중소기업정보기술융합학회 논문지 제2권 제1호 pp. 51-58, 2012

안드로이드 플랫폼에 WiFi 디바이스 탑재 기법

정우영[1], 주영관[2], 전중남[2*]
[1]충북대학교 컴퓨터과학과, [2]충북대학교 전자정보대학

A Porting Technique of WiFi Device on Android Platform

Uyeong Jeong[1] and Youngkwan Ju[2] and Joongnam Jeon[2*]
[1]Department of Computer Science, Chungbuk National University
[2]Department of Electrical & Computer Engineering, Chungbuk National University

요 약 안드로이드 플랫폼은 리눅스 2.6 커널을 기반으로 강력한 운영체제와 포괄적 라이브러리, 멀티미디어 환경, 사용자 인터페이스, 폰 애플리케이션 등을 제공한다. 안드로이드는 개방형 운영체제이기 때문에, 어느 벤더기기에든 탑재가 가능하다. 현재 스마트폰뿐만 아니라 넷북, 네비게이션, 카 PC, 태블릿 PC, 산업용 PC 등 여러 분야에서 사용되고 있다. 안드로이드를 다른 기기에 탑재하거나 안드로이드 플랫폼에 새로운 디바이스를 탑재하려면 많은 어려움이 따른다. 본 논문에서는 하드웨어 장치에서 발생한 데이터가 최상위 애플리케이션까지 전달되는 과정과 안드로이드 플랫폼이 하드웨어 디바이스를 관리하는 체계를 분석하고, WiFi 디바이스를 탑재하는 절차를 안드로이드 및 드라이버 컴파일 환경구축, 커널에서 WiFi 사용을 위한 프로토콜 지원, WiFi 디바이스를 커널에 탑재, 안드로이드 플랫폼에 디바이스 드라이버 등록, WiFi 관리서비스 데몬(wpa_supplicant)과 IP 할당서비스 데몬(dhcpcd) 등록, 데몬(wpa_supplicant)과 HAL의 통신을 위한 소켓 생성으로 제시하고 있다. 실험에서는 본 논문에서 제시한 방법을 이용하여 ARM 계열과 X-86 계열의 안드로이드 플랫폼에 WiFi 디바이스를 탑재했다. 안드로이드 플랫폼에 디바이스 탑재 시에는 안드로이드의 소프트웨어 계층을 이해하는 것이 매우 중요하며, 이러한 경험은 안드로이드 플랫폼에 새로운 디바이스를 탑재할 때에도 많은 도움이 될 것이다.

키워드 : WiFi, 안드로이드, WiFi 디바이스 탑재

Abstract Android platform is a powerful operating system developed on Linux 2.6 Kernel, and provides many features such as comprehensive libraries, a multimedia environment, and powerful interface for phone applications. Since Android is an open operating system, which can be installed in any vendors's equipments. Current smartphones as well as netbooks, navigations, car PCs, tablet PCs, Industrial PCs are used in various fields. It is difficult a lot that to mount to other devices on the Android platform or new devices. In this Paper, The process that data that occurred from a hardware was passed to the highest application and Android platform system for managing hardware devices were analyzed. Building Android & driver compilation environment, How to support the protocol for the use of WiFi in the kernel, How to Mount a WiFi device in the kernel, Device driver registration for the Android platform, WiFi Management Service Daemon (wpa_supplicant) and IP allocation services daemon (dhcpcd) registration, How to create a socket for communication between the daemon (wpa_supplicant) and HAL have been presented. In the experiment using the proposed method, WiFi devices were mounted on the Android platform in the X-86 & ARM family. Understanding the whole process of control flow in Android hierarchy is very important to porting a new device on it. The process included in this paper can help technicians who might encounter the obstacles in their porting works.

Key Words : WiFi Android, A Porting WiFi Device

*교신저자(joongnam@cbu.ac.kr)
접수일(2012년06월15일), 심사완료일(2012년06월30일)

1. 서론

2007년 11월 세계 각국의 이동통신관련 회사연합체인 OHA(Open Handset Alliance)가 안드로이드 플랫폼을 공개하였고, 세계적 검색엔진 업체인 구글이 안드로이드를 인수하여 개발하였다. 안드로이드 플랫폼은 리눅스 2.6 커널을 기반으로 강력한 운영체제와 포괄적 라이브러리, 멀티미디어 환경, 사용자 인터페이스, 폰 애플리케이션 등을 제공한다. 안드로이드는 개방형 운영체제이기 때문에, 어느 벤더기기에든 탑재가 가능하다. 스마트폰뿐만 아니라 넷북, 네비게이션, 카 PC, 태블릿 PC, 산업용 PC등 여러 분야에서 사용되고 있다[1,2].

안드로이드 플랫폼 개발자들은 하드웨어를 제어하는 계층보다 상위 애플리케이션 개발에 치중하고 있고, 애플리케이션 개발에 필요한 정보, 자료, 관련서적은 손쉽게 구할 수 있다. 그러나 안드로이드를 다른 기기에 탑재하거나 안드로이드 플랫폼에 새로운 디바이스를 탑재하려면 많은 어려움이 따른다. 안드로이드 플랫폼의 하드웨어 제어에 관련된 정보나 관련 서적은 극히 드물기 때문이다.

본 논문은 하드웨어 장치에서 발생한 데이터가 최상위 애플리케이션까지 전달되는 과정과 안드로이드 플랫폼이 하드웨어 디바이스를 관리하는 체계를 분석하고 WiFi 디바이스를 탑재하는 기법을 제시한다.

2. 안드로이드 제어 흐름

이 장에서는 안드로이드의 디바이스 제어흐름과 WiFi 디바이스가 안드로이드에서 제어하는 흐름에 대해서 기술한다.

2.1 안드로이드 계층구조

이 절에서는 안드로이드 계층의 구조에 대해서 기술한다.

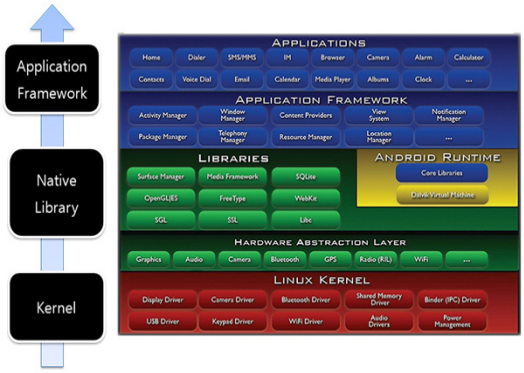

그림 1. 안드로이드 계층구조
Fig 1. Android layer structure

안드로이드는 그림 1에서 보는 바와 같이 크게 3영역으로 구분할 수 있고, 작게는 6영역으로 나눌 수 있다. 크게는 애플리케이션 계층, 라이브러리 계층, 커널 계층으로 나눌 수 있고 작게는 애플리케이션스 계층, 애플리케이션 프레임워크 계층, 안드로이드 런 타임 계층, 라이브러리 계층, HAL, 리눅스 커널 계층으로 나눌 수 있다. 각각의 계층을 최상위 계층부터 간략하게 살펴보면 다음과 같다.

- Applications : 첫 화면(Home), 주소록, E-mail 클라이언트, SMS 프로그램, 달력, 지도, 브라우저, 전화번호부, 그리고 다른 것들을 포함하는 핵심 애플리케이션들이 탑재되어 있고 다른 애플리케이션들을 설치하여 사용하는 유저영역에 해당된다. 모든 애플리케이션은 Java 프로그래밍 언어로 작성된다.
- Applications Framework : 일련의 시스템 서비스나 관리자 같은 것으로, 시스템 관리자는 패키지를 자원을 관리하거나 디바이스가 현재 있는 위치를 알려주는 것과 같은 매우 다양한 서비스를 제공한다. 개발자들이 애플리케이션 프레임워크 API에 완벽하게 접근할 수 있다. 그리고 Java기반의 프레임워크이며, 대부분이 JNI(Java Native Interface)로 C/C++ 코드로 작성되어 있다. 위와 같은 서비스들은 핵심 시스템 서비스를 담당하는 코어시스템 서비스들과 하드웨어와의 인터페이스를 담당하는 하드웨어 서비스들로 구성된다.
- Libraries : 안드로이드의 다양한 컴포넌트에 의해 사용되는 Java와 인터페이스를 하기 위한 C/C++

라이브러리 집합이다. 이러한 라이브러리의 기능들은 안드로이드 애플리케이션 프레임워크를 통해서 개발자들에게 제공된다.

- Android Runtime : 달빅이라 불리는 가상머신(VM)과 기본 제공하는 코어 라이브러리이다. 달빅은 모바일에서 복수의 가상머신을 효율 좋게 실행 가능하도록 설계되어 있고, 메모리의 사용량을 최소로 하는 것으로 최적화된 파일(.dex)을 실행한다. Java언어와 C++언어 간의 코드를 변환해주는 기능을 수행한다.
- Hardware Abstraction Layer : 안드로이드와 하드웨어를 연결해주는 인터페이스 역할을 하고 있는 계층이다. C/C++ 라이브러리 형태로 존재하고 표준화된 API로 구성되어 있다. 기존 임베디드 리눅스는 디바이스 드라이버들이 커널영역(Kernel Level)에서 동작하지만, 안드로이드의 디바이스 드라이버들은 유저영역(User Level)에서 동작한다. 안드로이드에서 드라이버들을 유저영역에 넣은 이유는 라이선스 문제와 안정성, 플러그인 방식으로 드라이버를 배포 할 수 있기 때문이다.
- Linux Kernel : 표준 리눅스 커널을 기반하고 있으나, 표준 리눅스는 아니다. X-Window와 같은 내장 윈도우 시스템을 포함하지 않으며, glibc를 지원하지도 않고 표준 리눅스 유틸리티 전체를 포함하지 않는다. 안드로이드는 리눅스 커널 버전 2.6.23, 2.6.24, 2.6.25, 2.6.27을 사용해 왔으며 안드로이드 지원을 위해 리눅스 커널 확장을 위한 패치를 포함한다. 안드로이드에서 리눅스를 사용하는 이유는 메모리 및 프로세스 관리, 퍼미션(Permission) 기반의 보안 모델, 검증된 드라이버 모델, 공유 라이브러리 지원, 오픈 소스 기반 등의 장점 때문이다.

2.2 안드로이드 디바이스 제어흐름

안드로이드에서는 디바이스들을 프레임워크 서비스를 통해서 제어하는데, 디바이스를 탑재하려면 안드로이드가 지원하는 서비스들을 알고 있어야 한다. 안드로이드 서비스는 크게 자바 시스템 서비스와 네이티브 시스템 서비스로 나눌 수 있다. 네이티브 시스템 서비스는 C++로 작성되어 있으며 라이브러리 계층에서 동작한다. 자바 시스템 서비스는 안드로이드 부팅 시 시스템 서버라는 시스템 프로세스에 의해서 일괄적으로 실행된다. 안드로이드는 기본적인 서비스 데몬을 제공하는 것도 있지만, 기존의 리눅스용 데몬(Daemon) 프로그램을 안드로이드가 사용하는 경우도 있다. 이러한 경우에 해당하는 디바이스에는 RILD(Radio Interface Layer Daemon), WiFi, 블루투스가 있다[4]. 안드로이드 플랫폼에 디바이스 드라이버를 탑재할 때에는 안드로이드 서비스의 흐름을 이해할 필요성이 있다.

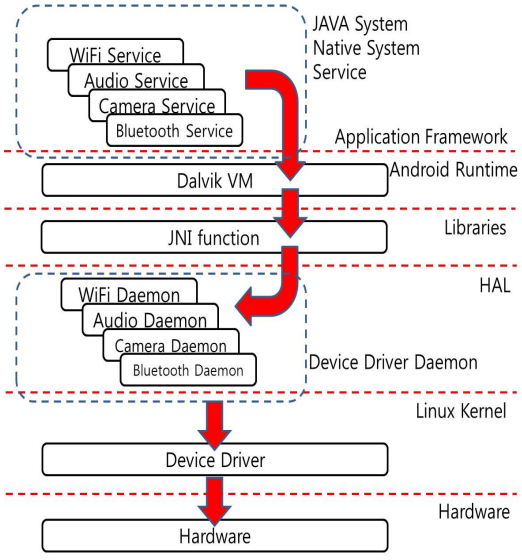

그림 2. 안드로이드 디바이스 제어흐름
Fig 2. Android Device Control Flow

안드로이드는 그림 2에서 보는 바와 같이 안드로이드는 블루투스, 카메라, 오디오, WiFi 등을 서비스 형태로 관리하는데, 이 서비스들은 데몬을 통해서 디바이스를 제어한다. Java 코드로 이루어진 애플리케이션 계층의 서비스를 호출하면 달빅을 통해서 C++ 코드로 변환되고, JNI 함수를 통해서 HAL(Hardware Abstract Layer)에 전달된다. 이 계층에서는 디바이스 드라이버를 제어하는 데몬들을 관리한다. 그리고 데몬들은 각각의 디바이스 드라이버들을 관리한다.

2.3 안드로이드 WiFi 제어 흐름

이 절에서는 안드로이드에서 WiFi 디바이스 제어흐름에 대해서 기술한다.

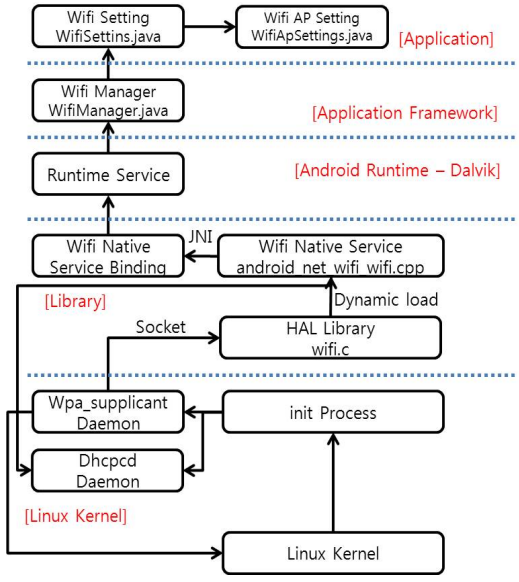

그림 3. 안드로이드 WiFi 제어 흐름
Fig 3. Android WiFi Control Flow

그림 3[3]은 안드로이드에서 WiFi 디바이스를 제어하는 흐름을 나타낸 그림이다. 전원이 인가되면 리눅스가 부팅되고, 리눅스커널에서 디바이스에 맞는 드라이버가 적재된다. 커널이 부팅된 후에 init 프로세스가 실행되는데 init 프로세스에서 wpa_supplicant 데몬과 dhcpcd 데몬을 실행한다. wpa_supplicant 데몬은 WiFi 보안과 접속을 관리하는 데몬이고, dhcpcd는 AP(Acess Point)에서 IP를 자동할당 받는 데몬이다. 안드로이드 플랫폼의 WiFi 세팅 다이얼로그에서 ON 버튼을 누르면 WiFi 매니저를 통해서 WiFi 네이티브 서비스를 호출한다. WiFi 네이티브 서비스는 HAL인 wifi.c에서 드라이버를 로드한다. 이때 HAL은 WiFi 디바이스 드라이버의 상태를 체크하고, WiFi 네이티브 서비스는 wpa_supplicant 데몬을 호출한다. wpa_supplicant 데몬은 HAL과 소켓을 통해서 통신한다. 소켓의 종류는 Android private 소켓과 Unix standard 소켓이 있다. 또 wpa_supplicant 데몬은 리눅스 커널과 WEXT 드라이버를 사용해서 통신한다. wpa_supplicant 데몬은 AP 스캔정보를 WiFi 네이티브 서비스로 전달하고, WiFi 네이티브 서비스는 신호세기 순으로 나열하여 저장된 AP 정보나 입력받은 AP를 등록한다. dhcpcd 데몬은 이 정보를 이용하여 IP를 할당한다. 안드로이드 애플리케이션 계층에서는 WiFi 상태를 ON으로 표기하고 상태표시 바에 WiFi 연결을 표시한다.

3. WiFi 디바이스 탑재 기법

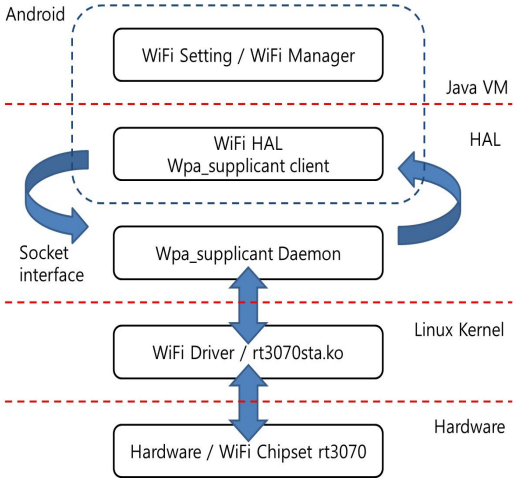

그림 4. WiFi 디바이스 데이터 흐름
Fig 4. WiFi Device Data Flow

이 장에서는 그림 4[5]에서 보는 바와 같이 리눅스 커널, wpa_supplicant, HAL, Application 계층에서 WiFi 디바이스의 데이터 흐름에 따라 안드로이드에 WiFi 디바이스를 탑재하는 방법에 대해서 기술한다.

3.1 안드로이드에 WiFi 설치 과정

이 절에서는 안드로이드 계층에 WiFi 디바이스 드라이버를 설치하는 과정에 대해서 기술한다.

설치단계는 제일 먼저 커널에서 WiFi 사용을 위한 프로토콜을 지원하도록 하고, 칩셋의 리눅스용 드라이버 소스 또는 펌웨어를 구해서 안드로이드 리눅스 커널에 적재한다. 그리고 안드로이드에 커널의 드라이버를 등록해준다. WiFi를 관리해 주는 서비스들을 등록하고, HAL과 wpa_supplicant와 데이터를 주고받을 수 있는 소켓을 생성하면 된다[5,6]. 본 논문에서는 디바이스 제어흐름에 따라서 탑재과정을 설명하는데 계층별 과정을 살펴보면 표 1과 같다.

표 1. WiFi 디바이스 설치 과정
Table 1. WiFi Device installation Procedure.

단계	계 층	목 적	수 행 내 용
1		안드로이드 및 드라이버 컴파일 환경구축	VM설치, 우분투 설치, 빌드 패키지들 설치
2	Linux Kernel	WiFi 사용을 위한 프로토콜 지원	리눅스 커널에 WiFi 지원 설정
3	Linux Kernel	WiFi 디바이스를 커널에 탑재	드라이버 컴파일 및 펌웨어 커널에 등록
4	HAL	안드로이드에 디바이스 드라이버 등록	wifi.c 소스파일 수정
5	HAL	WiFi 관리서비스 데몬(wpa_supplicant) 등록 IP 할당서비스 데몬(dhcpcd) 등록	init.rc 파일에 서비스 등록
6	HAL	데몬(wpa_supplicant)과 HAL의 통신을 위한 소켓 생성	init.rc 파일에 소켓생성 및 권한 부여

표 1의 내용을 단계별로 살펴보면 다음과 같다.

• 1단계 : 이 단계에서는 안드로이드를 컴파일하기위한 환경설정에 대해서 기술한다. 안드로이드는 우분투 커널을 기반으로 하고 있기 때문에 구글에서는 개발환경을 우분투를 권장하고 있다. 우분투에는 개발에 필요한 기본적인 패키지들이 설치되어 있지 않다. 개발에 필요한 패키지들을 apt-get 명령을 이용하여 우분투에 설치해야 한다. 기본적인 개발 패키지 중에서 개발자들이 많이 사용하는 NFS와 TFTP 설치한다. 또 안드로이드는 Java를 기반으로 하고 있기 때문에 Java JDK를 설치한다. 그리고 안드로이드를 빌드하는데 필요한 패키지들이 있다. 이 패키지들은 우분투 버전에 따라서 틀려진다. 표 2는 안드로이드 빌드에 필요한 패키지들을 우분투 버전에 따라서 나타낸 표이다. 그리고 마지막으로 ARM용 크로스 컴파일러를 설치하면 된다.

표 2. 안드로이드 빌드 환경 패키지
Table 2. Android Build Environment Package.

Ubuntu Bit	Ubuntu Version	필 요 한 유 틸 리 티
공 통	공 통	Phython, GIT1.5.4 이상, valgrid
Ubuntu 32bit	Ubuntu 10.10	sudo ln -s /usr/lib32/mesa/libGL.so.1 /usr/lib32/mesa/libG L.so
	Ubuntu 11.10	sudo ln -s /usr/lib/i386-linux-gnu/libX11.so.6 /usr/lib/i386-linux-gnu/libX11.so.6
	Ubuntu 10.10 이하	sudo apt-get install git-core gnupg flex bison gperf build-essential zip curl zlib1g-dev libc6-dev libncurses5-dev x11proto-core-dev libx11-dev libreadline6-dev libg1-mesa-dev tofrodos phython-markdown libxml2-utils xsltproc
Ubuntu 64bit	공 통	sudo apt-get install git-core gnupg flex bison gperf build-essential zip curl zlib1g-dev libc6-dev lib32ncurses5-dev ia32-libs x11proto-core-dev libx11-dev lib32readline5-dev lib32z-dev libg1-mesa-dev g++-multilib mingw32 tofrodos phython-markdown libxml2-utils xsltproc

• 2단계 : 이 단계에서는 WiFi 디바이스를 사용하려면 리눅스 커널의 설정을 기술한다. WiFi 통신을 하려면 표준 프로토콜을 따라야 하는데, 이 표준 프로토콜을 커널에서 지원해주고 있다. 단, 표준 프로토콜 중 WiFi가 어떤 프로토콜인지 확인하고 설정해야 한다. PAN, LAN, MAN을 정의하는 표준은 IEEE 802 프로토콜이다. 이 프로토콜을 커널에서도 지원하게 설정해야 한다. 그리고 커널에서 WiFi 디바이스 드라이버를 지원하도록 설정해야 한다. 설정방법은 커널이 있는 디렉토리에서 make menuconfig를 실행한다. 커널 설정에서 크게 두 가지 부분을 설정해야 하는데 설정 부분은 Networking support와 Device Drivers 부분이다. Networking support는 무선 표준 프로토콜을 설정하는 부분이고, Device Drivers는 각 벤더의 드라이버들을 설정하는 부분이다.

• 3단계 : 이 단계에서는 디바이스 드라이버를 커널에 탑재하는 방법에 대해서 기술한다. 리눅스 커널에서 WiFi를 사용하려면 디바이스 칩셋에 맞는 드라이버 또는 펌웨어가 필요하다. WiFi 드라이버는 커널에서 직접 WiFi 디바이스를 제어한다. 안드로이드 플랫폼은 리눅스 커널을 기반으로 하고 있다. 표준 리눅스 커널은 일반적인 벤더들의 WiFi 드라이버를 지원하고 있지만 모든 벤더들의 WiFi 드라이버를 지원하지는 않는다. 만약 표준 리눅스 커널에서 지원하지 않는 WiFi 칩셋이라면, WiFi 칩셋 제조회사에서 리눅스용 드라이버소스 또는 펌웨어를 웹상에서 다운로드해야 한다. 펌웨어는 안드로이드 플랫폼에서 지정된 위치에 있다면 안드로이드 플랫폼이 부팅될 때 자동으로 펌웨어를 로드하여 커널에 적재한다. WiFi 디바이스 드라이버가 커널에 적재할 때의 형태는

모듈형태와 빌드인 형태가 있다. 모듈 형태는 사용자가 직접 insmod 명령을 사용하거나, 스크립트를 이용하여 커널에 적재를 하고, 빌드인 형태는 커널에 드라이버를 포함하고 있기 때문에 자동으로 적재 된다. 칩셋 제조회사에서 다운로드한 WiFi 드라이버 소스를 사용하려면 소스를 자신의 개발환경에 맞춰 빌드하여 사용하면 된다. 빌드할 때는 자신의 안드로이드 플랫폼의 커널 환경과 크로스 컴파일 부분을 고려하면 된다.

• 4단계 : 이 단계에서는 커널의 WiFi 드라이버를 안드로이드에 등록하는 방법에 대해서 기술한다. 안드로이드에서는 HAL이 하드웨어를 제어한다. HAL은 JNI함수를 통해서 상위 계층에서 WiFi 디바이스를 제어한다. JNI함수 android_net_wifi_wifi.cpp 소스는 HAL소스 wifi.c의 함수들을 직접 호출한다. wifi.c에서는 드라이버를 insmod, rmmod, 펌웨어 로드, wpa_supplicant 접속 등 커널의 WiFi 드라이버를 제어한다. 안드로이드에 WiFi 드라이버를 등록하려면 HAL인 wifi.c를 수정하여 드라이버를 등록해야 한다. HAL에 드라이버를 등록하는 방법에는, 펌웨어를 이용하는 방법, 디바이스 드라이버를 이용하는 방법이 있다. 펌웨어를 등록방법은 wifi.c에서 WIFI_FIRMWARE_LOADER 부분에 펌웨어 파일의 경로를 수정하거나, 안드로이드의 펌웨어를 로드하는 디렉토리에 복사하면 된다. 드라이버 등록방법은 wifi.c에서 WIFI_DRIVER_MODULE_PATH 부분에 디바이스 드라이버의 경로와 파일이름을 수정하면 된다. 펌웨어는 커널계층에서 로드가 이루어져서 사용할 준비가 된다. 그러나 안드로이드에서는 드라이버가 준비된 것을 알 수가 없기 때문에 준비됐다는 상태를 알려주어야 한다. init 프로세스가 동작할 때 init.rc를 참조하는데 이 스크립트 파일 안에 다음과 같이 인터페이스 네임과 드라이버 상태를 setprop wifi.interface "wlan0", setprop wlan.driver.status "ok"를 넣어주면 된다.

• 5단계 : 이 단계에서는 안드로이드에서 WiFi를 관리하는 서비스 데몬들을 등록하는 방법에 대해서 기술한다. 안드로이드에서 WiFi 서비스를 관리하는 것은 wpa_supplicant 데몬이다. wpa_supplicant 데몬은 기존 리눅스에서 사용하는 것으로 안드로이드에서도 기존 데몬을 사용하고 있기 때문에 안드로이드에 데몬 서비스를 등록해야 한다. 그리고 IP를 자동으로 할당 받기 위해서 dhcp 데몬도 서비스로 등록해야 한다. 안드로이드에서 dhcp 데몬 서비스 이름은 dhcpcd 이다. 안드로이드 진저브레드, 즉 버전 2.3부터는 서비스 이름이 dhcpcd_wlan0로 바뀌었다. 서비스를 등록시키는 방법은 init.rc 파일에 서비스를 추가한다. 서비스 등록방법은 서비스 [네임] [실행파일] [옵션] 으로 이루어진다.

• 6단계 : 이 단계에서는 wpa_supplicant 데몬과 HAL과의 통신하는 소켓을 등록하는 방법에 대해서 기술한다. HAL은 WiFi 데이터와 드라이버 관리를 wpa_supplicant 라는 데몬으로 하는데, 여기서 HAL과 wpa_supplicant 데몬간의 통신은 소켓을 생성하여 통신한다. 소켓은 두 가지의 종류가 있다. 하나는 Unix standard socket과 Android private socket이 있다. 두 소켓은 각각 설정 방법이 다르다. 소켓을 무엇으로 사용할지는 wpa_supplicant.conf 파일에서 결정된다. 사용하는 소켓에 따라서 wpa_supplicant 서비스를 등록하는 방법도 다르다.

4. 실험 및 탑재 결과

이 장에서는 3장의 내용의 절차대로 WiFi 디바이스를 ARM 계열과 X-86계열에 탑재한 방법에 대해서 기술하고 있다.

4.1 ARM 계열과 X-86 계열의 플랫폼

이 절에서는 실험에 사용되었던 ARM 계열과 X-86 계열의 보드들의 플랫폼에 대해서 기술한다.

표 3. ARM & X-86 family의 하드웨어 사양
Table 3. Hardware Specification of ARM & X-86 family.

	ARM	X-86
CPU	Samsung S5PV210	Intel ATOM
Storage	NAND	HDD
WiFi 칩셋	RT3070	RT3070
WiFi 타입	USB ipTime N100UM	USB ipTime N100UA

표 3은 ARM 계열[8,9]과 X-86의 하드웨어적 내용을 나타내고 있다. ARM 계열은 삼성의 S5PV210 CPU를 사용하였고 X-86 계열은 인텔의 ATOM CPU를 사용하였다. 저장장치는 ARM 계열은 512 메가 NAND 플래시를

사용하였고, X-86 계열은 250 기가 HDD 를 사용하였다. WiFi 디바이스는 USB 타입으로 같은 회사의 제품이지만 모델명은 다르다. 그러나 두 제품 모두 WiFi 칩셋은 Ralink 사의 RT3070 칩셋을 사용했다.

표 4. ARM & X-86 family의 소프트웨어 사양
Table 4. Software Specification of ARM & X-86 family.

	ARM	X-86
Host OS	Windows XP	Windows XP
VM OS	Ubuntu 11.10	Ubuntu 11.10
Tool Chain	arm-eabi-gcc	gcc
Android Version	2.2 Froyo	2.2 Froyo
Linux Kernel Version	2.6.32.9	2.6.35.7

표 4는 실험에서 사용하였던 소프트웨어적 내용을 나타낸 표이다. 보는 바와 같이 호스트 PC는 윈도우XP를 사용하였고, 호스트 PC안에 다른 운영체제를 설치하기 위해서 가상머신 오라클 Virtual-Box를 사용했다. 안드로이드의 배포처인 구글에서 안드로이드 개발 플랫폼을 우분투로 사용하기를 권장하고 있다. 이에 본 논문에서도 우분투를 개발 플랫폼으로 사용했다. 우분투 버전은 11.10을 사용했고, 안드로이드 버전은 2.2인 프로요를 사용했다. 리눅스 커널 버전과 Toolchain은 각각 다르다. ARM 계열은 arm-eabi-gcc Toolchain과 리눅스 커널 2.6.32.9을 사용했고, X-86 계열은 표준 gcc Toolchain과 리눅스 커널 2.8.35.7을 사용했다.

4.2 rt3070 디바이스 탑재 결과

본 논문 실험에서는 ARM 계열과 X-86 계열에 드라이버 모듈과 펌웨어, 각기 다른 소켓을 이용하여 rt3070 칩셋 USB WiFi 디바이스를 탑재[7]했다. 탑재하고 난 후의 결과로 연구실의 AP로 WiFi를 접속하고, 안드로이드 플랫폼의 웹브라우저를 사용하여 네이버에 접속해 보았다. 그림 5는 ARM 계열과 X-86 계열의 결과이다.

그림 5. ARM계열과 X-86계열의 네이버 접속 결과
Fig 5. Results of Naver connected by WiFi of ARM & X-86 family.

5. 결론

본 논문에서는 안드로이드 디바이스 제어흐름을 분석하고, ARM 계열과 X-86 계열의 플랫폼에 USB 타입의 WiFi 디바이스를 탑재했다.

본 논문에서는 WiFi 디바이스를 탑재하는 절차를 안드로이드 및 드라이버 컴파일 환경구축, 커널에서 WiFi 사용을 위한 프로토콜 지원, WiFi 디바이스를 커널에 탑재, 안드로이드 플랫폼에 디바이스 드라이버 등록, WiFi 관리서비스 데몬(wpa_supplicant)과 IP 할당서비스 데몬(dhcpcd)등록, 데몬(wpa_supplicant)과 HAL 의 통신을 위한 소켓 생성으로 제시하고 있다. 실험에서는 본 논문에서 제시한 방법을 이용하여 ARM 계열과 X-86 계열의 안드로이드 플랫폼에 WiFi 디바이스를 탑재했다. 안드로이드 플랫폼에 디바이스 탑재 시에는 안드로이드의 소프트웨어 계층을 이해하는 것이 매우 중요하며, 이러한 경험은 안드로이드 플랫폼에 새로운 디바이스를 탑재할 때에도 많은 도움이 될 것으로 판단된다. 향후에도 안드로이드 계층, 서비스 분석 및 새로운 디바이스 탑재에 대한 연구를 지속적으로 수행할 예정이다.

참 고 문 헌

[1] B. Speckmann, "The Android mobile platform," emich.edu, 2008.
[2] 정승일, "안드로이드 플랫폼과 스마트폰 기술 발전 동향", 대한전자공학회, 제33권 제1호, 2010.
[3] 정우영, 전중남, "X-86 안드로이드 플랫폼에 Wifi 디바이스 탑재", 한국정보처리학회, 춘계학술발표대회, 논문집 제18권 제1호, 2011.
[4] 송형주, 김태연, 박지훈, 이백, 임기영, "인사이드 안드로이드", 위키북스, 2010.
[5] 고현철, 유형목, "안드로이드의 모든 것 분석과 포팅", 한빛미디어, 2011.
[6] 박선호, 오영환, "안드로이드 시스템 프로그래밍 완전정복", D&W Wave, 2010.
[7] http://whitesc3.blog7.fc2.com/blog-entry-111.html 2009.
[8] http://cafe.naver.com/embeddedcrazyboys 2008.
[9] http://www.aesop.or.kr 2009.

저 자 소 개

정 우 영(Uyeong Jeong) [비회원]

- 2006년 2월: 대전대학교 인터넷정보통신학과 학사
- 2007년 10월 : 모비안 연구원
- 2010년 3월 ~ 현재 : 충북대학교 컴퓨터과학과 석사과정

<관심분야> : Linux Kernel, 안드로이드 플랫폼

주 영 관(Youngkwan Ju) [비회원]

- 1999년 2월: 청주대학교 컴퓨터정보공학과 학사
- 2004년 2월: 충북대학교 전자계산학과 석사
- 2009년 2월: 충북대학교 전자계산학과 박사
- 충북대학교 전자정보대학 강사

<관심분야> : 임베디드시스템, 안드로이드 플랫폼, 클라우드컴퓨팅 등

전 중 남(Joongnam Jeon) [정회원]

- 1990년 연세대학교 전자공학과 공학박사
- 1996년~1998년 미국 Texas A&M 연구교수
- 현재 충북대학교 전자정보대학 교수

<관심분야> : 컴퓨터구조, 임베디드 시스템