# Nano-X Window System 기반의 모바일 플랫폼 개발을 위한 에뮬레이터 설계 및 구현에 관한 연구

# A study of Emulator Design and Implementation that Based on Nano-X Window System for Development of Mobile Platform.

윤지훈, 채영훈, 문승진
수원대학교 IT대학 컴퓨터학과

Ji-Hoon Yun, Young-Hoon Chae, Seung-Jin Moon
Dept. of Computer Science, The University of Suwon

E-mail : jhyoon01@suwon.ac.kr, yhchae@suwon.ac.kr, sjmoon@suwon.ac.kr

## 요 약

Java Virtual Machine이 기반이 되는 플랫폼 사용으로 모바일 폰 Java는 C/C++로 컴파일 된 Binary File에 비해 속도가 떨어지고 고성능의 프로세서를 필요로 하기 때문에 가격이 비싸질 수밖에 없다. Native Binary를 사용하는 Nano-X Window System Graphic Engine은 저 사양 프로세서에서 사용가능한 모바일 플랫폼으로써 Java Virtual Machine 보다 빠른 속도를 구현할 수 있고 GPL License를 따르기 때문에 생산단가도 절약할 수 있어 저가형 핸드폰의 대량 생산으로 인해 현재 떠오르고 있는 신흥시장에서 보다 경쟁력을 높일 수 있을 것으로 기대된다.

본 논문에서는 기존의 모바일 플랫폼과 Nano-X window System을 비교해보고 모바일 플랫폼으로써의 개발 방향에 대해 논해 보려 한다.

## 1. 서론

과거의 단순 음성, 텍스트 기반의 서비스에서 멀티미디어 중심으로 서비스가 변화하면서 국내의 휴대 단말기 보급의 빠른 증가와 맞물려 모바일 시장이 지속 적인 발전을 이루고 있다.

이러한 발전 가운데 BREW, GVM, KVM 등 플랫폼의 다양화로 인해 독자적인 콘텐츠 개발에 따른 고가의 이용요금, 다양한 콘텐츠의 부족, 인터페이스 호환성 문제, 운용의 불편함 등이 야기되었고 이러한 문제점을 해결하고자 개발된 WIPI도 SUN사의 의존도가 높고 로열티를 부담하고 있으며 새 플랫폼 적용에 따른 기술적 어려움과 제조사와의 협조 문제, WIPI의 도입 때와 달리 최근 정통부의 의지가 퇴색하면서 WIPI도 대안으로 완벽하다고 볼 수 없게 되었다.

그 대안으로 Native Binary를 사용하고 Device Independence의 특성을 가진 The Nano-X Window System을 이용한 플랫폼을 제시해 보고자 한다.

본 논문 2장에서는 현재 사용되고 있는 모바일 플랫폼과 The Nano-X Window System을 비교 각각의 장단점에 대해 논해보고 3장에서는 The Nano-X Window System의 플랫폼 개발 방향, 실제 Microsoft Windows에서 사용가능한 Emulator를 개발 방향과 구현에 대해 논하고 4장에서는 이에 따른 결론과 개발 과정의 문제점을 서술하고자 한다.

## 2. Mobile Platform

우리가 컴퓨터를 사용하려고 하드웨어 장비를 움직이는 데는 운영체제가 없으면 하드웨어 장비는 단순히 전기적인 장치뿐이고 우리가 사용하지 못할 것이다. 이렇게 하드웨어 장비에 정상적인 사용을 위해 만들어지는 환경을 플랫폼 이라고 한다.

모바일 플랫폼은 컴퓨터가 아닌 기존의 OS개념과 유사한 단말기 미들웨어를 이야기 한다.

각 이동통신 사업자의 입맛에 맞게 모바일 플랫폼은 상당히 많은 종류의 새로운 구조들로 태어났다.

실행 플랫폼은 VM(Virtual Machine) 기술은 프로그래밍 언어에 따른 개발 기준으로 크게 Java와 C 계열의 두 가지로 구분할 수 있다.[1]

| JAVA | C |
|---|---|
| 개발자 기반 넓음, 보안성 우수, 메모리 부담, 느린 처리속도, 고급형 단말기 요구, 전 세계적인 대세 | 적은 메모리 사용, 빠른 처리속도, 보안성 취약, 보급형 단말에 적용가능, 아직 한국, 일본에만 특화 |

표 1 JAVA와 C기반 VM 비교

| | | | |
|---|---|---|---|
| KVM | Java | Interpreter | LGT(SUN) |
| XVM | Java | Interpreter | SKT(XCE) |
| MAP | C/C++ | Binary | SKT(신지) |
| GVM | C/C++ | Interpreter | KTF(모빌탑) |
| BREW | C/C++ | Binary | KTF(퀄컴) |
| WIPI | Java,C,C++ | Binary, Complier | 이통3사,TTA |

표 2 모바일 Platform 기술현황

### 2.1 KVM

KVM은 Kilobyte Virtual Machine의 약자로 기존 자바 가상 머신 (Personal Java VM, Embedded Java VM, JVM)의 서브셋이다. 적은 메모리의 용량과 CPU 처리 능력 그리고 느린 무선 네트워크에 맞게 설계되었고, 그 크기는 45~70KB 정도이다. 기존 유선 환경의 컨텐츠 마이그레이션이 용이하고 다수의 자바 개발자로 인한 컨텐츠 생산 및 확보가 편리하며 자바 특성상 유선과의 연동이 유리해 유 · 무선 연계 서비스가 가능하다는 점이 장점이다.

하지만 인터프리터 방식에 따른 순차적인 컨텐츠의 해석과 운영으로 시스템 성능을 급격히 저하시켜 매우 제한적 성능을 가짐으로 인한 경제적인 모바일 환경 구축에 많은 제약을 주고 일찍 시장에 선보인 것치고는 탑재되어 있는 단말기 대수가 적고 가장 적은 용량, 적은 컬러를 담아내서 모바일 게임 같은 경우 퀄리티가 떨어지는 단점이 있다. [2][3][4][5]

### 2.2 MAP

MAP는 Mobile Application S/W Plug-in Service의 약자로 VM이 가지고 있는 느린 속도와 그래픽 사운드 등 한정도니 표현의 웹브라우저 게임 한계를 극복할 수 있다. 또한 온라인이나 오프라인 등 어떤 형태의 컨텐츠 운영도 가능하며 단말기 고유의 소프트웨어와 밀착된 형태로 상대적으로 빠른 구동을 위해 자체적으로 이미지, 사운드 컨버트 지원도 가능하다.

하지만 특정 단말기의 포팅 지원에 따른 제약과 운영 환경과 개발 환경이 어려워 현재는 사용자의 규모가 크지 않다. [2][3]

### 2.3 GVM

GVM(General Virtual Machine)은 신지소프트가 국내 순수 기술로 개발한 mini C(mobile C) 기반으로 설계된 VM형 플랫폼이다. mobile C로 작성된 어플리케이션은 단말기에 최적화되고 그래픽 처리를 간소화하여 실행속도 면에 있어서 상당히 빠르다. 아울러 120여개의 내장 라이브러리를 제공하며, 확장 U 코드 채택을 통해 운영체제에 독립적인데다 CDMA, GSM, GPRS, PDC 등 모든 무선 전송 규격을 지원하여 강력한 멀티미디어 기능을 제공한다.

하지만 GVM을 이용하는 콘텐츠 사업자는 매출의 4.5%의 로열티를 내야하며 자체 저장영역이 작고 인터프리터 방식을 이용함으로 인해 제한적 성능이 문제가 된다. [2][3][4][5]

### 2.4 BREW

BREW는 Binary Runtime Environment for Wireless의 약자로 CDMA 용 무선 장치들을 위한 Qualcomm의 응용프로그램 개발용 플랫폼으로서, 원시코드가 개방되어 있다. BREW는 응용프로그램과 칩의 운영체계 사이에서 동작하므로, 응용프로그램이 시스템 인터페이스를 코딩하지 않는 것은 물론, 심지어 무선 응용프로그램에 대한 아무런 이해 없이도 그 장치의 기능들을 사용할 수 있다.

이러한 장점에도 불구하고 2004년 한해 6,500억원(단말기 대당 5% 선)에 달하는 많은 로열티가 해외로 나가고 있는 실정이다. [2][3][4][5]

### 2.5 WIPI

WIPI는 Wireless Internet Platform for Interoperability의 약자로 이동통신 단말기에 탑재되어 응용 프로그램에 실행환경을 제공하는 모바일 플랫폼에 대한 표준 규격이다. 각 사업자의 별도 플랫폼에 따른 개발의 중복성, 사용자 선택권 제한 등을 해결하고자 플랫폼의 표준화 필요성이 높아져 2001년 7월 한국무선인터넷 표준화 포럼 내에 모바일 플랫폼 특별 분과가 신설되어 표준화가 추진된 이래 2002년 2월 WIPI V1.0 규격이 , 2004년 2월 V2.0이 발표됐다.

장점으로는 표준 규격을 정하여 각 사업자들의 개발 환경을 복잡하지 않고 간결하게 해주었으며 멀티태스킹이라는 강력한 기능을 제공한다. Java Virtual Machine을 올려서 사용할 수 있도록 해주어 Native Binary와 Java모두 개발 언어로 사용할 수 있도록 해 주었다.

하지만 Sun과의 저작권 분쟁으로 MIDP 규격이 필수로 포함되게 되어 결국 Sun에 주는 로열티는 피할 수 없게 되었다. 또 다른 플랫폼에 비해 아직 관련 개발 인력이 미흡한데다 추진 및 책임 주체도 상대적으로 느슨한 상태이고 이동통신사별로 별도의 확장기술을 채택할 경우 콘텐츠의 자유로운 호환은 보장 받기 어려울 것이다. [2][3][6][7]

### 2.6 The Nano-X Window System

The Nano-X Window System은 Open Source Project의 일환으로 Microsoft Windows나 X-Windows 같은 Graphical Windowing System을 지향하고 있다. 하지만 Microsoft Windows나 X-Windows보다 적은 용량과 적은 메모리를 필요로 하기 때문에 Embedded 환경을 위한 플랫폼으로 현재에는 포켓 PC등의 플랫폼으로 사용되고 있다.

장점으로는 GPL Lisence를 따르는 무료 Open Source 플랫폼이며 독자적인 Graphics Engine을 통한 Device Independent 즉징을 가지고 있어 제작 및 어플리케이션의 확장이 자유롭다. 또한 서버/클라이언트 방식을 지원하여 멀티태스킹 기능이 가능하고 컴파일 시 수백 KB의 작은 용량만을 필요로 하기 때문에 메모리의 활용도가 더욱 높아지게 된다. The Nano-X Window System은 Nano-X API와 함께 Win32 API와도 유사한 API를 지원함으로써 X-Window 프로그래밍에 익숙한 개발자와 Win32 API 프로그래밍에 익숙한 개발자 모두가 쉽게 개발 할 수 있는 환경을 제공한다. 또한 Native Binary형식의 파일을 지원하고 있어 Java Virtual Machine에서의 구현 속도보다 빠른 실행을 할 수 있게 된다. [8]

## 3. Mobile Platform의 개발

### 3.1 Mobile Platform 개발을 위한 API 제작

The Nano-X Window System에서 제공하는 함수를 사용하여 Mobile Phone 전용 API의 제작이 필요하다.

제작해야 할 분야는 크게 두 가지로 나눌 수 있다.

첫 번째는 Handset Hardware의 제어를 API로 써 HAL(Handset Adaptation Layer)을 통해 모바일 폰의 사운드, 진동 등의 제어를 위한 API의 작성이 필요하다.

두 번째는 그래픽 부분에 대한 API의 작성이다. The Nano-X Window System에서 제공하는 별도의 Widget이 없지만 FLNX[9], TinyWidget [10] 은 일반 윈도우용 Application에 적합하게 구성되어 있으므로 모바일 폰 전용 Widget의 제작이 필요하다.

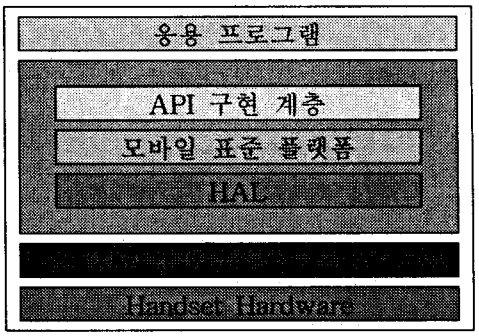


그림 1 모바일 폰의 계층도

### 3.2 Application 개발을 위한 Emulator 환경 구축

The Nano-X Window System은 리눅스에서 최적화 되어 있지만 리눅스 환경에서의 개발자가 적기 때문에 Microsoft Windows 환경에서 실제의 모바일 폰과 동등한 상태에서 테스트 할수 있는 Emulator 개발이 필요로 하다.

이를 위해서 선행되어야 할 과제는 단말기마다 다른 H/W 및 기본의 S/W 사양에 관계없이 Basic API Set나 Extened API Set을 구현하기 위한 일종의 장치 드라이버라고 할수 있는 HAL의 제작이다.

HAL이 Target Machine에 Porting되면 Hardware를 제어 하기위한 기능을 하게 되고

Microsoft Windows 환경에서 Porting하게 되면 Emulator를 위한 기능을 수행하게 된다.

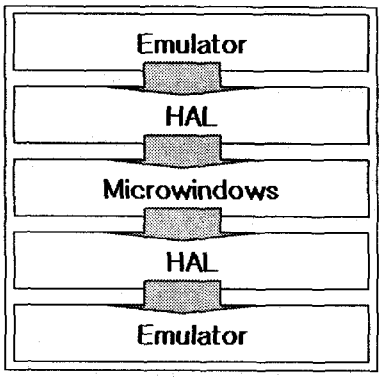


그림 2 Emulator 처리 순서

Emulator는 그림 2와 같이 Emulator에서 받은 입력 값은 HAL을 거쳐서 The Nano-X Windows System에서 처리 후 다시 HAL을 거쳐서 Emulator에서 출력하는 방법을 사용한다.

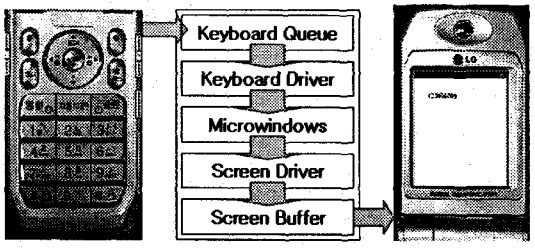


그림 3 실제 Key 입력 시 화면에 출력되는 처리순서

그림 3에 보여지 는 것처럼 Emulator 상의 Key 값은 입력 시 HAL의 Queue에 저장된다. 이는 The Nano-X Window System의 Driver를 거쳐서 다시 Emulator 상의 Screen Buffer에 출력하게 된다.

### 3.3 Emulator를 이용한 응용 어플리케이션 개발

실제 Emulator 이용하여 Key 입력과 Screen 출력을 이용하여 모바일 폰의 기본적인 기능인 알람, 계산기, 주소록 등을 구현 할 수 있다.

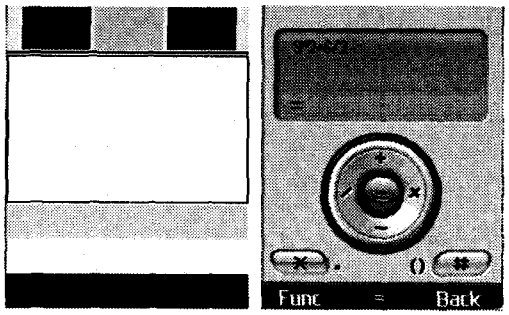


그림 4 주소록 그림 5 계산기

## 4. 결론 및 연구과제

The Nano-X Window System은 현재까지는 모바일 폰을 위한 플랫폼으로써의 개발 환경이 완벽하게 갖추어진 상태는 아니지만 Open Source로써 무한한 발전 가능성을 내재하고 있고 GPL Lisence를 통한 무료 플랫폼 이라는 메리트를 가지고 있고 작은 용량만을 필요로 하여 메모리의 효율도 커지게 되어 모바일 폰 생산 비용의 절감효과가 기대된다.

또한 Win32 API 개발자와 X-Windows 개발자 모두를 위해 익숙한 개발환경을 제공하고 있어서 미래에 많은 개발자들이 모일 것이라는 기대를 하고 있다.

하지만 그전에 The Nano-X Window System이 Mobile Platform으로써의 가능성을 보여주기 위해서 3장에 설명한 API Set의 구현을 계획하고 Microsoft Window 상태에서의 Emulator와 응용 Application을 개발 중에 있다.

## 5. 참고문헌

[1] 한국경제 2004년 3월 11일
http://www.roxia.co.kr/prroom/press_view.php?idx=3&page=1

[2] 배석희 : 모바일 플랫폼 표준화 동향 및 향후 발전방향

[3] 김창환 : 모바일 플랫폼 표준화 동향

[4] 강상원, 임석진, 심양섭 : 위피 프로그래밍

[5] 월간 디지털 콘텐츠 10월호
http://exit7.co.kr/blog/?no=175

[6] 권기경 : WIPI 플랫폼 개발 기술 요소 및 서비스 환경

[7] 김정훈 : 표준 무선 인터넷 플랫폼 WIPI 기술 및 표준화 동향

[8] The Nano-X Window System :
http://www.microwindow.org/microwindows_architecture.html

[9] Fast Light Toolkit :
http://www.fltk.org

[10] The TinyWidgets Project :
http://www.tinywidgets.sourcefore.net/