterLCDDevice.class

AbstractCharacterLCDDevice 클래스는 칩셋 이름, CLCD의 라인수, 라인 당 표시할 수 있는 문자수 지정 속성과 CLCD에 출력할 수 기능을 갖는다.

- IPacket.class

IPacket 클래스는 패킷의 사이즈를 반환하는 메서드를 제공하는 인터페이스이다. 통신 디바이스 관련 클래스는 IPacket 클래스를 상속 받아야 하며, 어플리케이션에서 IPacket 클래스를 구현하여 필요한 데이터를 추출해야 한다.

- ICommunicationDevice.class

통신 디바이스는 OSI 7계층의 데이터 링크 계층과 동일한 기능을 수행한다. Packet 단위로 통신을 함으로 IPacket이라는 인터페이스를 사용하여 클래스를 운용한다. 또한 통신 디바이스 클래스는 패킷의 송수신을 여부를 파악하기 위하여 Runnable 인터페이스를 상속하여 이를 감시토록 한다.

- ICommand.class

디바이스 제어를 위한 명령 인터페이스로 디바이스의 기능을 실행하는 메서드를 제공한다. 디바이스의 제어 명령을 하나의 인터페이스로 묶기 위한 인터페이스로 각각의 제어 명령 하나하나를 이 인터페이스를 통하여 구현한다.

- ITranslator.class

하위 레이어인 데이터통신계층에 쓰이는 Packet과 그 위 계층에서 사용하는 명령 사이를 변환시켜주는 기능을 정의한 인터페이스이다. ICommand를 IPacket형태로 변환시키는 메서드가 제공된다.

문자 디스플레이(CLCD) 관련 API의 구현은 CLCD는 제조사마다 다른 명령어 셋을 사용하여 디바이스를 제어함으로 디바이스가 변경될 때마다 서로 다른 명령어셋을 사용해야 한다.

- LCD_II.class

AbstractCharacterLCDDevice 클래스를 상속받아 어플리케이션에 사용할 수 있는 클래스로 CLCD의 On/Off 및 초기화, 출력 관련 메소드를 제공한다.

- ILCD_IICommandSet.class

LCD II Command Set 관련 상수를 정의한 인터페이스이다.

- CharacterLCDMember.class

ClaracterLCDMember 클래스는 CLCD에 들어갈 문자열을 가지는 클래스로 시작 컬럼, IMemberContent 인스턴스를 속성으로 갖는다.

- CharacterLCD.class

CharacterLCD 클래스는 총 슬롯 개수, 슬롯 추가 삭제 시 사용되는 카운터, 해시 테이블을 이용한 맴버 그룹 저장소, AbstractCharacter- LCD Device 객체를 속성으로 갖는다. member, memberGroup의 추가/삭제 메소드, member의 CLCD 출력 메소드를 제공한다.

- IMemberContent.class

member에 포함되는 컨텐츠 인터페이스로 문자열이 저장된다.

- CharacterLCDMemberGroup.class

CharacterLCDMemberGroup는 CLCD의 라인수와 CharacterLCDMember 인스턴스를 갖는다.

지그비 관련 API의 구현은 클래스를 구현하기 위해서는 먼저 지그비 기반의 패킷 전송을 위한 패킷 분석 방법과 상위단인 자바에서 이를 제어하기 위한 최소한의 JNI 로직이 필요하다.

- CC2420.class

CC2420 클래스는 ICommunicationDevice 클래스를 구현해야 한다. cc2420 지그비 컨트롤러를 초기화하고, JNI를 통하여 테이터를 송수신 할 수 있는 메소드를 제공한다.

버튼 이벤트 관련 API의 구현은 버튼 장치의 경우는 OS에서 버튼 클릭 시, Event를 발생시켜 자바의 다른 EventListener와 동일하게 구동해야 하지만 디바이스 특성상 완전한 EventListener 형태로 구현하기는 어렵다. 따라서 특정 옵저버 패턴을 사용하여 주제(Subject)역할을 하는 ButtonHook 클래스를 구축하였고, 실제 디바이스를 감시하는 PollThread를 사용하여 디바이스 감지 ButtonEvent를 발생시킨다.

- ButtonEvent.class

버튼의 클릭 시, 발생되는 이벤트를 나타내는 클래스로 EventObject를 상속받는다.

- ButtonEventListener.class

버튼 이벤트를 수신하는 버튼 이벤트 리스너 인터페이스로 EventListener를 상속 받는다. 버튼 클릭 시, 수행되는 메서드인 button- Pressed()와 버튼의 클릭 해제 시 수행되는 메서드인 buttonReleased()가 제공된다.

- ButtonHook.class

ButtonHook 클래스는 버튼의 클릭 여부 감시자로 버튼 클릭을 감지하고자 하는 인스턴스이다. 이 클래스를 생성 후 addEventListener로 자신을 추가해야 한다.

- PollThread.class

PollThread 클래스는 Thread를 통한 지속적인 버튼 감시를 통하여 버튼 상태를 체크하는 클래스이다. 장치를 감시하기 위한 JNI 루틴이 포함된 메서드로 다른 장치를 사용할 경우 이 클래스를 참조해야 한다.

실험 환경은 호스트 컴퓨터와 타겟 컴퓨터로 구성된다. 호스트 컴퓨터는 RedHat FC4 운영체제에서 gcc와 J2SDK 1.4 버전의 개발툴을 이용하였다. 타겟 컴퓨터는 하이버스의 XHyper270-TKU를 이용하여 임베디드 시스템용 JAVA API를 개발하였고, 임베디드 시스템의 JVM은 JamVM을 이용하였다. JamVM은 GNU Classpath 자바 클래스 라이브러리를 이용하게 설계되었다.

자바 API를 구현하였으므로, 이들 API가 올바르게 동작하는지를 알아보기 위하여 간단한 임베디드 어플리케이션으로 테스트를 한다.

<그림2>에서 CLCDTest는 테스트용 클래스로 main 메소드만을 가진다. ①은 HAL 관련 자바 API인 AbstractCharacterLCDDevice 클래스를 상속받아 어플리케이션에 사용할 수 있는 네이티브 LCD_II 객체를 생성한다. LCD_II 클래스는 CLCD의 On/Off 및 초기화, 출력 관련 메소드를 제공한다. ②는 네이티브 LCD 객체를 인수로 하여

어플리케이션 영역에서 사용할 수 있는 CLCD 객체를 생성한다. ③은 CLCD 화면에 출력할 2줄의 컨텐츠을 생성한다. ④⑤⑥에서는 멤버그룹을 등록하고, 생성된 컨텐츠를 멤버그룹에 등록한다. ⑦은 CLCD에 출력하는 메소드이다.

```
public class CLCDTest {
  public static void main(String[] args){
    AbstractCharacterLCDDevice nativeLCD = new
LCD_II(); ---①
    CharacterLCD clcd = new CharacterLCD(2,
nativeLCD); ---②
    IMemberContent line1Content = new
LineContent("Line1:Hello World"); ---③
    IMemberContent line2Content = new
LineContent("Line2:USNLAB");
    CharacterLCDMemberGroup Line1 = new
CharacterLCDMemberGroup(0, 16, ""); ---④
    CharacterLCDMemberGroup Line2 = new
CharacterLCDMemberGroup(1, 16, "");
    Line1.addMember(new
CharacterLCDMember(0,line1Content)); ---⑤
    Line2.addMember(new CharacterLCDMember(0,
line2Content));
    clcd.addMemberGroup(Line1); ---⑥
    clcd.addMemberGroup(Line2);
    clcd.displayAll(); ---⑦
    try {      Thread.sleep(1000);    }
    catch (InterruptedException e) {
e.printStackTrace();  }
    clcd.cursorOnOff(true);
  }
}
```

<그림2> CLCDTest.java

<그림3> 임베디드 시스템 결과 화면

<그림3>은 XHyper270TKU에 임베디드용 자바 API를 이용하여 제작한 CLCDTest를 실행한 화면이다.

본 논문에서 연구한 임베디드용 자바 API를 활용하면 다양한 임베디드용 어플리케이션을 개발할 수 있다.

참고문헌

[1] http://java.sun.com/javase/6/docs/technotes /guides /jni/spec/jniTOC.html

[2] Java(TM) Native Interface: Programmer's Guide and Specification by Sheng Liang, Sun Microsystems

[3] http://www.xml.com/ldd/chapter/book/

[4] http://www.stanford.edu/class/ee281/handouts /hd44780_lcd_controller_datasheet. pdf

[5] http://jamvm.sourceforge.net/