

임베디드 시스템의 소프트웨어 (미들웨어)

제주대학교

김 도 현

주요 학습 내용

- 임베디드 시스템의 소프트웨어 구조

- 임베디드 시스템의 소프트웨어 분류

- 임베디드 시스템의 미들웨어

- 임베디드 시스템의 미들웨어 종류

- 임베디드 시스템의 프로그래밍 언어

15 Billion
connected devices

5 Billion
connected devices

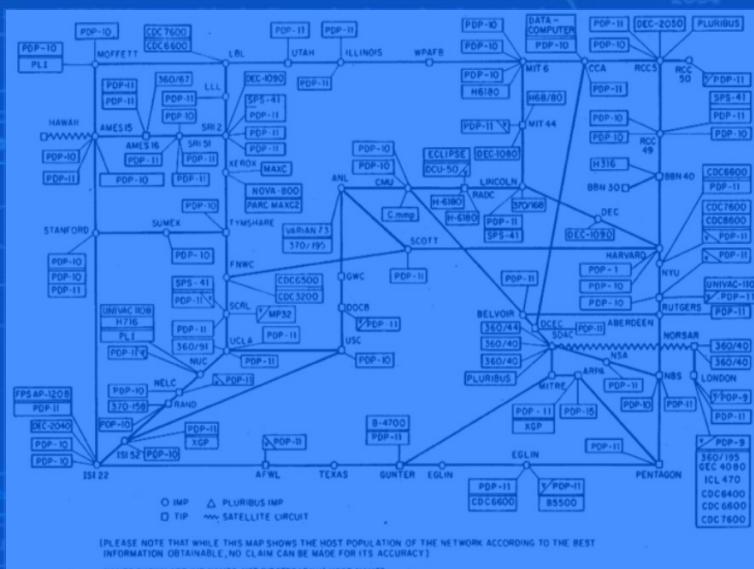
2 Billion
connected devices

93,047,785
connected devices

313,000
connected devices

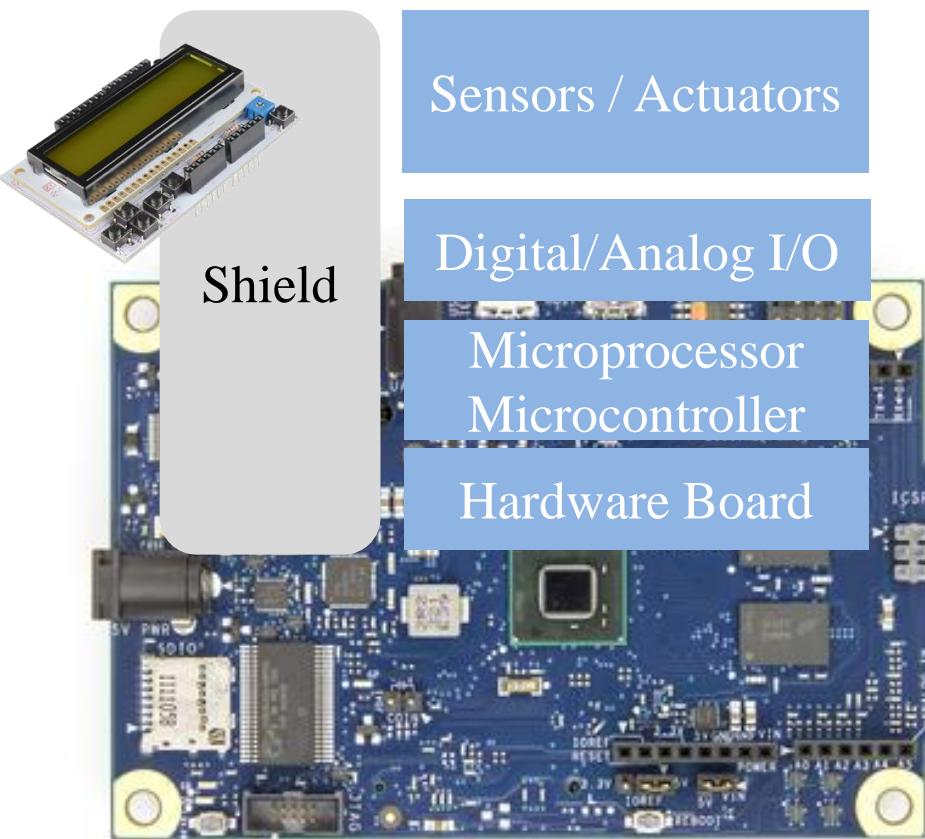
188
connected devices

13
connected devices



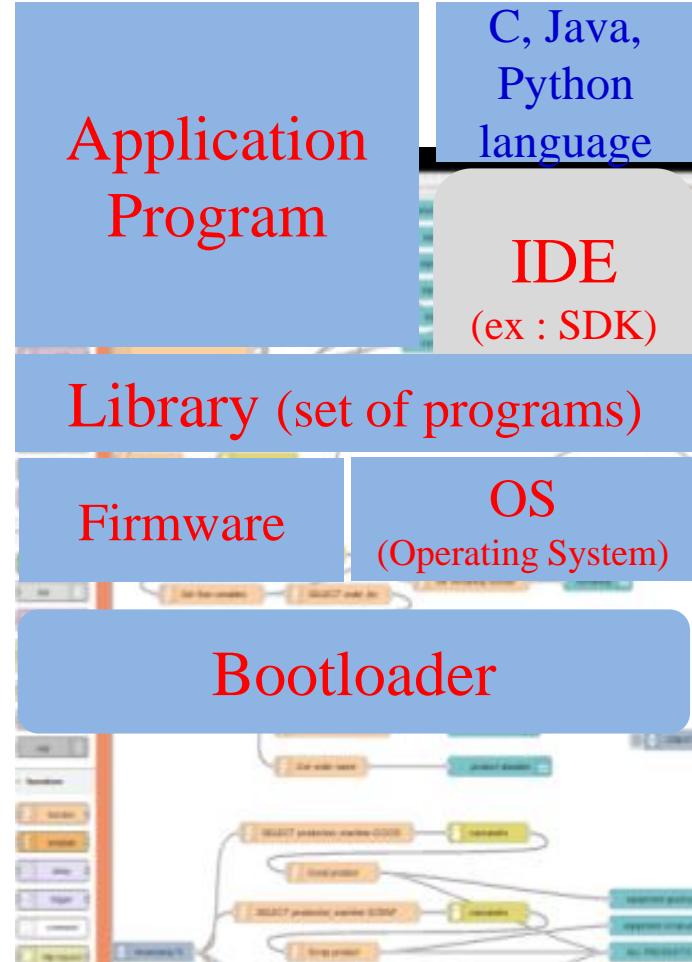
임베디드 시스템 구성

- 임베디드 소프트웨어 구성



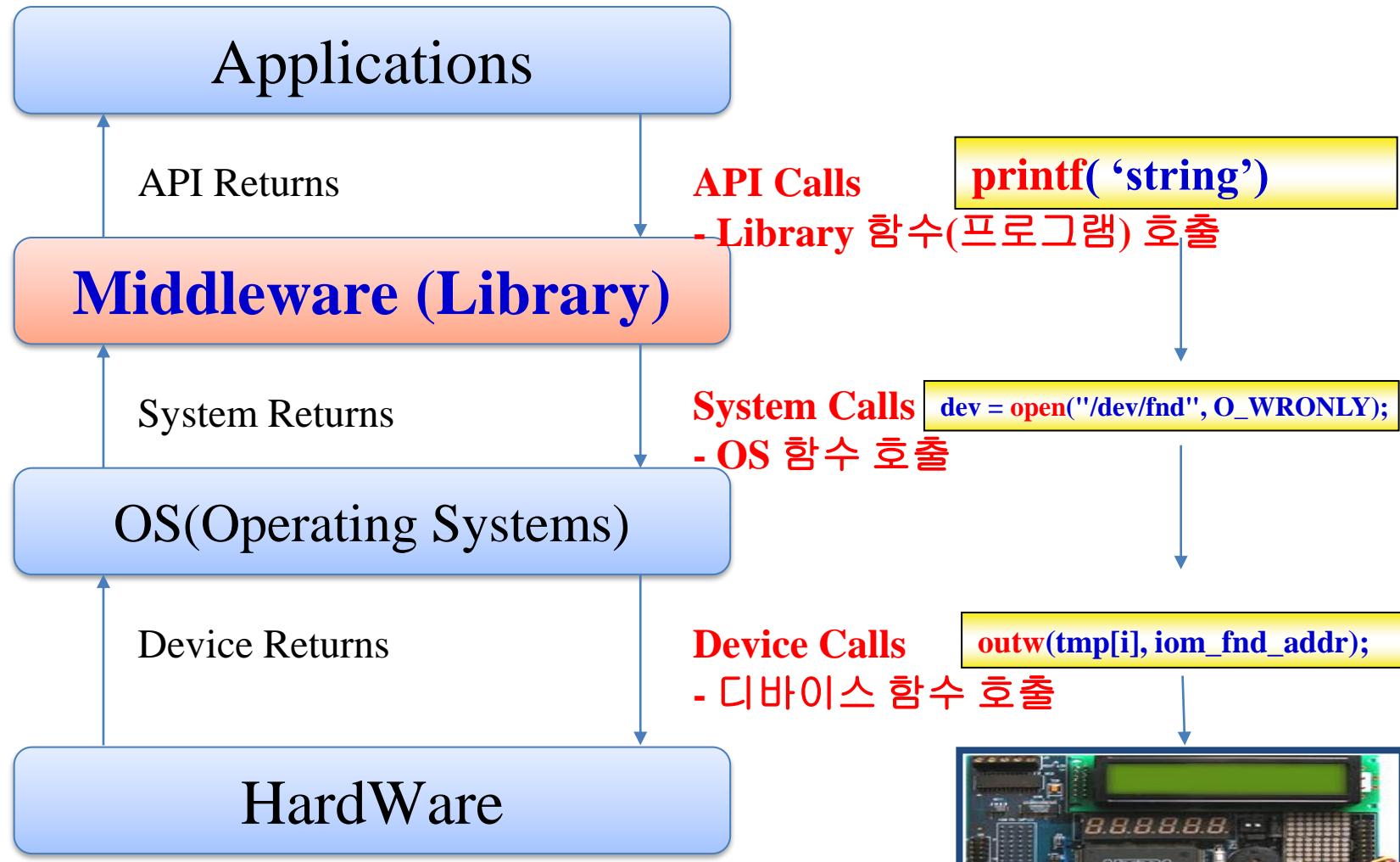
임베디드 하드웨어

(IDE : Integrated Development Environment, **SDK**)



임베디드 소프트웨어

임베디드 시스템의 소프트웨어 구조



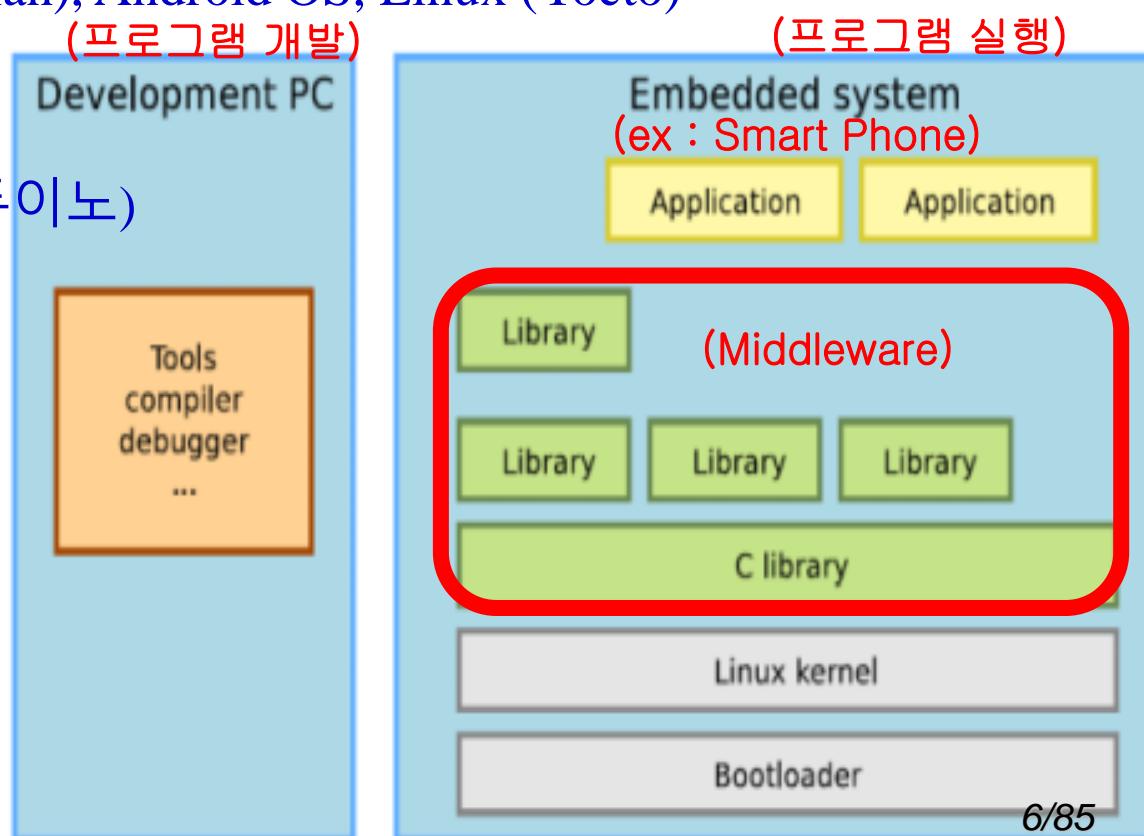
API (Application Programs Interfaces)

임베디드 시스템의 소프트웨어 구조

- 운영체계 OS
 - 하드웨어 동작(운영)시키는 프로그램
- 미들웨어 (플랫폼) Library
 - 여러 응용에서 사용하는 공통 기능을 프로그램들(입출력...라이브러리)
 - API? (Application Programs Interfaces(연결고리, 지점))
 - ex : printf(‘string’)
 - 방/복도 Interface : 문
 - 임베디드 응용 개발할 수 있도록 미들웨어(라이브러리)에서 제공하는 프로그램의 연결 지점
- 응용 (Application)
 - 표준 app (보조프로그램, ..)
 - 사용자 app (카톡, ..)

임베디드 시스템의 소프트웨어 구조

- 운영체계
 - Linux (**Debian, Ubuntu, Fedora**), BSD, Windows Embedded, Pidora
 - Archlinux, Raspian(Debian), Android OS, Linux (Yocto)
- 임베디드 프로그래밍 언어
 - (Assembly), C/C++(아두이노)
 - Python
 - Javascript/Node.js
 - Java (Java ME), 기타
- Bootloader
 - Uboot, GRUB, 등
- 각종 라이브러리



임베디드 시스템의 소프트웨어 구조

- 안드로이드 플랫폼

구분	주요 요소
임베디드 OS	<ul style="list-style-type: none">- 임베디드 OS 커널- 기본 디바이스 드라이버 및 라이브러리- GUI(Graphical User Interface) 도구
미들웨어	<ul style="list-style-type: none">- 멀티미디어 CODEC- Network Protocol Stack- Application Engine- 임베디드 VM(Virtual Machine)
애플리케이션 소프트웨어	<ul style="list-style-type: none">- PIMS(Personal Information Management System)- 웹 브라우저- 미디어 플레이어- 메시징 애플리케이션- 카메라 애플리케이션- DMB 플레이어- 특수 업무용 애플리케이션

(Library)

임베디드 시스템의 소프트웨어 구조

Embedd ((ex : Smart Phone Android))

PC

(ex : Smart Phone)

어플리케이션 소프트 웨어

홈(대기화면)

연락처

웹브라우저

Google Maps

...

App

App Framework

어플리케이션 프레임워크

Activity Manager

Window Manager

Content Providers

View System

Notification Manager

Package Manager

Telephony Manager

Resource Manager

Location Manager

XMPP Service

표준 라이브러리

M/W

Surface Manager

Media Framework

SQLite

안드로이드(Android) 런타임

OpenGL ES

FreeType

WebKit

코어 라이브러리

SGL

SSL

libc

Dalvik 가상 머신(VM)

커널(Linux 2.6.23)

OS

하드웨어

개발툴

Android Debug Bridge (adb)

Android Development Tools Plugin

Dalvik Debug Monitor Service (ddms)

Android Asset Packaging Tool

Android Interface Definition Language

mksdcard

Class file 컨버터

에뮬레이터
(OEMU,goldfish)

임베디드 시스템의 소프트웨어 구조

- 임베디드 응용 소프트웨어

구분	세부 기술
통합개발 솔루션 응용 (Application) 개발하는 프로그램(SDK)	<ul style="list-style-type: none">- 원격 디버거 기술- 시스템 성능 분석 및 실시간 모니터- 개발 자동화 도구
시험자동화 솔루션 SDK : TEST	<ul style="list-style-type: none">- 시험 데이터 생성 기술- 정적/동적 분석 기술- 모델 검증 및 자동 증명 기술
설계자동화 솔루션 SDK : Design	<ul style="list-style-type: none">- 개발 프로세스 기술- 시스템 분석 및 모델링 기술- 아키텍처 기반 개발 기술

임베디드 시스템의 소프트웨어 분류

- 임베디드 시스템의 소프트웨어 분류

기술구분	활용사례
임베디드 기본 운용 App	멀티미디어 재생기, MAP Viewer , 브라우저, PIMS , 게임, 모바일 속 CNS(GPS,GIS)
임베디드 미들웨어 Middleware	COBRA, COM, XML, TMO 등의 분산 미들웨어, JVM, J2ME 등 자바 미들웨어, Jini, UPnP, Havi 등 제어 미들웨어, 스트리밍 및 Codec 등 멀티미디어 미들웨어, WRAN, WPAN 관련 통신 미들웨어
임베디드 시스템 소프트웨어 OS	임베디드 OS , 디바이스 드라이버, 유무선 통신 프로토콜 및 멀티미디어 프로토콜 지원, 라이브러리, 임베디드 DBMS
임베디드 시스템 개발 도구	설계 도구, 시험 검증 도구, IDE , 타깃 시스템 재설정 도구, 각종 시뮬레이터, 크로스 컴파일러, 실시간 원격모니터, 원격 모니터, 디바이스 드라이버 툴킷, 설계 도구, 시험 검증 도구 응용 (Application) 개발하는 프로그램(S)
임베디드 소프트웨어 플랫폼	MS WinCE 닷넷, Sun J2ME , Brew , WIPI

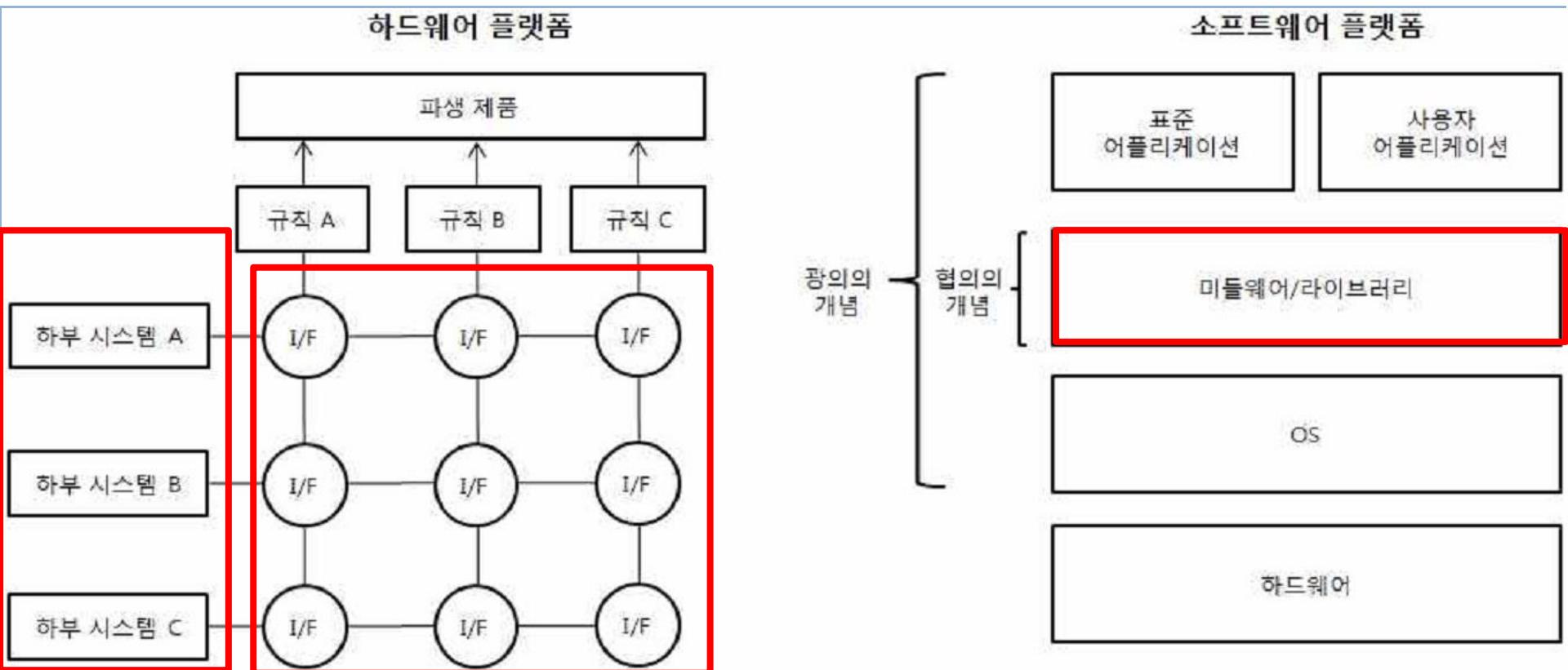
임베디드 시스템의 소프트웨어 분류

- 오픈소스 임베디드 시스템의 소프트웨어 비교

	Arduino	Raspberry Pi	Beaglebone Black	Intel Galileo
운영체제 OS	Toolkit Firmware	Pidora, Archlinux, Raspbian(Debian); Android OS, Firefox OS, ...	Linux (Debian, Ubuntu, Fedora), BSD, Windows Embedded, 기타	Linux (Yocto) (with Grub)
개발환경 SDK	독자 IDE	Eclipse 등	Eclipse 등	Arduino IDE
프로그래밍 언어	Arduino C	Python 중심	Node.js 중심	Arduino C 또는 Linux 개발환경
라이브러리 Middleware	Arduino Library	Linux 표준 library 등	Linux 표준 library 등	Linux 표준 library 등
기타	H/W 중심	SBC	SBC	Arduino with Intel Quark Inside! (400MHz)

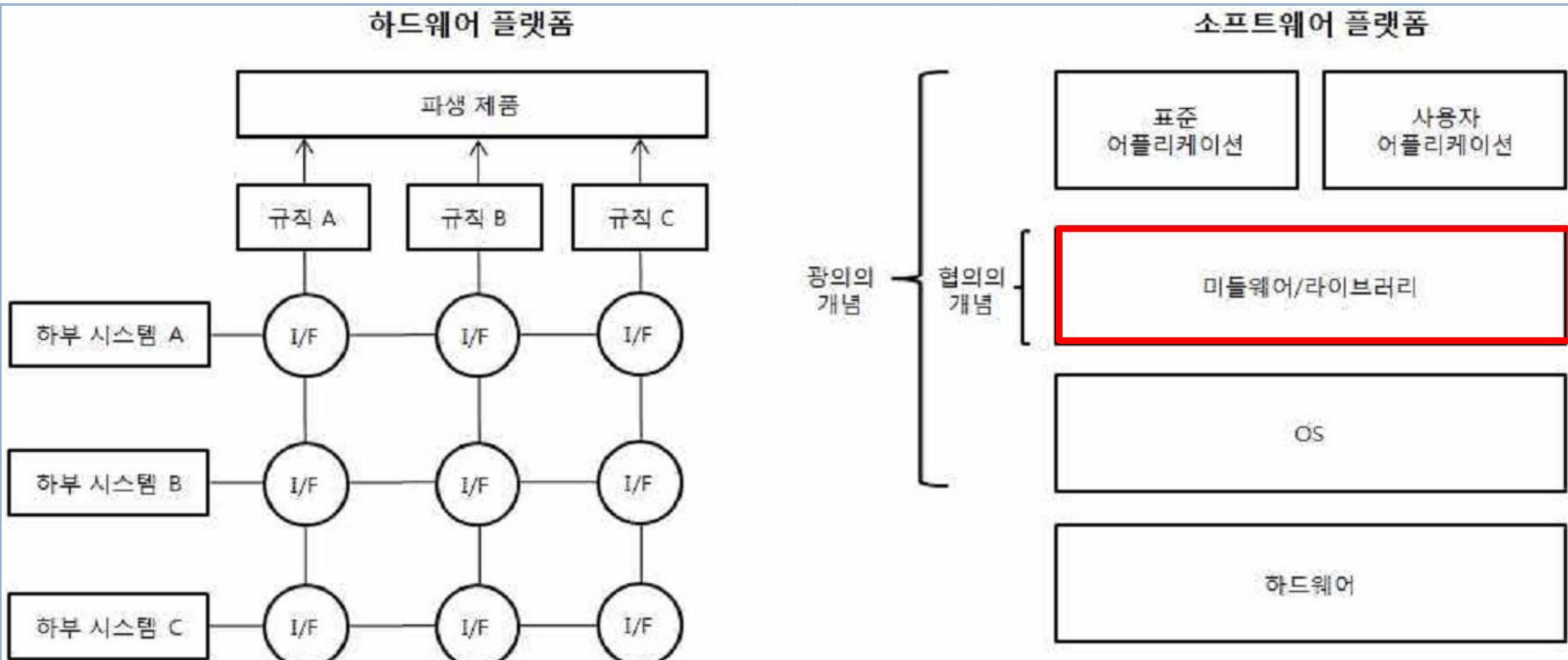
임베디드 시스템의 미들웨어

- 제품 자체뿐만 아니라 제품을 구성하는 부품과 다른 서비스와의 연계를 도와주는 기반 서비스나 소프트웨어
- API (Application Programs Interfaces) 제공
 - 개발의 유연성을 확대하고 소프트웨어의 다양성 제공



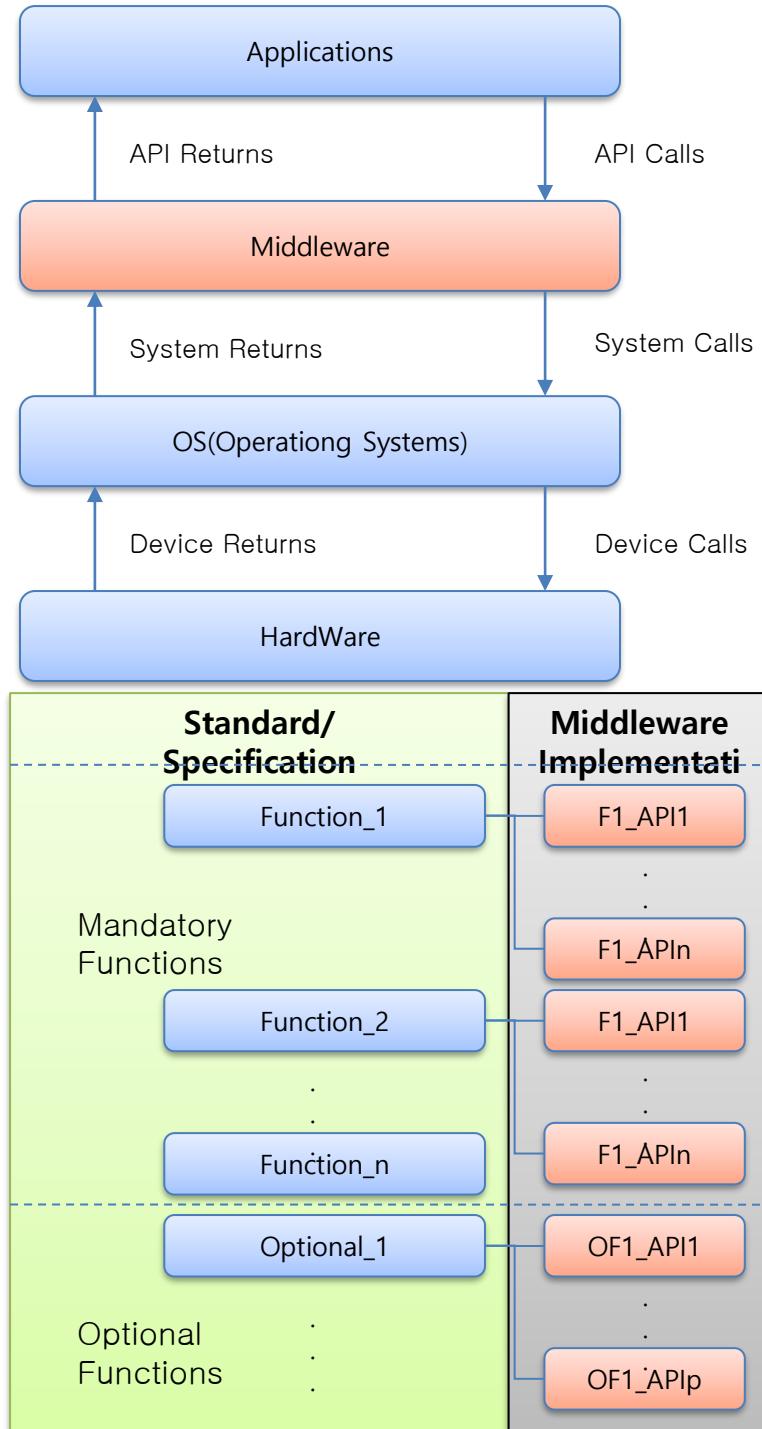
임베디드 시스템의 미들웨어

- 미들웨어 (플랫폼)
 - 여러 응용에서 사용하는 공통 기능을 프로그램들(입출력...라이브러리)
 - API? (Application Programs Interfaces(연결고리, 지점) ex : printf('string'))
 - 방/복도 Interface : 문
 - 응용 개발할 수 있도록 미들웨어(라이브러리)에서 제공하는 프로그램의 연결



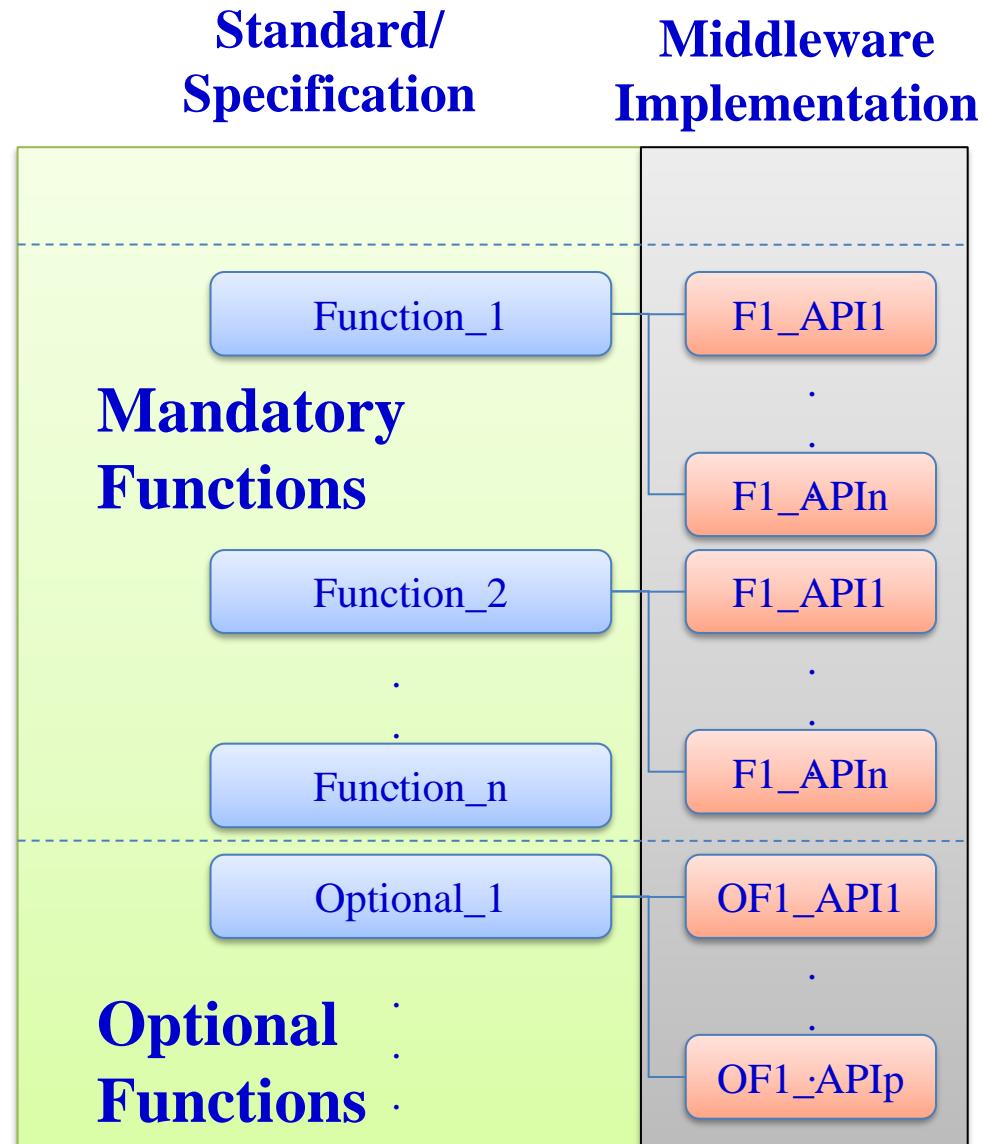
임베디드 시스템의 미들웨어

- 임베디드 미들웨어 정의
 - 응용들의 **공통으로 필요한 기능을** 제공해 응용 개발을 쉽게 할 수 있도록 도와주는 라이브러리 역할
 - 시스템 S/W와 응용 S/W의 중간에 위치하여 특정 응용 분야에 최적화 된 공통 프레임워크



임베디드 시스템의 미들웨어

- 임베디드 미들웨어 기능
 - 여러 개의 API를 호출하는 함
수로 해당 기능을 구현
 - 필수적인 기능과 부가적인 기
능으로 구성
 - 해당 지원하는 미들웨어 API
기능 안에서만 구현 가능



임베디드 시스템의 미들웨어

Level	Check Points	Tools
System	<ul style="list-style-type: none">■ Integration Reliability, Robustness■ Exception Handling in the system level	Chamber, Automated Tools (LabView)
Control Logic	<ul style="list-style-type: none">■ Customer Specification & Requirement Functionality	Stub(Speed 2k, CAN)
Transport	<ul style="list-style-type: none">■ Protocols IEEE Standard, De-facto Standard, Specific Protocol■ Performance Load & Stress, Scalability, Optimization, Benchmark■ Compatibility Test Inter-operability, Compatibility, Reliability■ Resource Allocation	Dedicated Tools for each Protocols (CanOE)
O/S	<ul style="list-style-type: none">■ Device Driver Reliability Test Functionality, Compatibility■ Resources Allocation	Tool Chains, Cross compiler, Oscilloscope
Hardware	<ul style="list-style-type: none">■ PCB/SOC Level Testbed■ Timing & Clock debugging■ Hardware Compatibility■ Reliability & Performance	Oscilloscope, Digital Analyzer, JTAG, in- circuit emulator

Application

Middleware

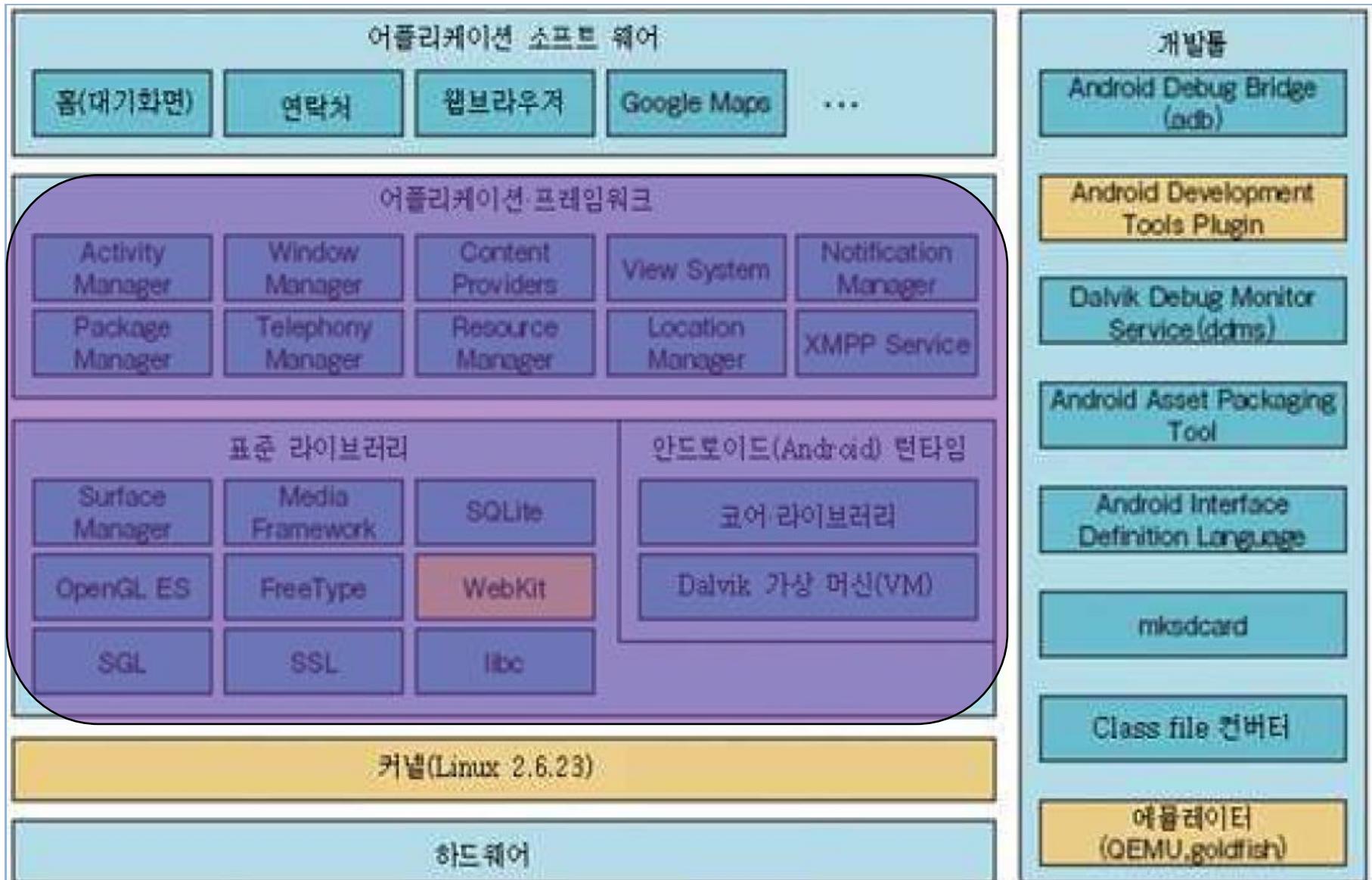
OS

임베디드 시스템의 미들웨어 종류

구분	주요 요소
임베디드 OS	<ul style="list-style-type: none">- 임베디드 OS 커널- 기본 디바이스 드라이버 및 라이브러리- GUI(Graphical User Interface) 도구
미들웨어	<ul style="list-style-type: none">- 멀티미디어 CODEC- Network Protocol Stack- Application Engine- 임베디드 VM(Virtual Machine)
애플리케이션 소프트웨어	<ul style="list-style-type: none">- PIMS(Personal Information Management System)- 웹 브라우저- 미디어 플레이어- 메시징 애플리케이션- 카메라 애플리케이션- DMB 플레이어- 특수 업무용 애플리케이션

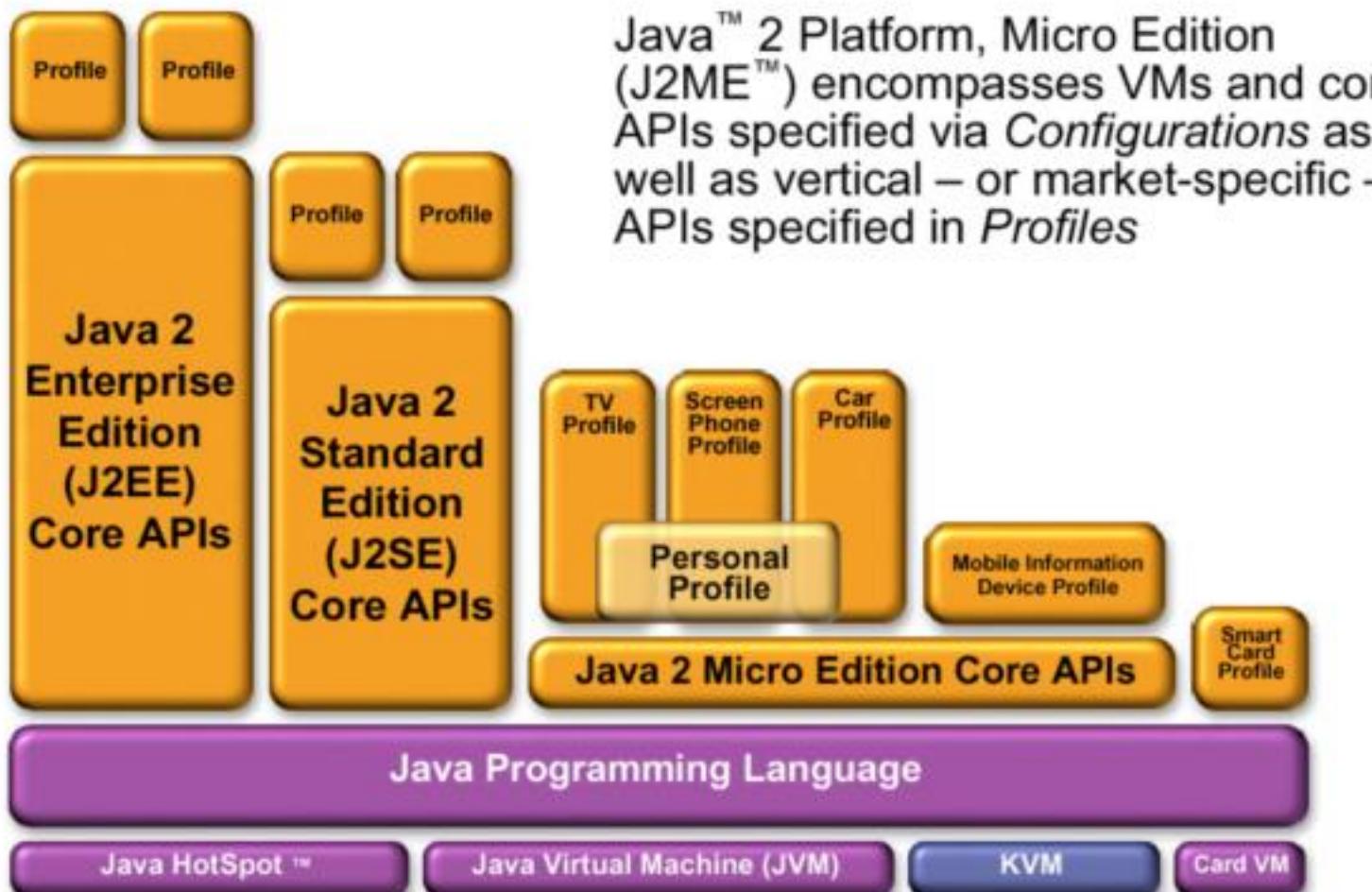
임베디드 소프트웨어				
기본응용	미들웨어	임베디드 S/W 플래폼	임베디드 S/W 개발도구	임베디드 S/W 프레임워크
<ul style="list-style-type: none">멀티미디어 재생기MAP ViewerPIMS모바일샵CNS(GPS,GIS)게임	<ul style="list-style-type: none">분산 미들웨어자바 미들웨어제어 미들웨어멀티미디어 미들웨어WLAN,WPAN 관련 통신 미들웨어	<ul style="list-style-type: none">임베디드 OSDevice Driver유무선 통신 및 Multimedia Protocol 지원 LIB임베디드 DBMS	<ul style="list-style-type: none">설계도구시험 검증 도구재설정 도구임베디드 시스템 시스템각종 시뮬레이터실시간 원력모니터 툴킷IDE	<ul style="list-style-type: none">MS닷넷 컴팩트 프레임워크SUN ONEWIPIBrew

임베디드 소프트웨어 플랫폼 or 미들웨어 (Android)



임베디드 시스템의 프로그래밍 언어

- Java



학습 내용 정리

- 임베디드 시스템의 소프트웨어 구조에 대해 알아본다.
- 임베디드 시스템의 소프트웨어 분류에 대해 알아본다.
- 임베디드 시스템의 미들웨어에 대해 살펴본다.
- 임베디드 시스템의 미들웨어 종류에 대해 알아본다.
- 임베디드 시스템의 프로그래밍 언어에 대해 살펴본다.



(PLEASE NOTE THAT WHILE THIS MAP SHOWS THE HOST POPULATION OF THE NETWORK ACCORDING TO THE BEST INFORMATION OBTAINABLE, NO CLAIM CAN BE MADE FOR ITS ACCURACY)
NAMES SHOWN ARE IMP NAMES, NOT NECESSARILY HOST NAMES



Thank you for listening

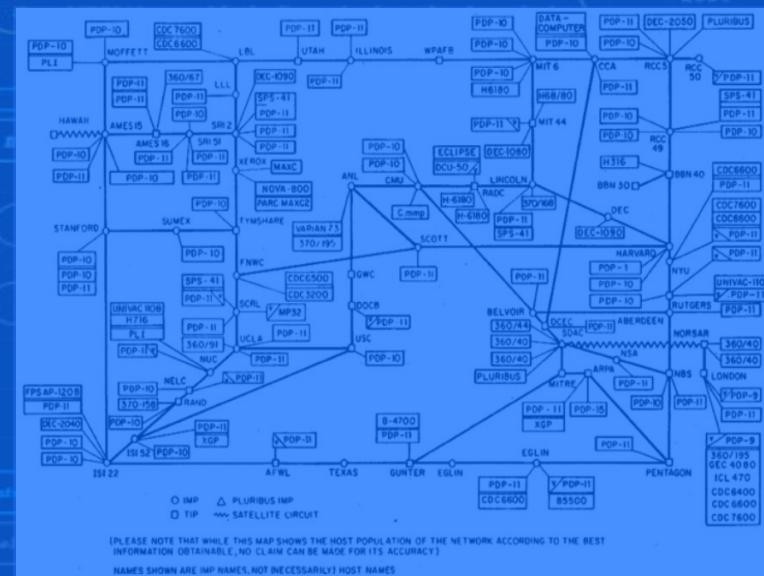
임베디드 시스템의 소프트웨어 (운영체계)

제주대학교

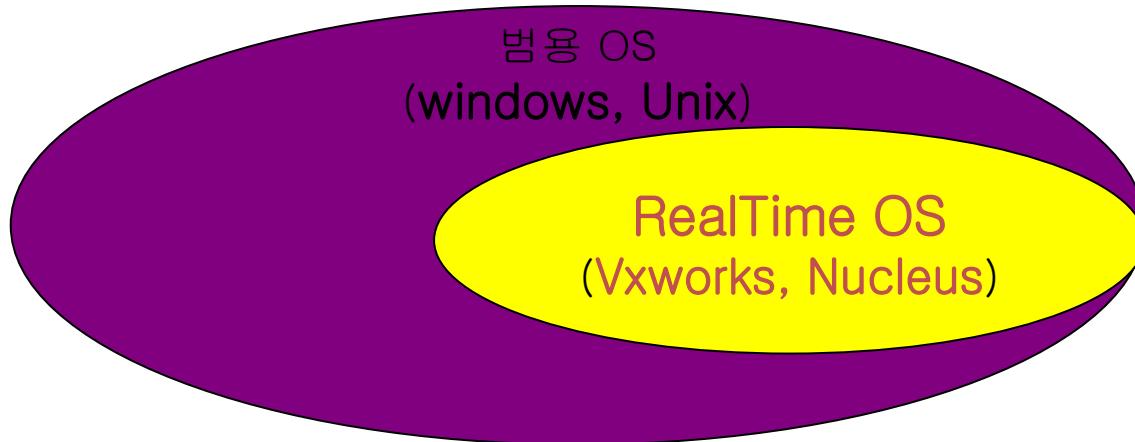
김 도 현

주요 학습 내용

- 임베디드 시스템의 운영체제
- 임베디드 시스템의 운영체제 구성
- 임베디드 시스템의 운영체제 주요 기능
- 임베디드 시스템의 운영체제 종류
- 임베디드 리눅스 개요
- 임베디드 리눅스 특징
- 임베디드 리눅스 구조
- 임베디드 리눅스 응용

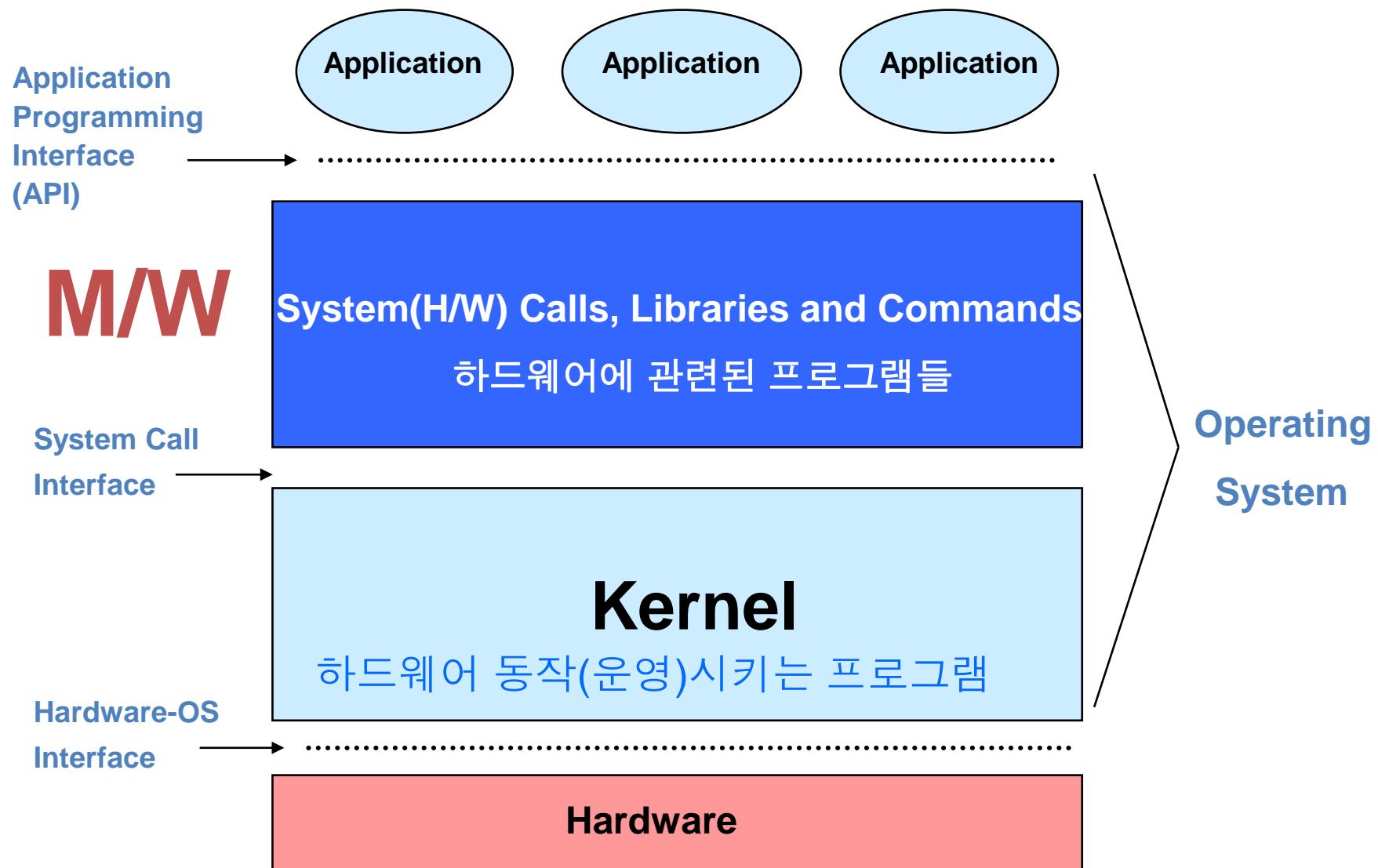


임베디드 시스템의 운영체제



- 임베디드 하드웨어 동작(운영)시키는 프로그램 또는 S/W(Program set)
- 임베디드 하드웨어를 관리하는 프로그램
- 임베디드 하드웨어를 관리할 뿐 아니라 응용 소프트웨어를 실행하기 위하여 하드웨어 플랫폼과 공통 시스템 서비스를 제공하는 소프트웨어

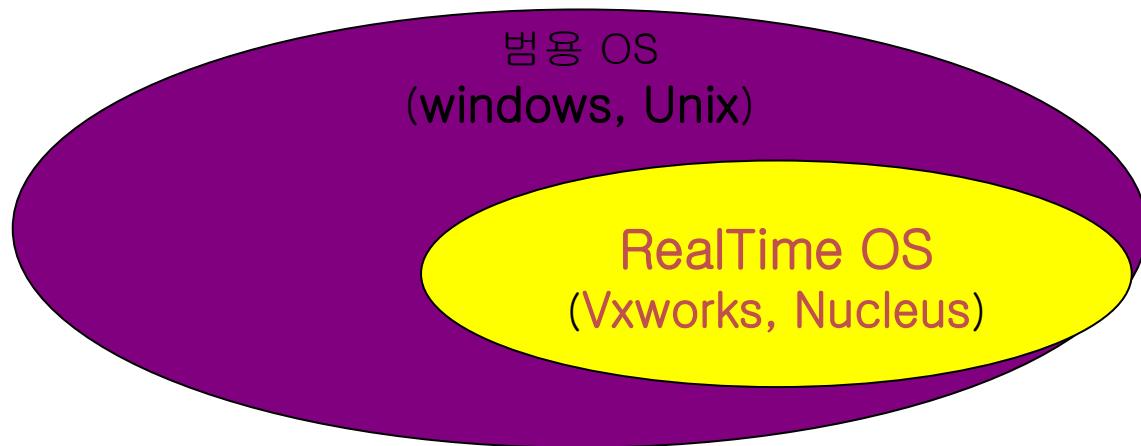
임베디드 시스템의 운영체제 구성



임베디드 시스템의 운영체제 주요 기능

- **멀티태스킹**
 - 여러 개의 응용 프로그램이 동시에 실행하는 기능
- **스케줄링**
 - 응용프로그램들 간에 실행 우선 순서를 정의하고, 각 응용프로그램의 얼마 동안의 시간을 배정하여 운영하는 것을 결정하는 기능
- **메모리 관리** (CPU에서 응용 프로그램 동작하기 위해서는 하드디스크에서 메모리(RAM)으로 어떻게 옮기고, 어디에 옮길 것이냐??)
 - 여러 개의 응용프로그램 간에 메모리 공유와 배정하는 관리 기능
- **입출력 관리**
 - 하드디스크, 프린터, 디지털 포트 등, 장착되어 있는 하드웨어 주변장치를 관리하고, 입출력 데이터를 처리하는 기능
- **메시지 관리**
 - 응용프로그램이나 사용자 또는 오퍼레이터 등에게 운영상황이나 에러 등에 관한 메시지 전달 기능

임베디드 시스템의 운영체제 종류

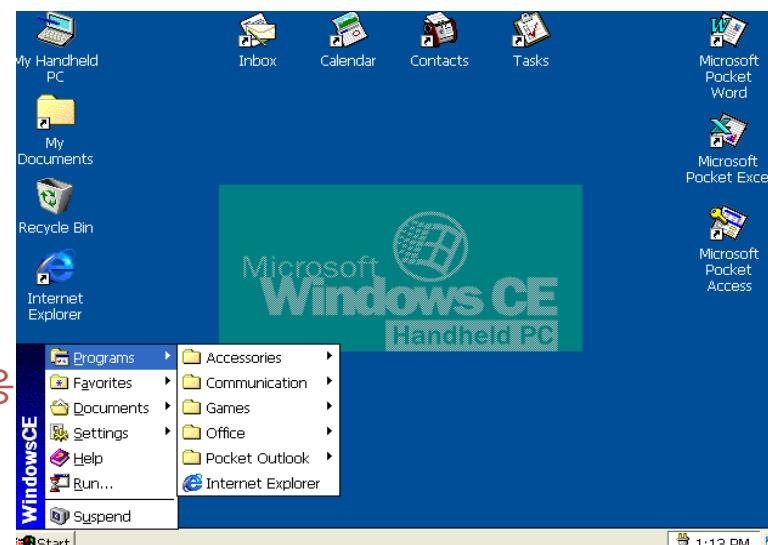


- 범용 OS (ex : MS Windows)
 - 모든 태스크(Task : 처리할 일거리)를 공평하게 처리 (먼저 Task부터 처리)
- 실시간 (real-time) OS
 - 어떤 원하는 시간내에 처리하는 운영체계
 - 우선권이 높은 태스크부터 처리
 - 시간 제약성, 신뢰성 등을 일반 운영체제 보다 중요시 함
 - 일반적으로 한가지 목적에 최적화 되어있음

임베디드 시스템의 운영체제 종류

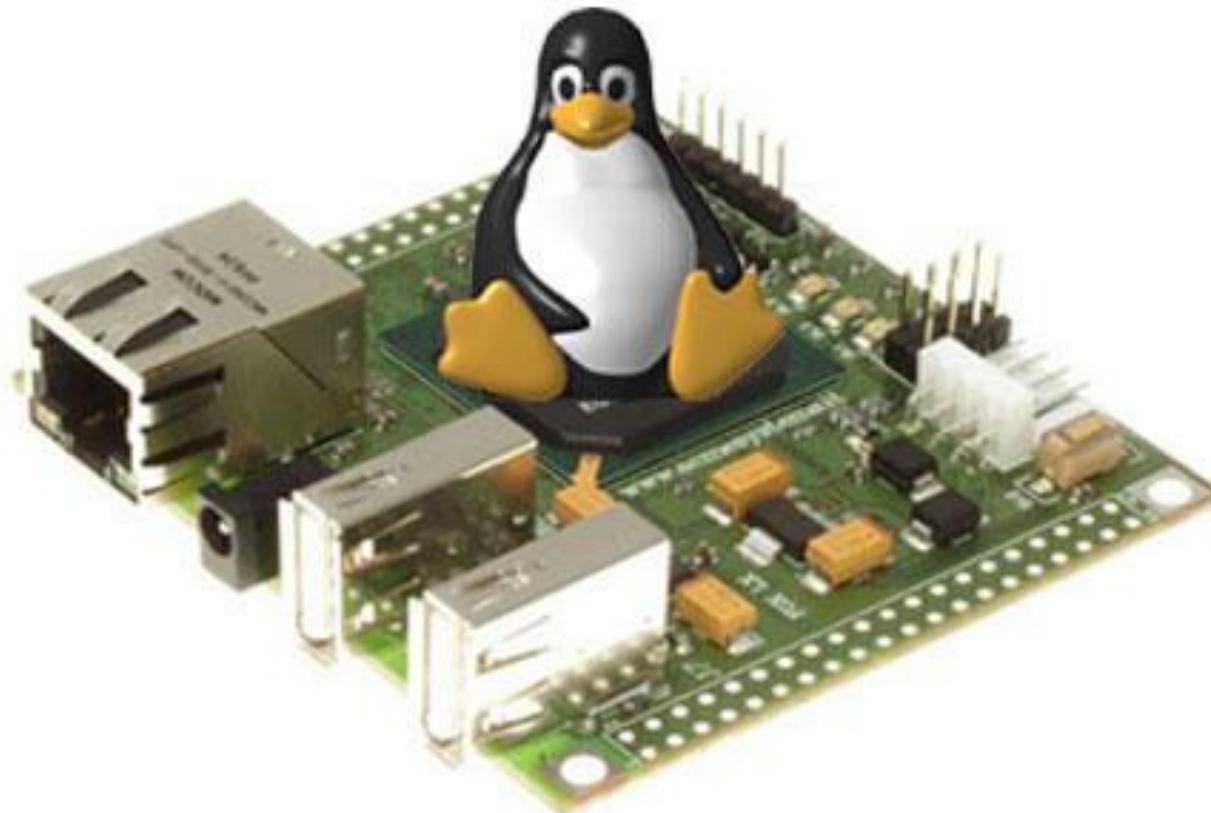
- **MS Windows CE (범용)**

- MS사에서 임베디드 시스템을 위하여 제공하는 운영체제
(32bit윈도우 운영체제와 호환성)
 - 기존의 데스크 탑 PC와 동일한 윈도우 환경 제공
 - MS사에서 제공되는 라이브러리에 종속적, 자유로운 개발 어려움
 - 데스크 탑 윈도우 및 응용 프로그램과의 **호환성 우수**
 - 프로그램 개발 환경이 아주 우수. (특히, GUI 개발 환경 우수)
 - 실행 환경에서 요구되는 H/W 사양이 높고, 가격이 비쌈
 - 개발툴 : 윈도우 CE용 VC++6.0, 윈도우 CE용 VB6.0
 - TCP/IP, PPP, IrDA 프로토콜 : 다른 시스템에 쉽게 연결
 - Intel 의 x86계열을 포함한 다수의 32bit 프로세서 지원
 - 기존의 많은 개발자들이 응용프로그램 개발에 쉽게 적응



임베디드 시스템의 운영체제 종류

- **임베디드 리눅스**
 - 낮은 성능의 프로세서(CPU)와 작은 크기의 메모리(RAM)를 가진 내장형 시스템용으로 개발된 리눅스



임베디드 시스템의 운영체제 종류

- pSOSytem
 - 휴대폰, 각종 통신 장비와 네트워크 장비등에서 사용
 - 커널을 중심으로 여러 개의 독립적인 모듈인 소프트웨어 컴포넌트들로 구성
 - 선점형(**우선순위**) 태스크 스케줄링(같은 **우선순위일 경우 Time slicing**에 의한 라운드로빈 방식)
 - 처리 가능한 **최대 태스크 수** : 256
 - 장점 : i386, M68K MIPS, SPARC, ST20등 다양한 CPU지원
- VxWorks
 - pSOSytem과 유사
 - 선점형 멀티미디어 태스킹
 - 256개의 태스크 우선순위 (같을 경우 라운드로빈)
 - 태스크간의 통신을 위해 세마포어, 메세지큐, 공유메모리, 소켓, 시그널 등 제공
 - 표준 TCP/IP를 이용한 네트워크 통신이 가능하며 ROM이나 로컬 디스크 등 다양한 부팅가능
 - MS-DOS와 TR-11등의 파일 시스템 지원

임베디드 시스템의 운영체제 종류

- **VRTX**

- 국내에서 가장 많이 쓰였던 실시간 운영체제로 Mentor Graphics사에서 개발
- 높은 신뢰성
- 통신장비, 네트워크장비, 휴대폰, 경주용자동차의 제어시스템 발전소의 모니터링 시스템 전동차의 제어시스템 및 모니터링 시스템 등 다방면의 분야에 사용
- 선점형 멀티 태스킹 커널
- 모듈형태로 구성 개발자들이 모듈을 선택적으로 사용하여 운영체제 구성가능
- MS-dos6.1포함, ANSI-C라이브러리 제공
- ARM 프로세서를 지원하며 등록만 하면 소스코드 공개

임베디드 시스템의 운영체제 종류

- **Nucleus**

- 실시간 운영체제로서 저작권 없는 정책 때문에 널리 사용되어짐
- ARM, MIPS, PPC, M68K, SH 등 마이크로프로세서를 비롯하여 Analog와 TI, DSP등 지원
- VRTX 제조원인 멘토 그래픽스와 합병되어진 상태
- Web Brower는 HTTP1.0준수, HTML3.2, 프레임지원

임베디드 리눅스 개요

- **임베디드 리눅스**
 - 임베디드 시스템을 위해 경량화된 리눅스
 - 임베디드 시스템에 포팅(Porting)된 리눅스
- **포팅(Porting) : APP와 운영체계를 설치**
 - 어떤 프로그램을 특정 플랫폼(embedded system)에서 동작될 수 있도록 다른 시스템(PC)에서 프로그래밍하여 설치하는 기술
 - 예
 - 윈도우 기반의 MSN Messenger → Linux 기반의 MSN Messeger
 - Intel processor PC 기반의 Linux → ARM processor 보드 기반의 Linux

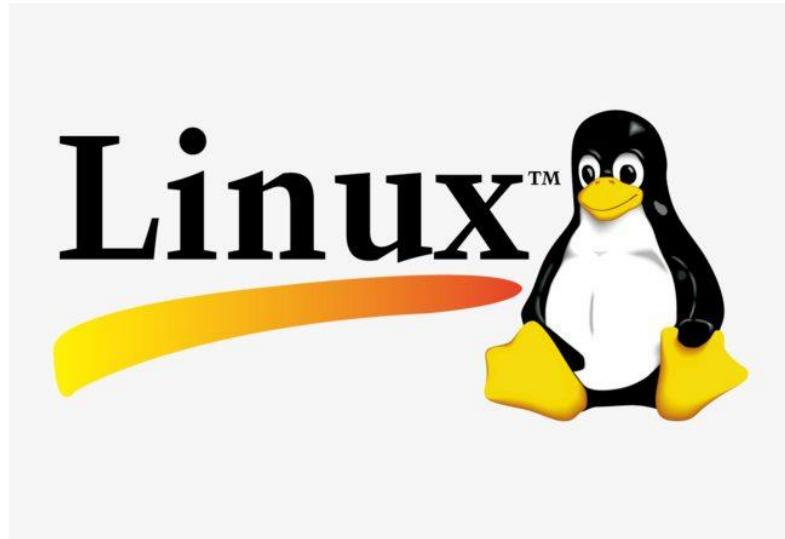
임베디드 리눅스 개요

	리눅스	임베디드 리눅스
주 사용 용도	데스크탑 PC 혹은 서버	임베디드 시스템 장치/장비
사용 CPU	인텔, AMD 데스크탑/서버 CPU	ARM, AVR32, MIPS, PowerPC, m68k 등
설치 방법	리눅스 배포판 CD/USB/Network로 손쉬운 설치	Bootloader, 리눅스 Kernel, FileSystem 등을 별도로 Target Board에 전송하여 사이즈를 줄여서 설치. 다소 절차가 복잡하고 난이도가 있다.
주 사용자	일반 사용자 혹은 서버 관리자, 서버 개발자	임베디드 시스템 개발자
시스템 자원 소모	충분한 HDD, RAM 등의 시스템 자원 사용. 시스템 자원 소모에 대한 부담이 덜하다.	한정된 RAM, Flash 등의 자원으로 인해 최소화/최적화가 필요함.
대표 패키지	레드햇, 데비안, 우분투, 민트 리눅스	안드로이드 SDK, BuildRoot, OpenWRT
S/W 업그레이드 용이성	자체 OS 지원의 원활함으로 인해 용이함.	업그레이드는 개발자가 해줘야 함으로써 제조사가 해주지 않으면 즉각 업그레이드는 어렵다.

임베디드 리눅스 특징

- **임베디드 리눅스 장점**

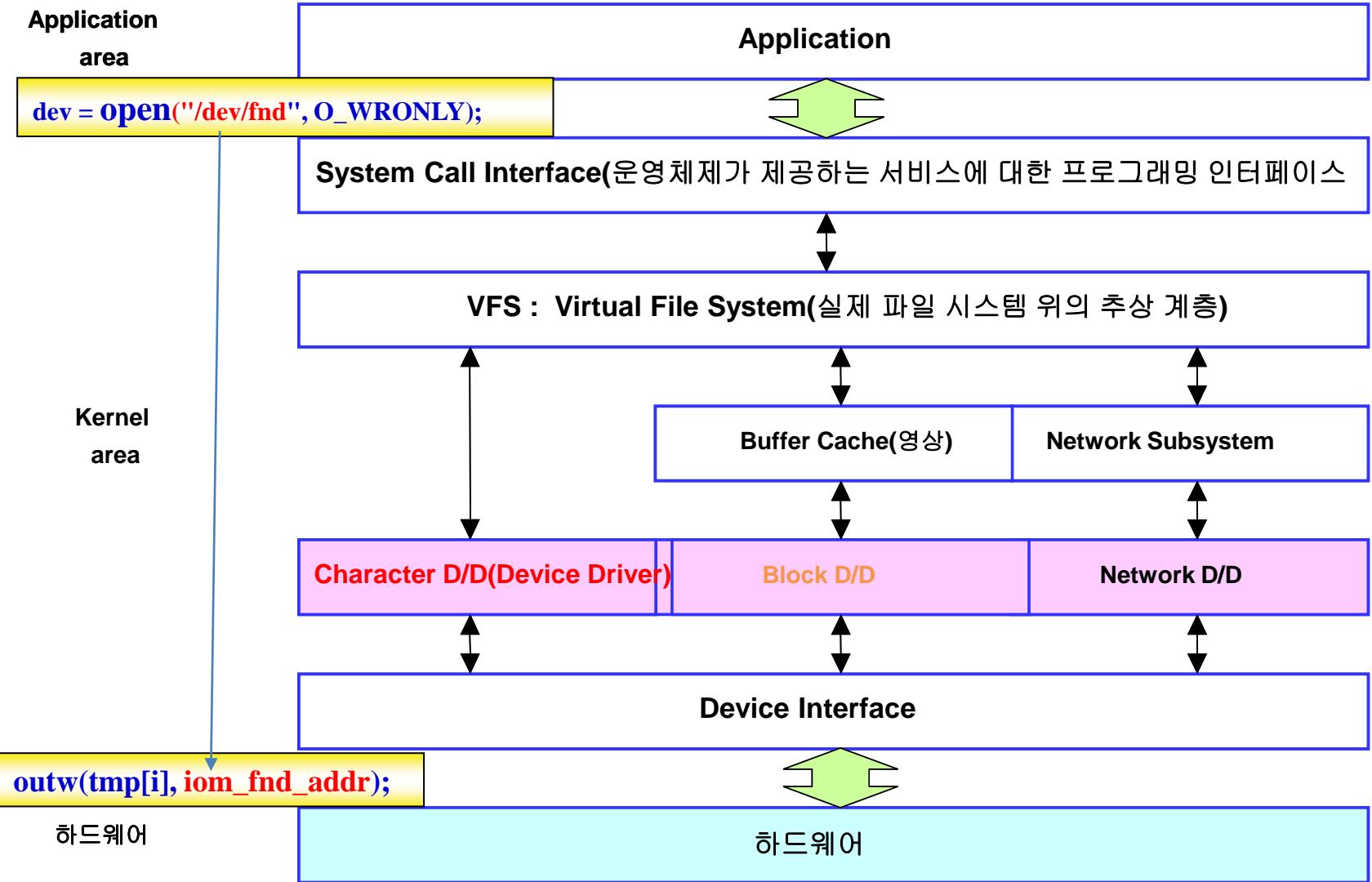
- Open Source (무료)
 - 학습 용이
- **안정성과 신뢰성 우수**
- 다양한 CPU Platform 지원
- 개발 비용이 비교적 저렴
- 기능성과 확장성이 우수 (리눅스 이용에 따른 장점)
- PowerPC, ARM, MIPS 등 **다양한 CPU Platform** 지원함
- 로열티가 없으므로 가격 경쟁력이 우수
- 사용자 층이 넓어 오류 수정이 빠르고 안정성이 우수
- 기존의 데스크 탑 개발 환경과 동일하여 개발이 용이함



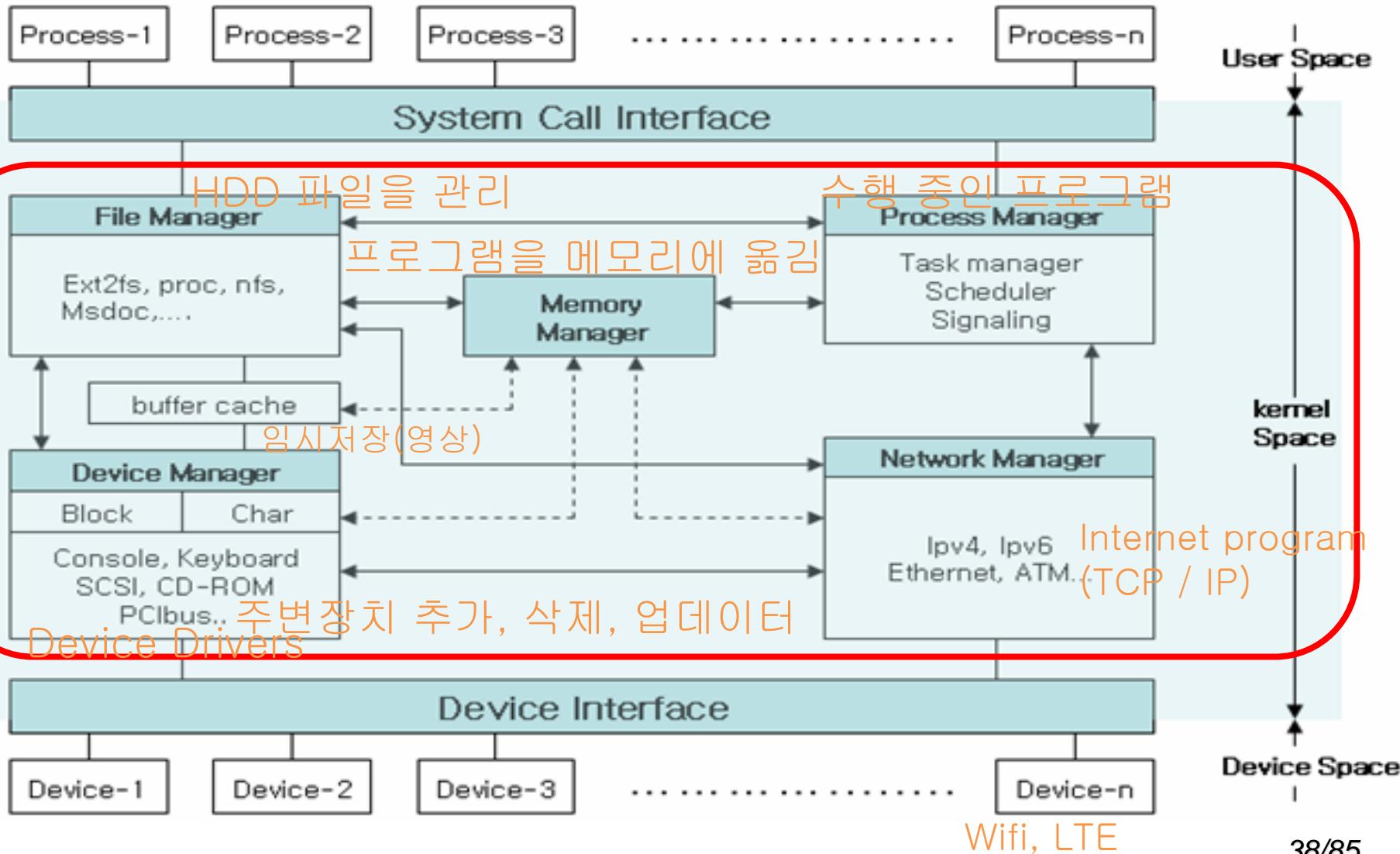
임베디드 리눅스 특징

- 단점
 - 커널 크기 (중간)
 - 상용 RTOS < Linux < 범용OS
 - 완전하지 못한 Real Time
 - 제한적 preemption (우선순위 일부 지원)
 - 개발 환경의 부족
 - 일반적으로 Text 기반
 - 불투명한 안정성
 - 기존의 RTOS 보다 많은 메모리를 요구함
 - 범용 OS로 설계되어 Real-Time을 지원하지 못함
 - 개발 환경이 Text 기반의 환경임으로 개발에 어려움이 있음
 - GUI 환경을 개발하기 어려움
 - 제품화하기 위한 솔루션 구성이 어려움
 - 많은 업체들과 개발자들이 독자적으로 개발하고 있어 표준화가 어려움

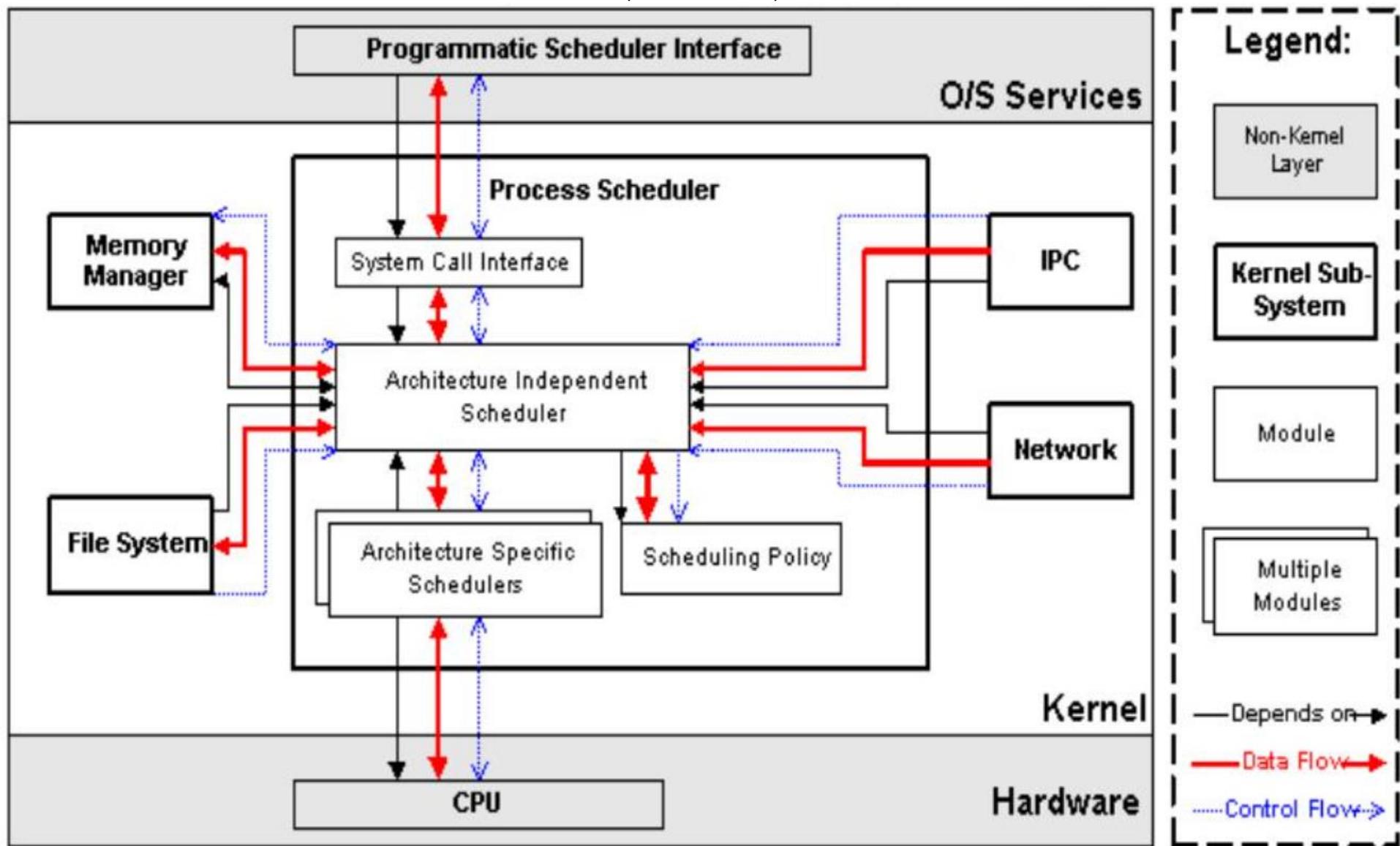
임베디드 리눅스 구조(Architecture)



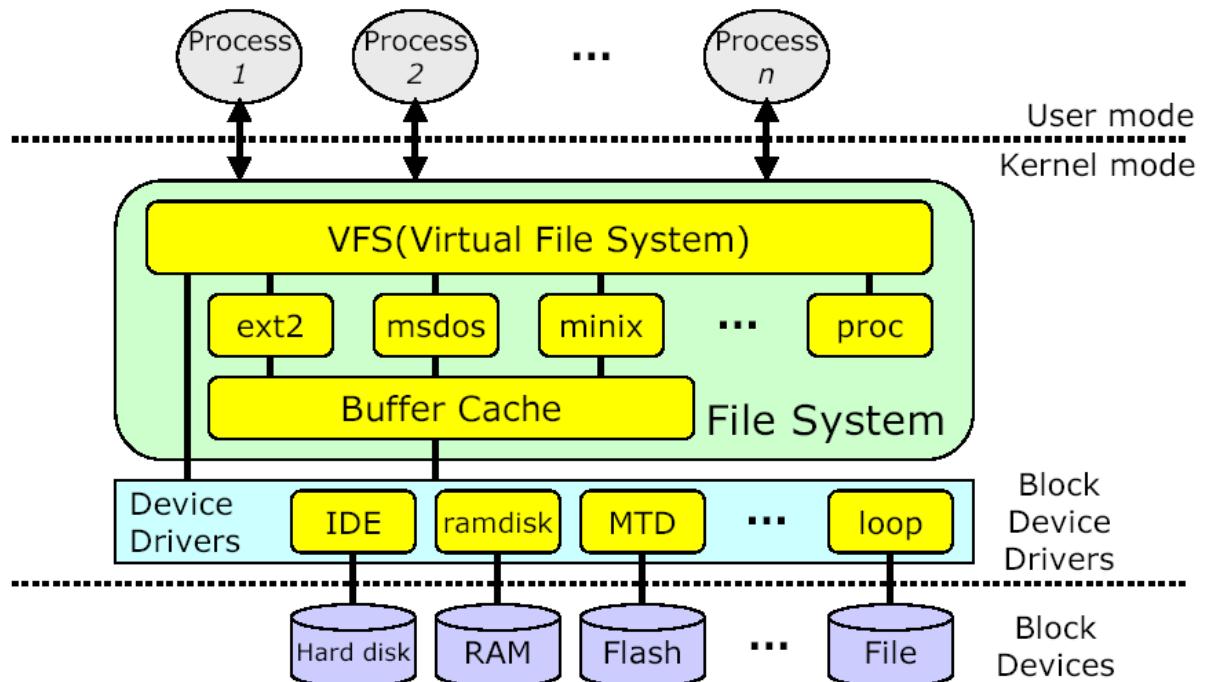
임베디드 리눅스 구조 (커널)



임베디드 리눅스 구조 (커널)

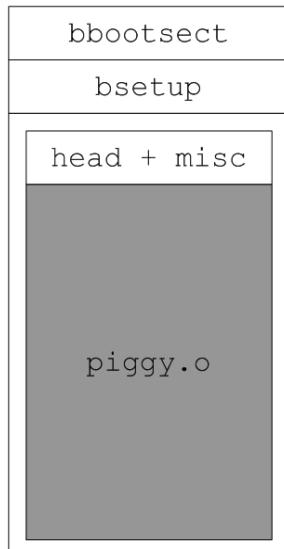


임베디드 리눅스 구조 (파일시스템과 램디스크)



- 파일시스템
 - 보조 기억장치와 그 안에 저장되는 파일을 관리하는 알고리즘 및 자료구조의 통칭
- 램디스크 (**Ramdisk**)
 - 임베디드 시스템과 같이 별도의 보조 장치를 둘 수 없을 때 사용
 - 하드디스크와 같이 별도의 물리적 저장 장치 없이 메모리의 한 부분을 파일 시스템으로 사용하는 파일시스템

임베디드 리눅스의 구성

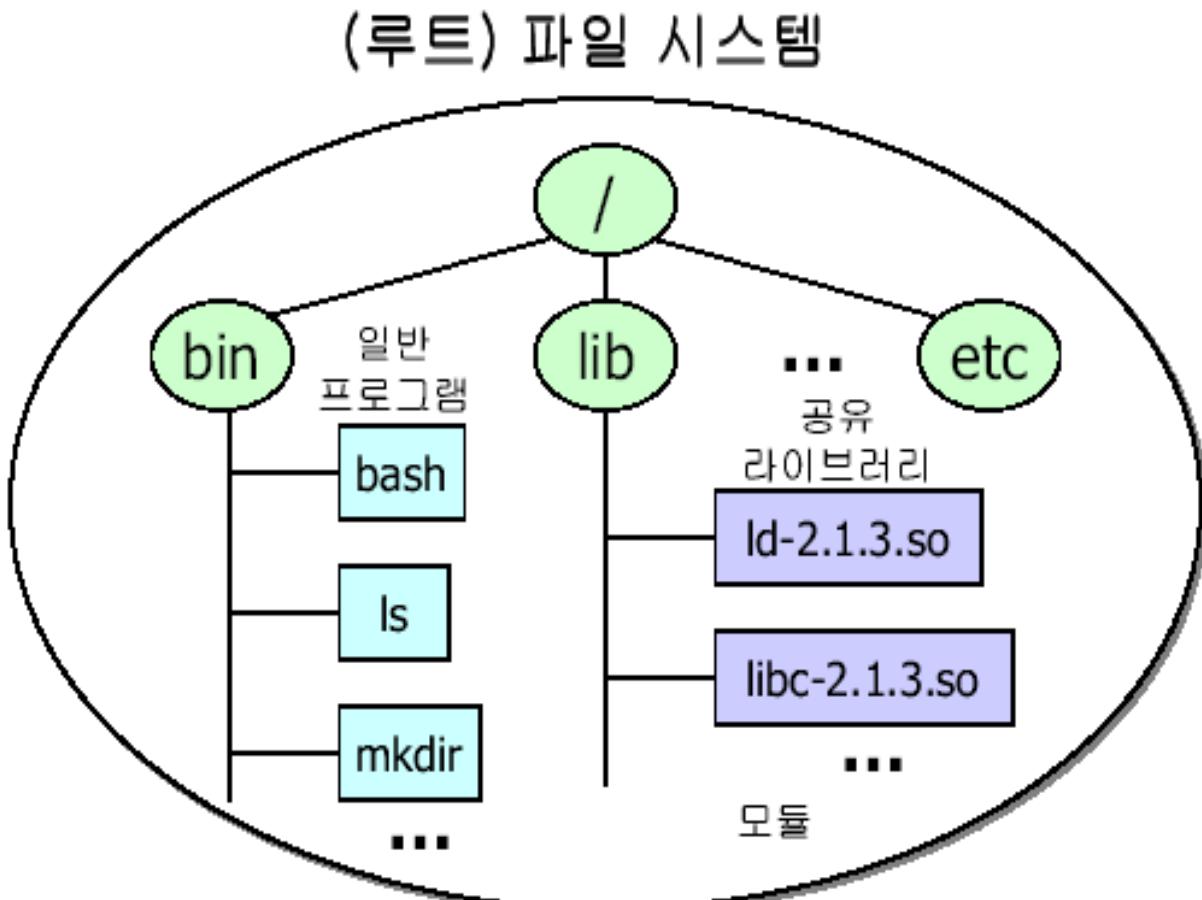


- 커널이미지

커널이 하나의 파일로
디스크에 저장되어 있는 것
zImage 와 bzImage 커널 구조

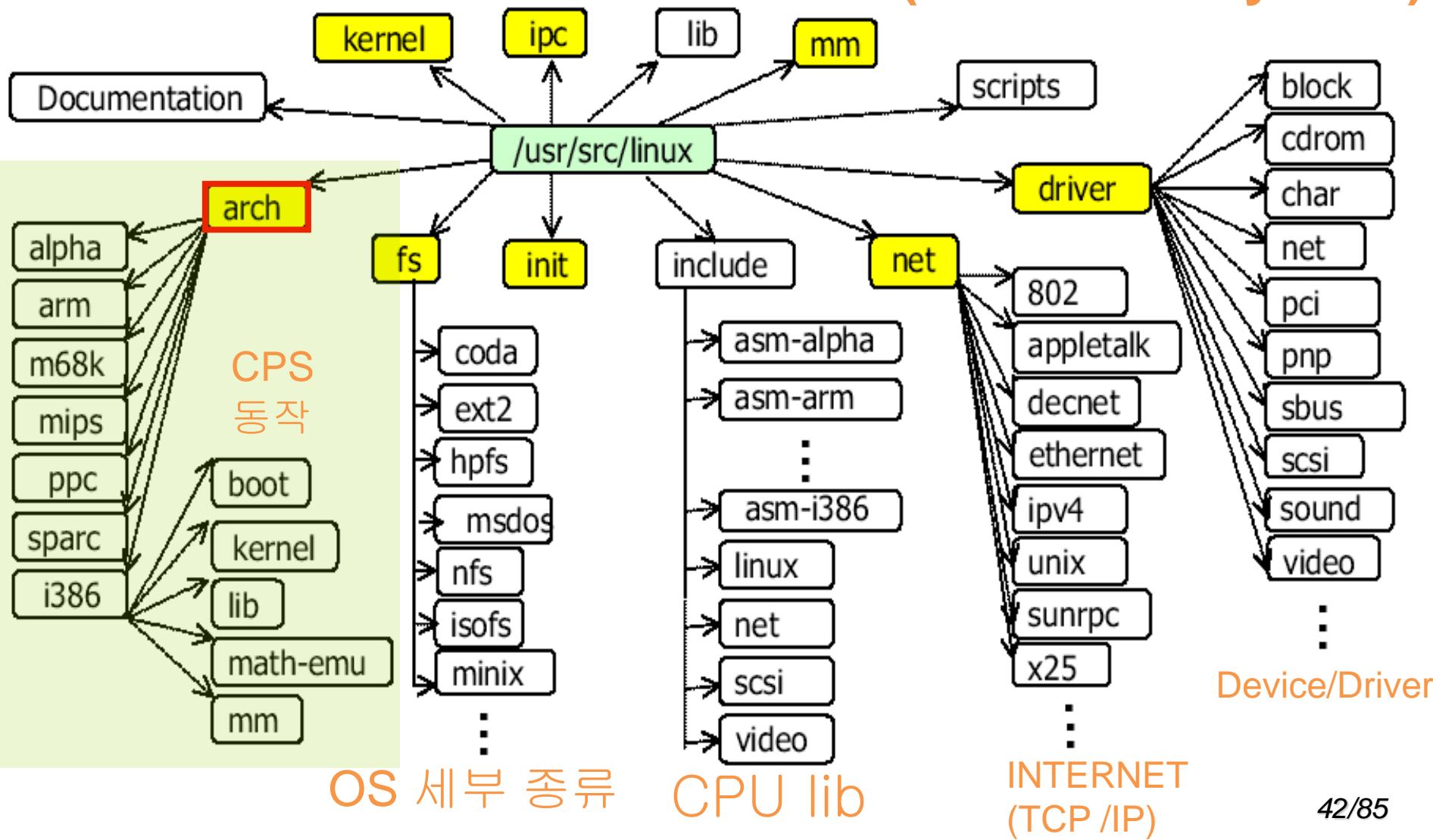
- 로더(LILO 또는 loadlin)

커널 이미지를 찾아서 커널을
메모리에 올려놓는 일을 수행



임베디드 리눅스의 구성 (커널 소스 구조)

VFS(Virtual file System)



임베디드 리눅스용 시스템 개발 도구

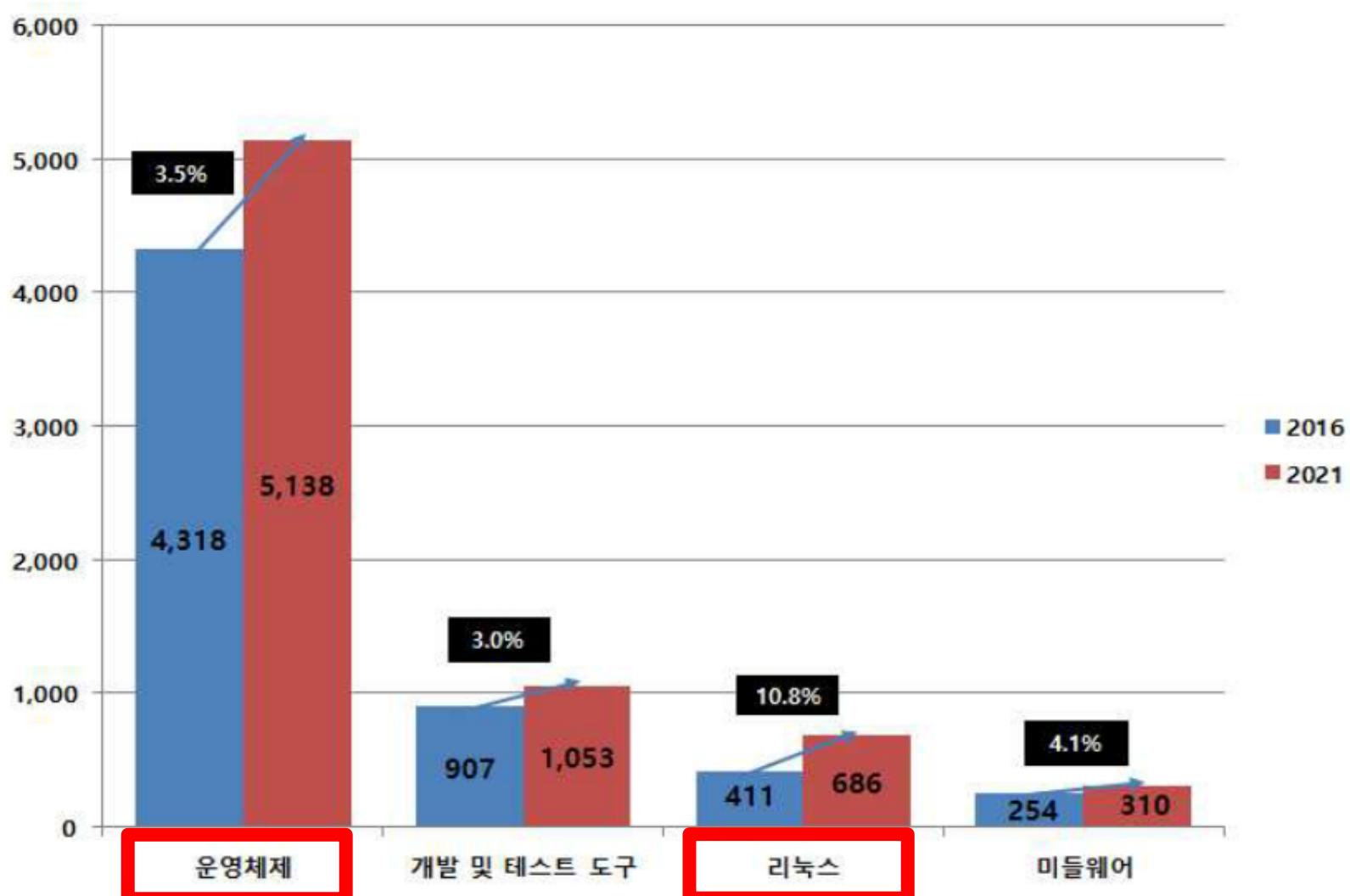
제 품	기 능	Qplus-p Target Builder	Embedix Target Wizard	Montavista HardHat Linux	Red Hat ELDS
Integrated Config (kernel+app)	YES	YES	NO	NO	
Config Language	CML2(+QPD)	ECD	?	XML	
Dependency Resolving	YES	YES	NO	NO	
Kernel Upgrade	Easy	Hard (Make ECD)	Easy	Easy	
Package Maintenance method	RPM, SRPM	SRPM	prebuilt binary	SRPM	
Library Optimization	YES	YES	YES	?	
Deploy Method Support	GOOD	GOOD	POOR	POOR	
Support Package	<20	>100	>100	?	
Platform(CPU)	x86	x86,PPC, StrongArm, SH,MIPS	x86, PPC, StrongArm, SH, MIPS	x86,PPC, StrongArm,SH	
User Friendly	GOOD	VERY GOOD	POOR	POOR	43/85

임베디드 리눅스 응용

	Smart Mobile Phone	PDA/ Web Pad	Web Screen/Video Phone	Internet Digital Set-top Box	Industry Application
DRAM	2 – 32MB	4 – 32MB	8 – 32MB	8 – 32MB	2 – 64MB
Flash (Mask ROM)	4 – 32MB	8 – 32MB	16 – 32MB	16 – 32MB	2 – 64MB
Special Features	Small Memory Low Power	General Platform	Multimedia Communication	Real-time Requirement	High Availability
Special Devices	Dual CPU (Phone + PDA)	USB, MMC	Camera, ISDN, ADSL, ...	MPEG2, ...	PCI, DOC, ...
Applications	Phone, Small GUI, WAP	MP3	H.32x (VoIP)	Internet Shopping	Real-time Data Acquisition

임베디드 운영체계 시장

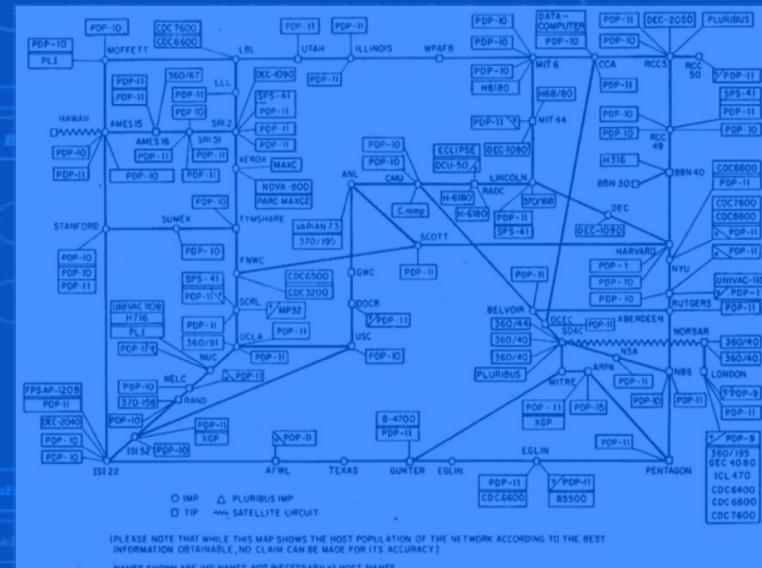
(단위: 백만 달러)



(출처: BCC Research, Embedded Systems: Technologies and Markets, 2016)

학습 내용 정리

- 임베디드 시스템의 운영체제에 대해 살펴본다.
 - 임베디드 시스템의 운영체제 구성에 대해 알아본다.
 - 임베디드 시스템의 운영체제 주요 기능에 대해 살펴본다.
 - 임베디드 시스템의 운영체제 종류에 대해 알아본다.
 - 임베디드 리눅스 개요에 대해 알아본다.
 - 임베디드 리눅스 특징에 대해 살펴본다.
 - 임베디드 리눅스 구조에 대해 알아본다.
 - 임베디드 리눅스 응용에 대해 살펴본다.





Thank you for listening

임베디드 시스템의 소프트웨어 개발 환경 구축

제주대학교

김 도 현

주요 학습 내용

- 임베디드 시스템의 소프트웨어 개발 과정

- 임베디드 시스템의 소프트웨어 개발 환경

- 임베디드 시스템의 소프트웨어 개발 환경 구축

1. 플래시 메모리에 Bootloader 탑재 : JTAG Fusing 프로그램

2. 임베디드 S/W 크로스 컴파일(Cross Compiler Toolchain)

3. 임베디드 시스템 원격 접속 및 제어(Minicom)

4. 주소 할당 및 Bootp 데몬 설치

5. TFTP를 이용한 커널, 사용자 파일 전달

15 Billion
connected devices

5 Billion
connected devices

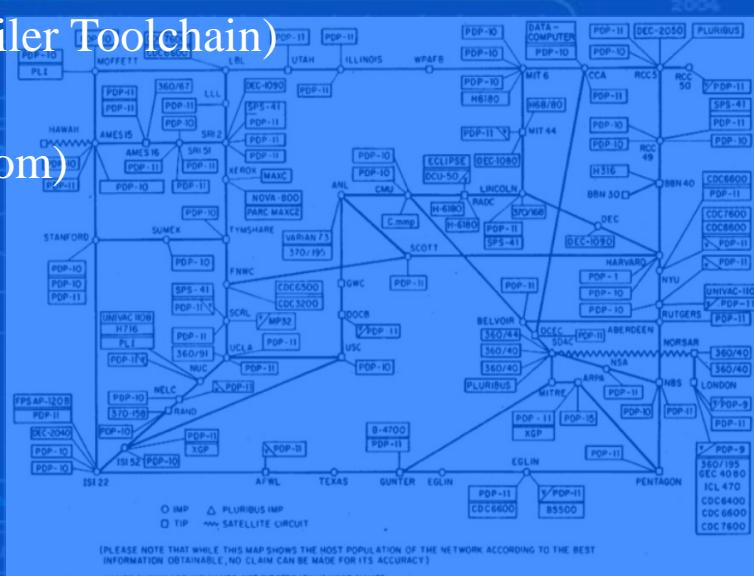
2 Billion
connected devices

93,047,785
connected devices

313,000
connected devices

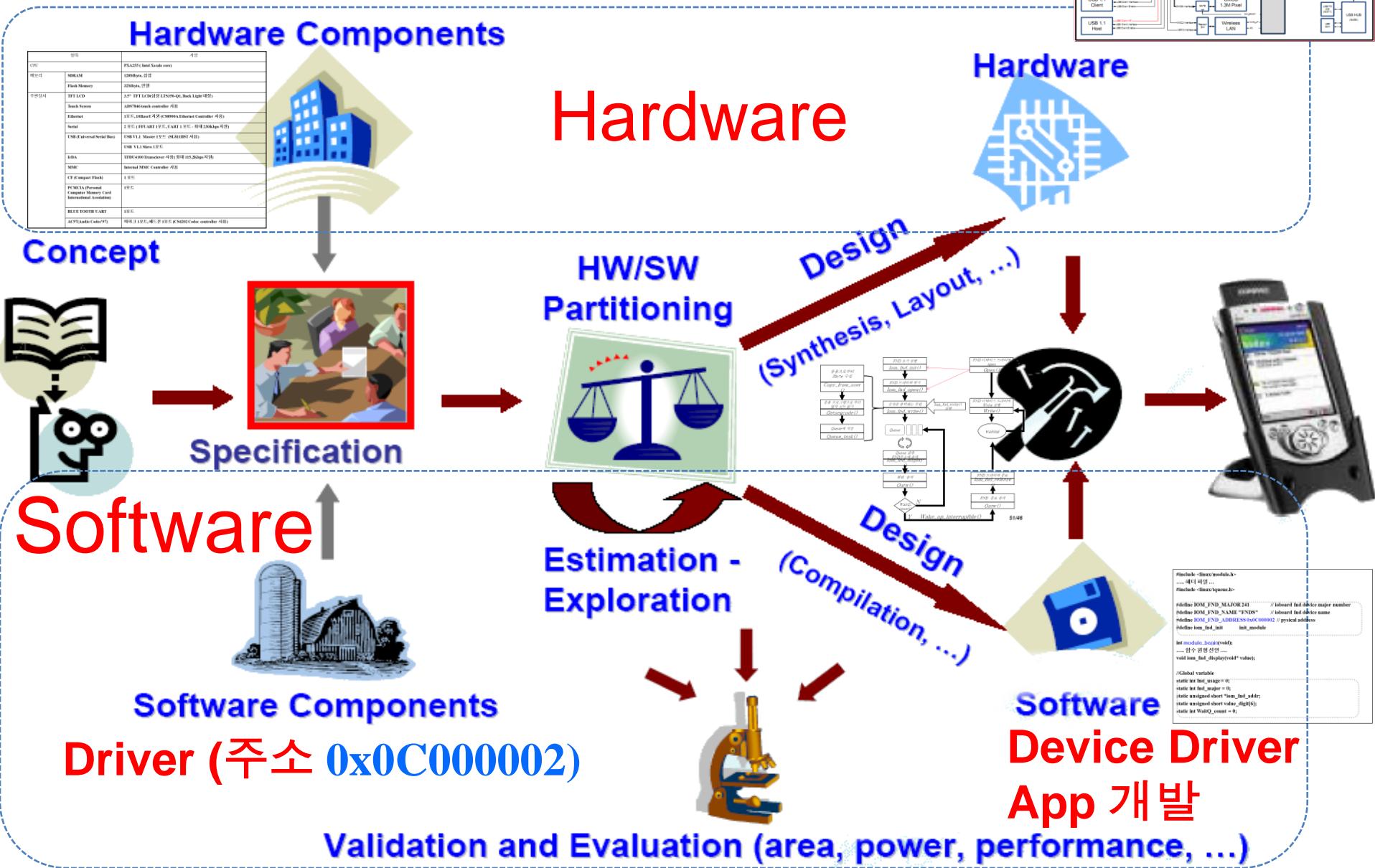
188
connected devices

13
connected devices



(PLEASE NOTE THAT WHILE THIS MAP SHOWS THE HOST POPULATION OF THE NETWORK ACCORDING TO THE BEST INFORMATION OBTAINABLE, NO CLAIM CAN BE MADE FOR ITS ACCURACY)
NAMES SHOWN ARE IMP NAMES, NOT NECESSARILY HOST NAMES

임베디드 시스템 개발 과정

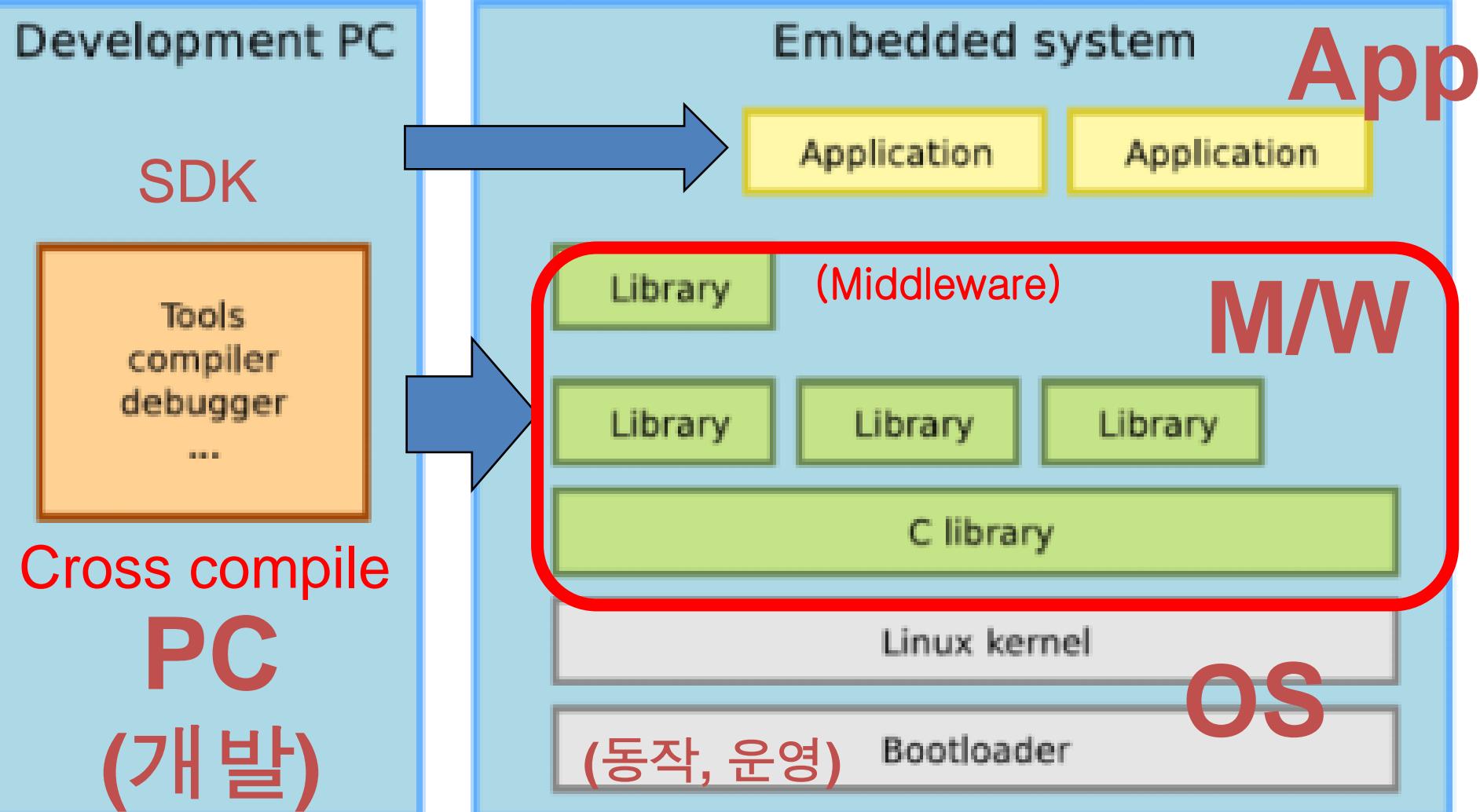


임베디드 시스템의 소프트웨어 개발 과정

- 임베디드 시스템의 소프트웨어 개발 과정
 - 임베디드 소프트웨어 개발 환경 구축
 - 디바이스 드라이버 개발
 - 기본 응용 프로그램 개발 (디바이스 드라이버 테스트)
 - 다양한 네트워크 기능, GUI(Graphic User Interface) 등의 기능 개발
 - 프로그램 통합 및 패키징

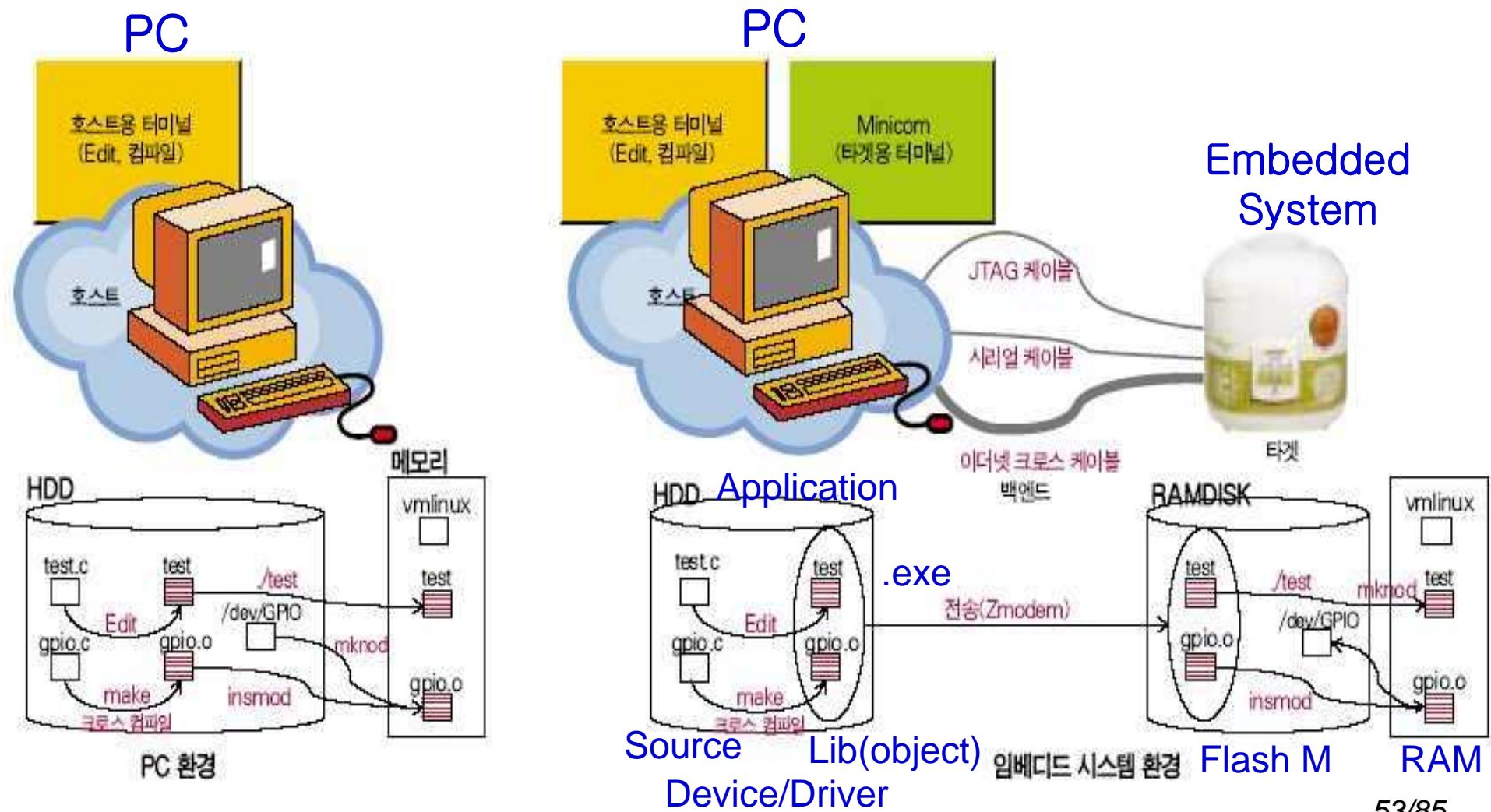
임베디드 시스템의 소프트웨어 개발 환경

- 임베디드 시스템의 프로그램은 PC 개발하여.. 임베디스 하드웨어에 Upload(deploy, Porting)



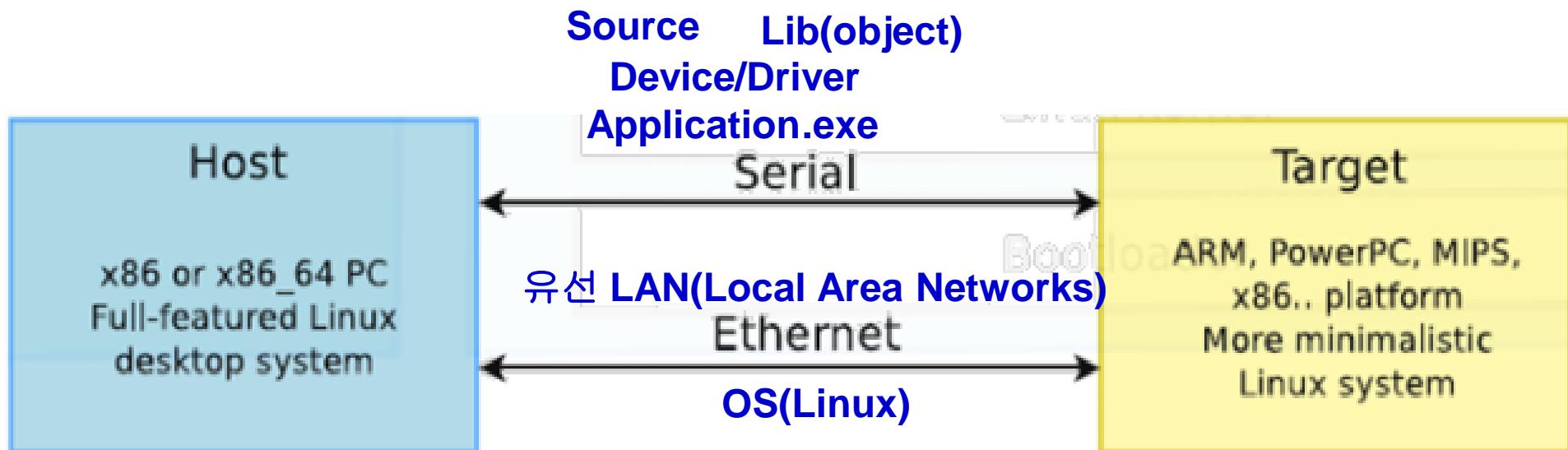
임베디드 시스템의 소프트웨어 개발 환경

- 호스터 컴퓨터와 임베디드 시스템 연결



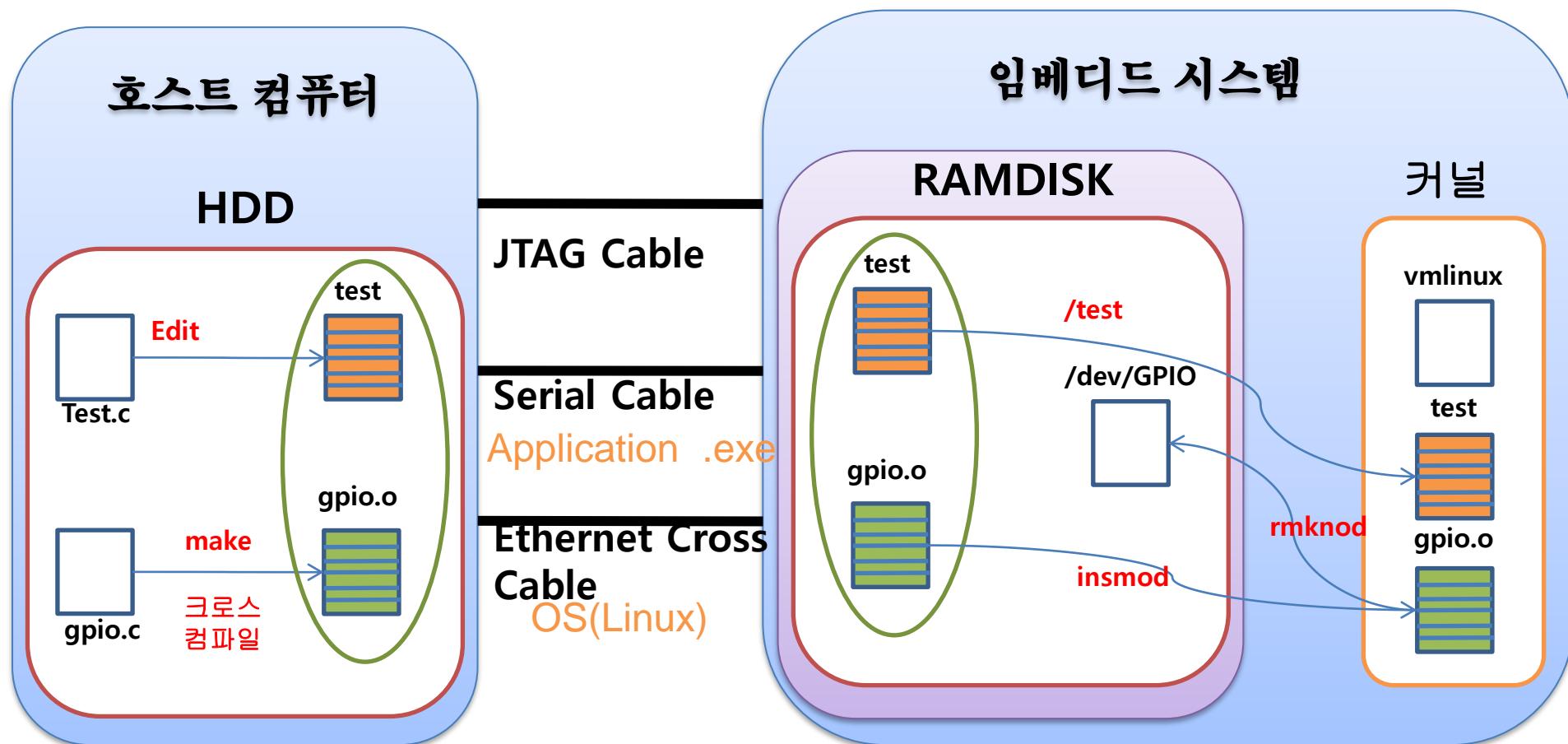
임베디드 시스템의 소프트웨어 개발 환경

- 호스터 컴퓨터와 임베디드 시스템 연결



임베디드 시스템의 소프트웨어 개발 환경

- 호스터 컴퓨터와 임베디드 시스템 연결



임베디드 시스템의 소프트웨어 개발 환경 (디바이스 드라이브 및 응용 개발 환경 구축 절차)

호스트 컴퓨터(pc)와 임베디드 시스템 연결
(**JTAG Cable + Parallel Cable**, Serial Cable, Ethernet Cross Cable)



임베디드 프로그램 개발 환경(컴파일)
(Cross Compiler Toolchain 설치)



플래시 메모리에 **부트로드 탑재**
(JTAG Fusing 프로그램 설치
Using JTAG Cable + Parallel Cable)



원격 접속 및 제어 (**Minicom** 프로그램 설치 및 설정)
Using Serial Cable

```
[root@huins FND_Device]$ls
fnd_driver.o test_fnd
[root@huins FND_Device]$insmod fnd_driver
init module, fndioprt major number : 2
[root@huins FND_Device]$mknod /dev/FND 2 2
[root@huins FND_Device]$./test_fnd 0x1234
[root@huins FND_Device]$
```

임베디드 운영체계 및 사용자 파일 전달

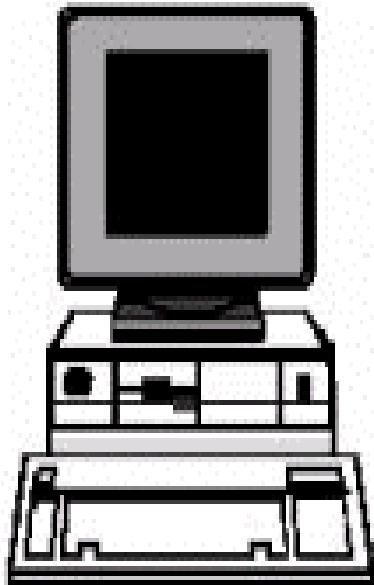
Using Ethernet Cross Cable

- 주소 할당 (Bootp 데몬 설치)

- 파일 전송 (TFTP 프로그램 설치 및 설정)

임베디드 시스템의 소프트웨어 개발 환경 요소

호스트 컴퓨터

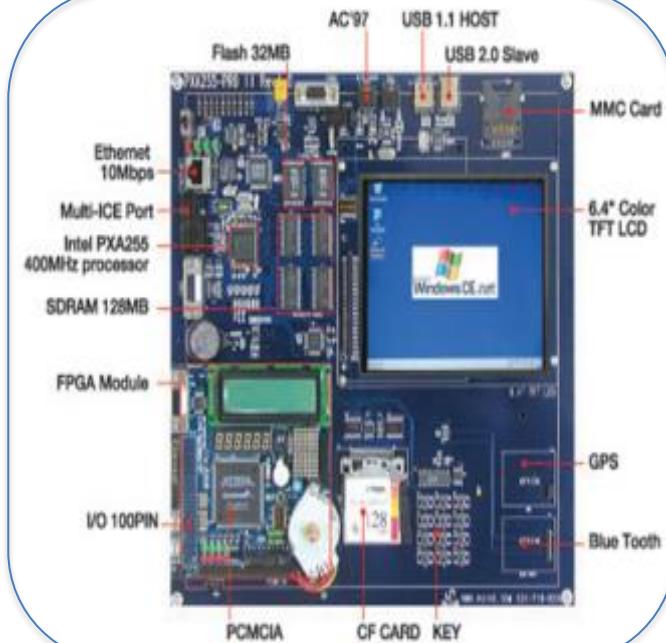


Hardware

Cable 연결

- JTAG Cable/Parallel Cable
- Serial Cable
- Ethernet Cross Cable

타겟 임베디드 시스템



Software

크로스
컴파일러
(Toolchain)

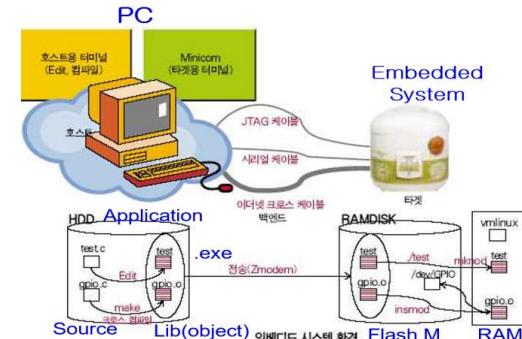
하드웨어
제어
프로그램
(JTAG
Fusing)

원격접속
프로그램
(Minicom
또는 GTK
Term)

커널 / 데이터
전송 프로그램
(Bootp,
TFTP)

임베디드 시스템의 소프트웨어 개발 환경 구축 (프로그램 설치 및 명령어)

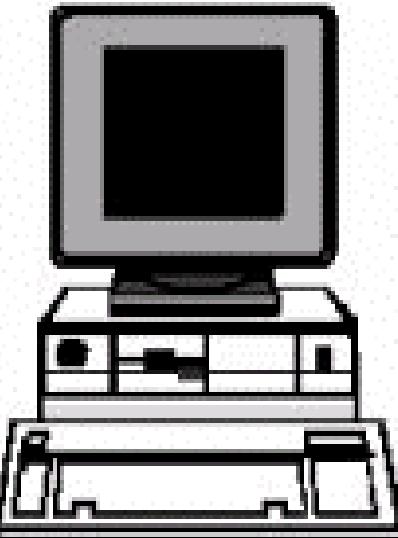
- 임베디드(임베디드 시스템) 시스템 교차 컴파일 개발환경(cross compile) 구축 과정
 - 호스트에 임베디드 시스템 디바이스용 리눅스를 개발하기 위한 모든 환경을 구축
 - 부트로더를 컴파일하여 해당 코드를 플래시 메모리에 탑재
 - 부트로더를 수정하여 원하는 기능을 구현
 1. 플래시 메모리에 Bootloader 탑재 : JTAG Fusing 프로그램 설치
 2. 임베디드 S/W 컴파일 : Cross Compiler Toolchain 설치
 3. 임베디드 시스템(embedded system:PX-A255-PRO) 접속 및 제어 : Minicom 설정
 4. 주소 할당 : Bootp 데몬 설치 (호스트와 임베디드 시스템 (embedded system:PX-A255-PRO))
 5. 커널, 사용자 파일 전달 : TFTP 설정



임베디드 시스템의 소프트웨어 개발 환경 구축 (프로그램 설치 및 명령어)

HOST System

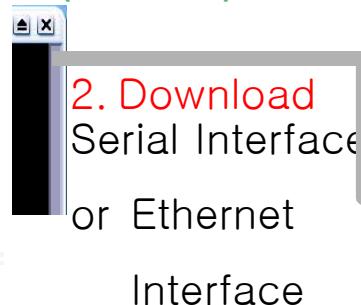
- 임베디드 S/W 컴파일
(Cross Compiler Toolchain 설치)



- 플래시 메모리에 Bootloader 탑재
(JTAG Fusing 프로그램 설치)

• 임베디드 시스템(embedded system:PXA255-PRO) 제어
(Minicom 설정)

- 주소 할당
(Bootp 데몬 설치)
- 커널, 사용자 파일 전달
(TFTP 설정)



JTAG Interface

1.Fusing

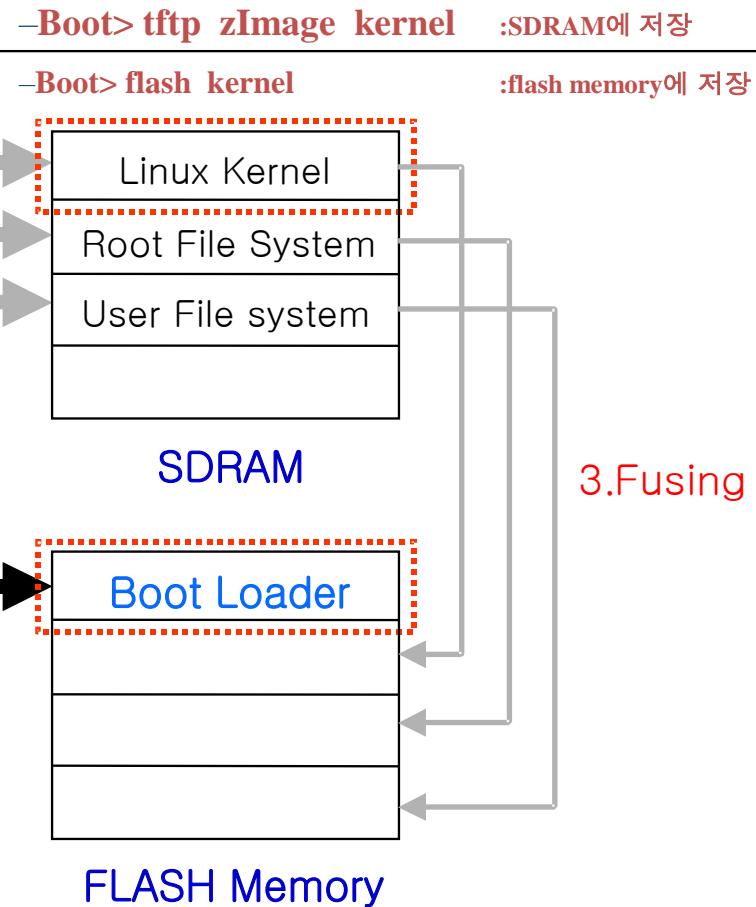
#jtag

jtag) cable parallel 0x378 PXA255

jtag) detect

jtag) flashmem 0 <파일명>

임베디드 시스템(embedded system:PXA255-PRO)



임베디드 시스템의 소프트웨어 개발 환경 구축 (Flash memory (HDD) 프로그램 구성)

- Flash memory (HDD) 구성

- 부트로드

- Blob: 256 Kbytes

- Param Block: 256 Kbytes

- 커널

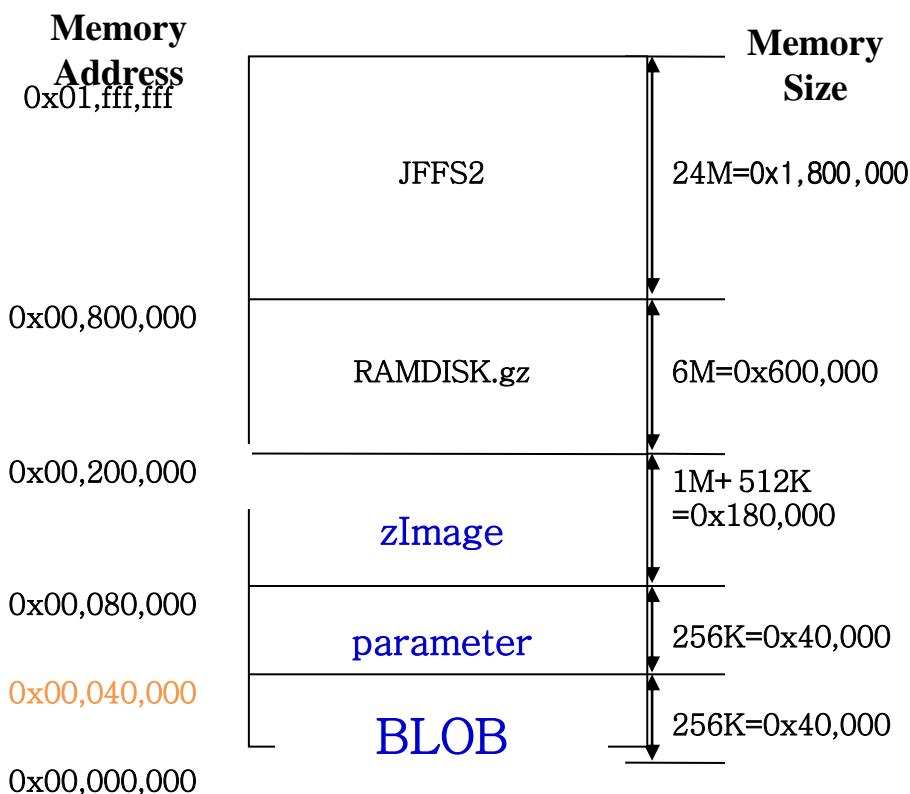
- Kernel: 2 Mbytes

- 파일 시스템

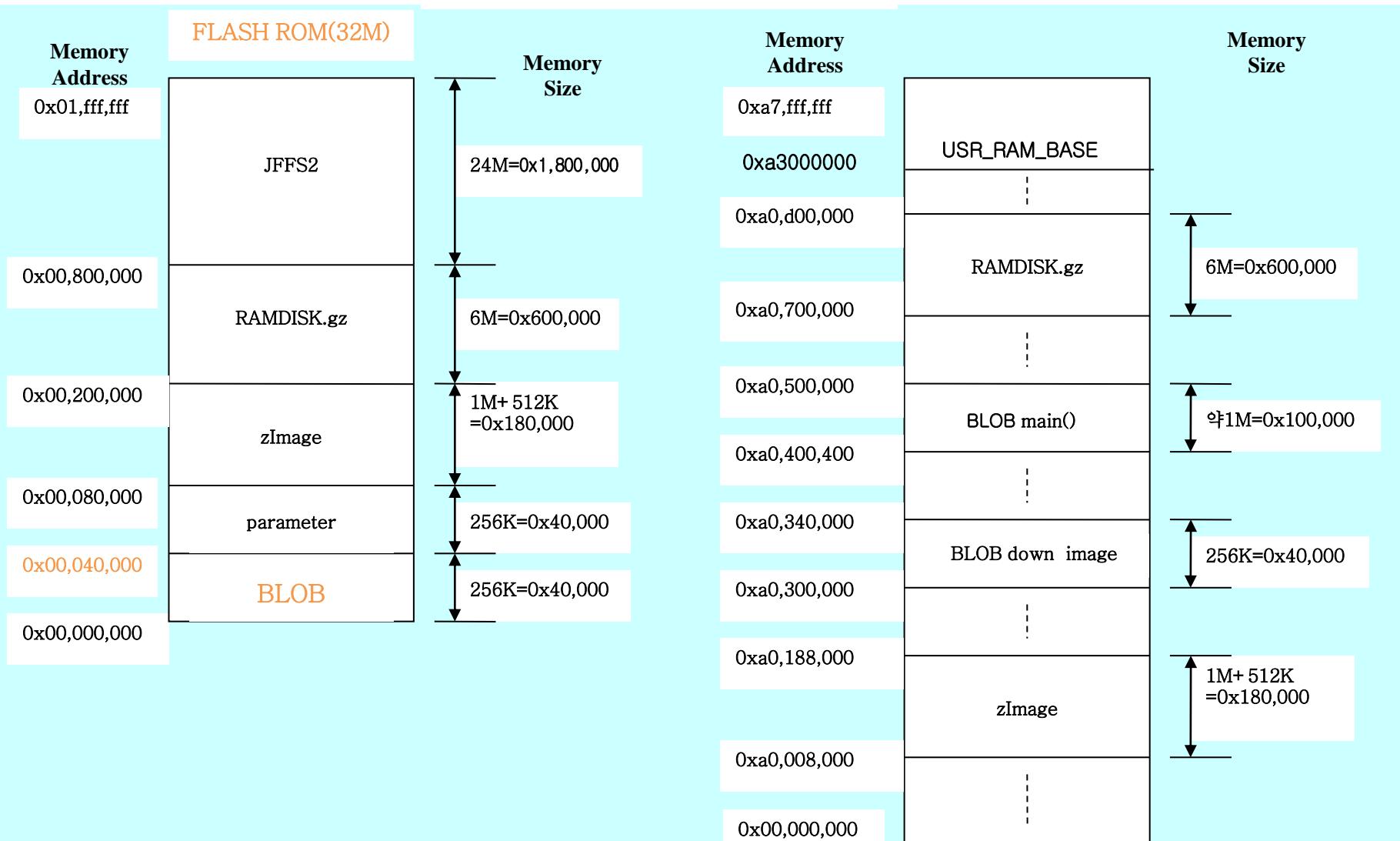
- Ramdisk: 2 Mbytes

- 사용자 파일 시스템

- User file system: 27 Mbyte + 512 Kbytes

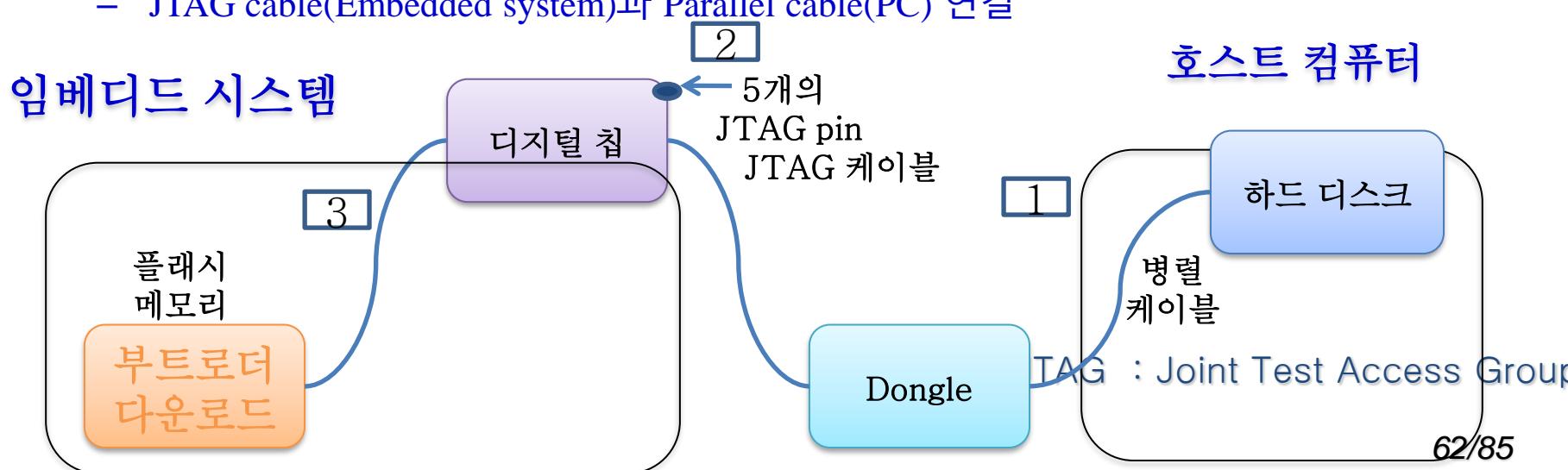


임베디드 시스템의 소프트웨어 개발 환경 구축 (임베디드(임베디드 시스템) 시스템의 Memory Map)



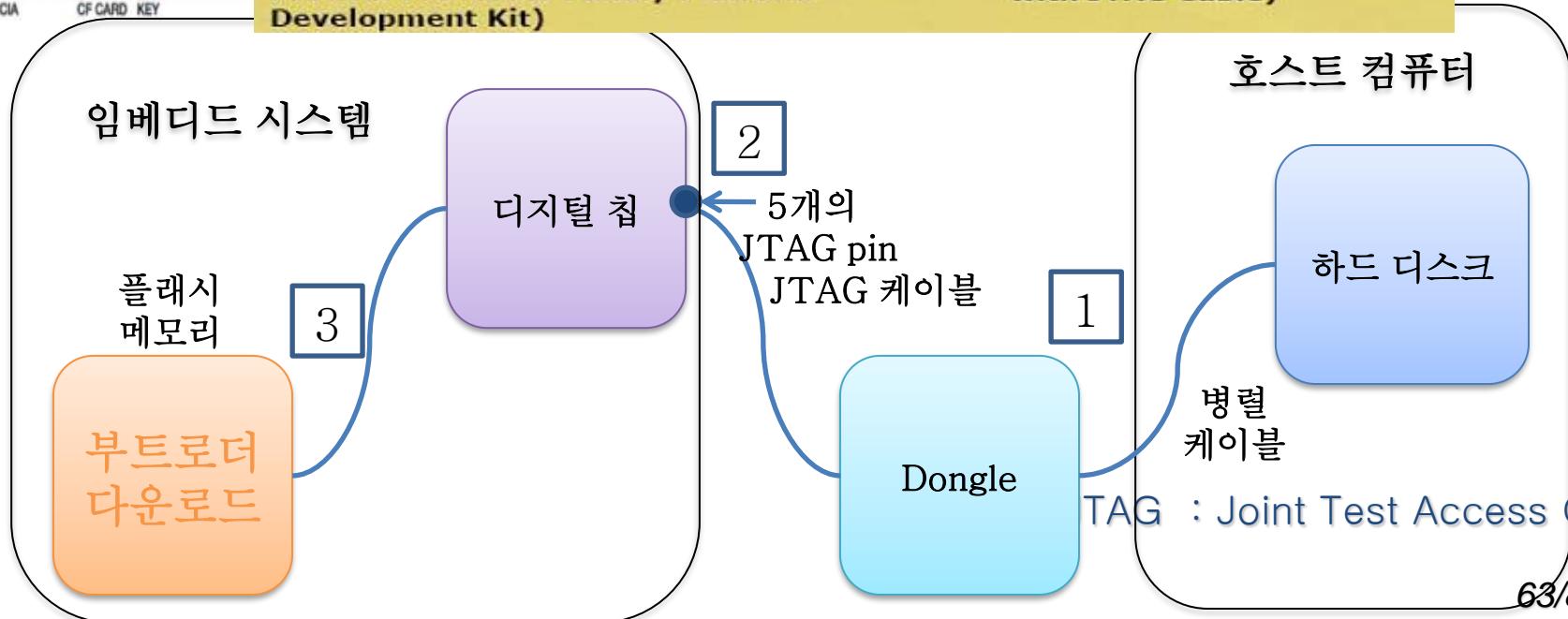
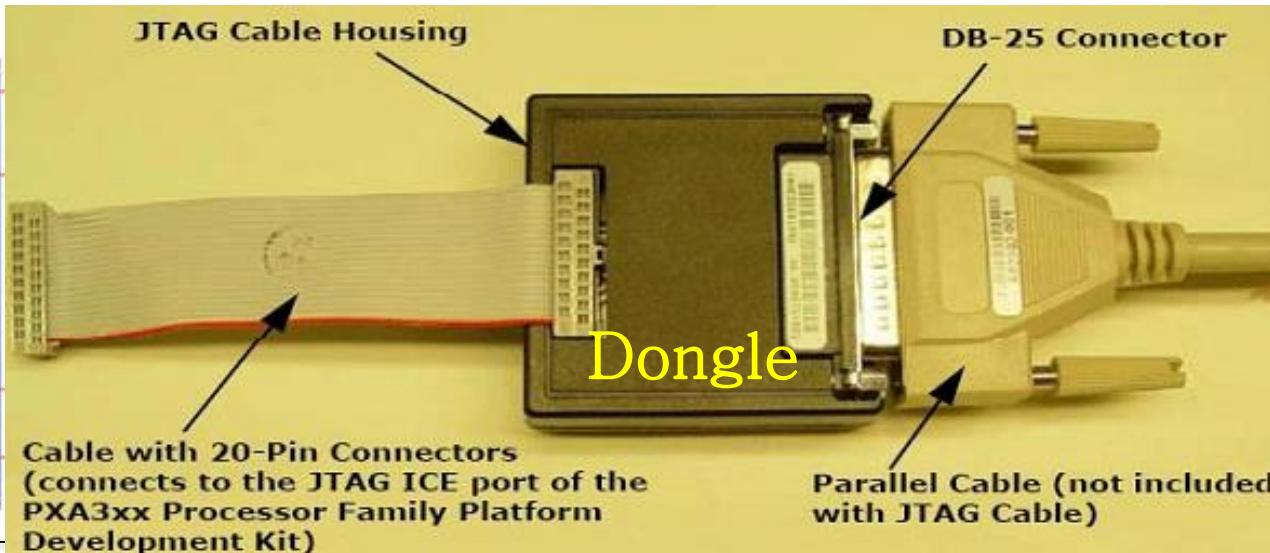
임베디드 시스템의 소프트웨어 개발 환경 구축 (부트로더 설치)

- 임베디드 시스템의 플래시 메모리에 Bootloader 탑재
 - JTAG Fusing 프로그램 설치
 - XScale 전용 디버깅 케이블인 Marvell Black Stone Cable을 이용하여 JTAG을 사용하기위해 필요한 프로그램
- JTAG (Joint Test Access Group) cable
 - PCB(printed circuit boards)와 IC(Integrated Circuit)를 테스트 목적
 - 플래시 메모리에 부트로드 탑재
- 병렬 케이블(Parallel cable)
 - JTAG cable(Embedded system)과 Parallel cable(PC) 연결



임베디드 시스템의 소프트웨어 개발 환경 구축

(부트로더 설치, JTAG을 이용한 Flash 메모리 fusing)



임베디드 시스템의 소프트웨어 개발 환경 구축 (부트로더 설치)

- **부트로더(Bootloader) (Booting : 초기 시동)**
 - 초기적재프로그램
 - 운영 체제가 시동되기 이전에 미리 실행되면서 커널이 올바르게 시동되기 위해 필요한 모든 관련 작업을 마무리하고 최종적으로 운영 체제를 시동시키기 위한 목적을 가진 프로그램
- **바이오스(BIOS; Basic Input/Output System)**
 - 가장 기본적인 소프트웨어이자 컴퓨터의 입출력을 처리하는 펌웨어
 - 운영 체제 중 가장 기본적인 소프트웨어이자 컴퓨터의 입출력을 처리하는 기능 프로그램



임베디드 시스템의 소프트웨어 개발 환경 구축 (부트로더 설치)

- 부트로더 Bootloader 기능

- 하드웨어 보드 초기화

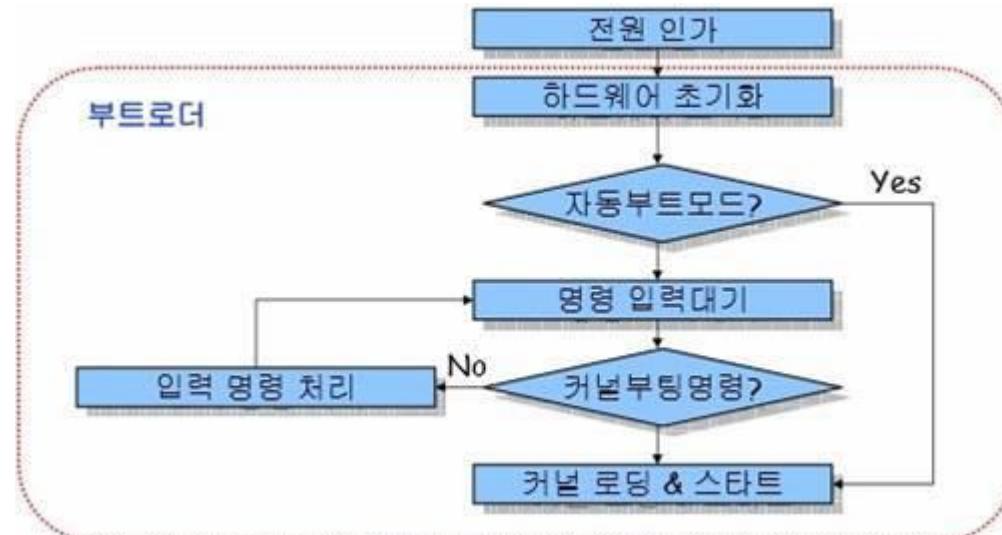
- Memory setting
 - CPU Clock setting
 - GPIO setting
 - Serial setting
 - MAC address 획득 및 Ethernet port setting

- 커널 부트 관련 기능

- Image download from host by tftp
 - Copy image from Flash to ram or from ram to flash.
 - Jump to kernel
 - Command mode 제공

- 하드웨어 보드 상태검사

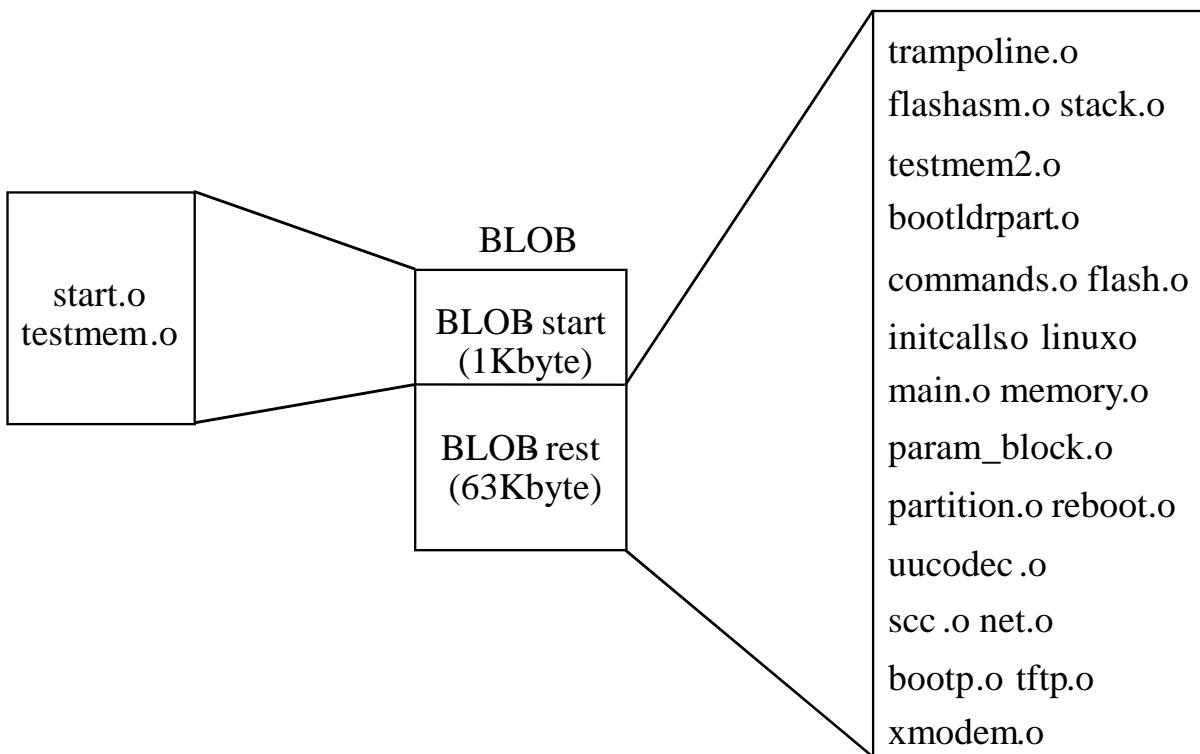
- Command line interface
 - Status, set, erase



임베디드 시스템의 소프트웨어 개발 환경 구축 (부트로더 설치)

- JTAG 케이블을 이용한 부트
로더 Fusing 프로그램 설치

JTAG Fusing : `./configure --with-include=/temp_cd/JTAG/pxa255-jtag /include-0.2.3 --prefix=/temp_cd/JTAG/jtag`



```
# mkdir /temp_cd/JTAG
# cd /temp_cd/JTAG
# cp /mnt(media)/cdrom/Application/pxa255-jtag.tar.gz .
# tar xfz pxa255-jtag.tar.gz
# cd pxa255-jtag
# cd jtag-0.4
# ./configure
# make
# make install
# cd /temp_cd/JTAG/jtag/bin
# cp jtag /usr/bin
# jtag
```

임베디드 시스템의 소프트웨어 개발 환경 구축 (부트로더 설치)

- JTAG 케이블과 프로그램을 이용한 부트로더 Fusing 과정

```
# jtag
jtag) cable parallel 0x378 PXA255
jtag) detect
jtag) flashmem 0 <파일명>

# cd /temp_cd/Bootloader/blob/src/blob
# jtag
jtag) cable parallel 0x378 PXA255
jtag) detect
jtag) flashmem 0 blob
```



The terminal window shows the following session:

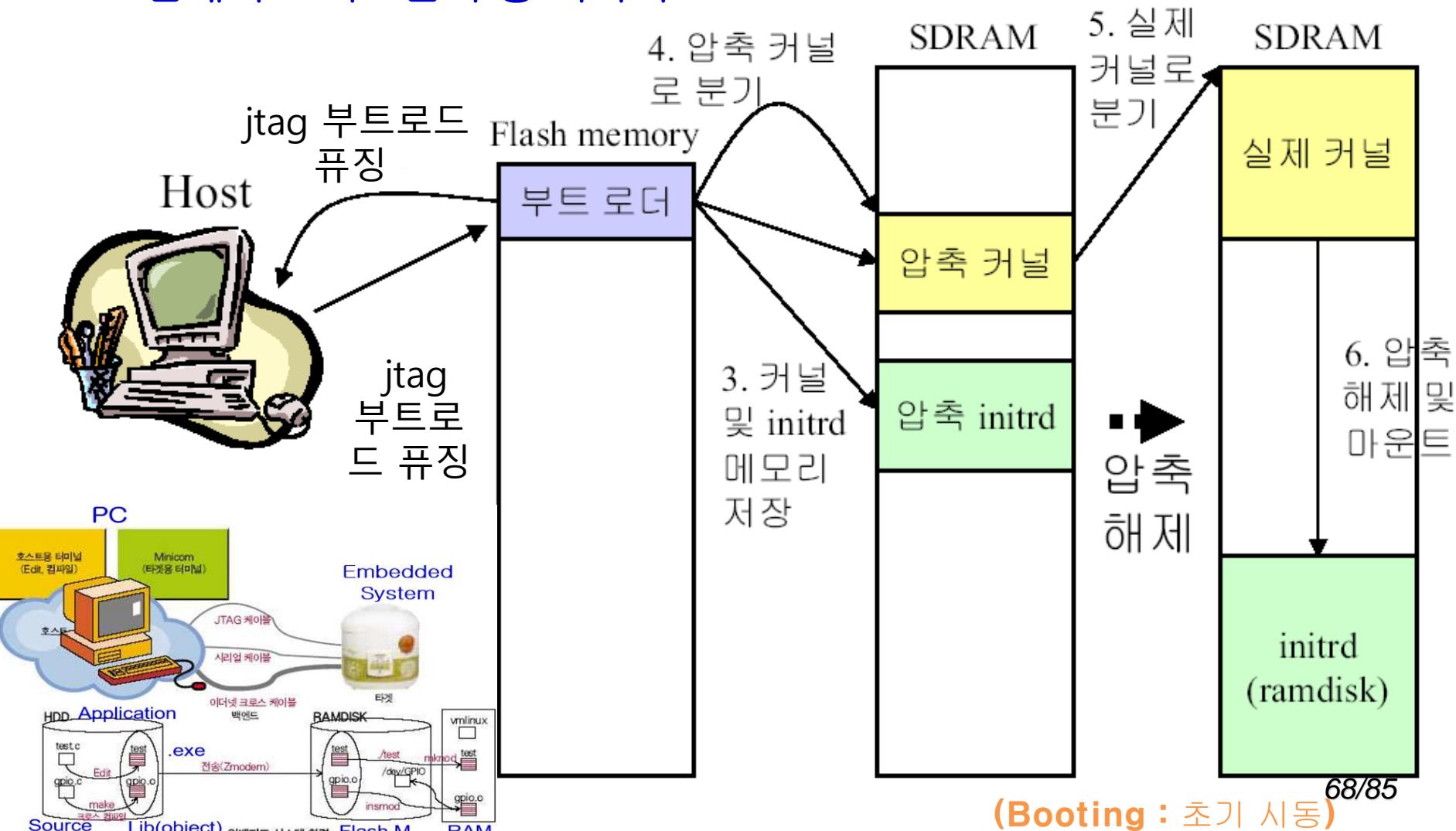
```
root@localhost:/usr/local/pxa255/blob
파일(E) 편집(E) 보기(V) 터미널(T) 가기(G) 도움말(H)
Filename :blob,
Load Addr : A0300000
Loading: #####
done
Bytes transferred = 56873 (DE29 hex)
Received 56873 (0x0000de29) bytesboot>
boot> erase 0x00000000 0x00040000
Erase at 0x00000000erasing dirty block at 0x00000000
boot> flash blob

      nwords = 14219      Saving blob to flash
0000378b (14219) words from 0xa0300000 to 0x00000000
Writing at 0x0000dac0
14219 words source image
14213 words written to flash
6 words skipped
0 erase operations
0 words scanned down

boot> █
```

임베디드 시스템의 소프트웨어 개발 환경 구축 (부트로더 설치)

- 임베디드 시스템 부팅 시나리오



임베디드 시스템의 소프트웨어 개발 환경 구축 (교차 컴파일)

- 임베디드 S/W 컴파일러 설치 (PC 컴파일, 임베디드에서 실행)
 - 컴파일러가 실행되는 플랫폼이 아닌 다른 플랫폼에서 실행 가능한 코드를 생성할 수 있는 컴파일러
 - Cross Compiler Toolchain 설치 (기계어로 변환, PC-> E/System)
 - GCC : 컴파일러
 - Binutils : 어셈블러 및 로더, 바이너리 파일 편집 유틸리티
 - Glibc : 크로스 컴파일을 위한 라이브러리 및 일반 라이브러리

```
# cd /temp_cd/ Toolchain
# tar xvzf arm-cross-compiler.tar.bz2
# cp -rf arm-linux/usr/local
# vi
    /root/.bash_profile
# source
    /root/.bash_profile
```

임베디드 시스템의 소프트웨어 개발 환경 구축 (교차 컴파일)

- test_compiler.c 를 서로 다른 compiler로 compile할 경우
 - object code 두 개가 생성
 - # **gcc** -o host_compiler test_compiler.c
 - ./host_compiler 를 실행한다.
 - 그러면 정상으로 progam이 실행되어 결과가 화면에 display 된다.
 - file ./host_compiler (파일의 유형 확인)
 - # **arm-linux-gcc** -o cross_compiler test_compiler.c
 - file ./cross_compiler (파일의 유형 확인)
 - ./cross_compiler를 실행한다. 그러면 실행이 되지 않음.
 - 이것은 arm processor으로 compile되었기 때문에 실행이 안됨.
 - 나중에 임베디드 시스템 보드(embedded system:PXA255-PRO)에 download해서 실행시키면 정상적으로 실행이 됨.

임베디드 시스템의 소프트웨어 개발 환경 구축 (교차 컴파일)

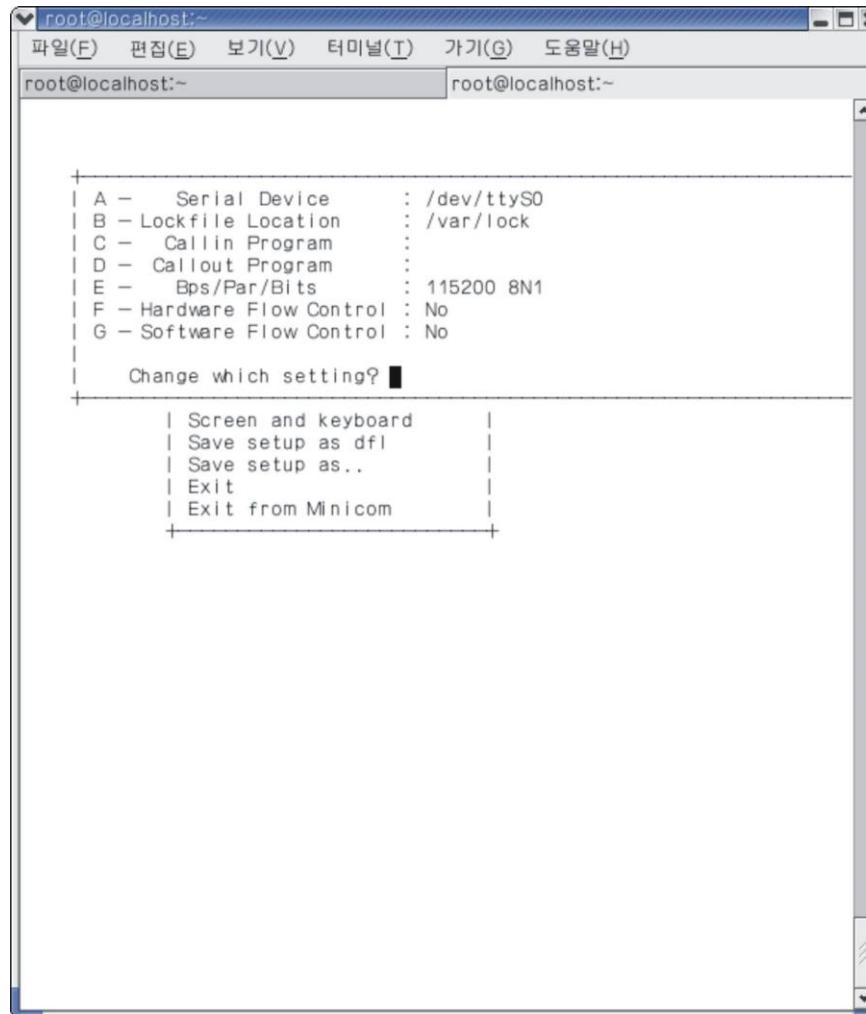
```
root@localhost:/home/negi97/test
파일(F) 편집(E) 보기(V) 터미널(T) 가기(G) 도움말(H)
root@localhost:~          root@localhost:/usr/local/ar... root@localhost:/home/negi97...
[root@localhost test]# ls
test_compiler.c
[root@localhost test]# cat test_compiler.c
#include <stdio.h>

int main(void)
{
    printf (" This is test_program for compiler~!\n");
    return 0;
}
[root@localhost test]# gcc -o host_compiler test_compiler.c
[root@localhost test]# arm-linux-gcc -o cross_compiler test_compiler.c
[root@localhost test]# ls
cross_compiler host_compiler test_compiler.c
[root@localhost test]# ./host_compiler
This is test_program for compiler~
[root@localhost test]# ./cross_compiler
bash: ./cross_compiler: cannot execute binary file
[root@localhost test]# file host_compiler
host_compiler: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), f
or GNU/Linux 2.2.5, dynamically linked (uses shared libs), not stripped
[root@localhost test]# file cross_compiler
cross_compiler: ELF 32-bit LSB executable, ARM, version 1 (ARM), for GNU/L
inux 2.0.0, dynamically linked (uses shared libs), not stripped
[root@localhost test]# █
```

임베디드 시스템의 소프트웨어 개발 환경 구축 (원격 접속)

- 임베디드 시스템(embedded system:PXA255-PRO) 제어 (원격 접속)

- Minicom 또는 GTK Term 설정
- 주소 할당
- Bootp 데몬 설치
- 직렬 케이블 (Serial Cable) 이용
 - 9핀
 - com1 또는 com2
 - Embedded system 원격 접속 및 제어



임베디드 시스템의 소프트웨어 개발 환경 구축 (원격 접속)

- 임베디드 시스템(embedded system:PXA255-PRO) 제어 (원격 접속)

Aovlex-link

- Minicom 설정

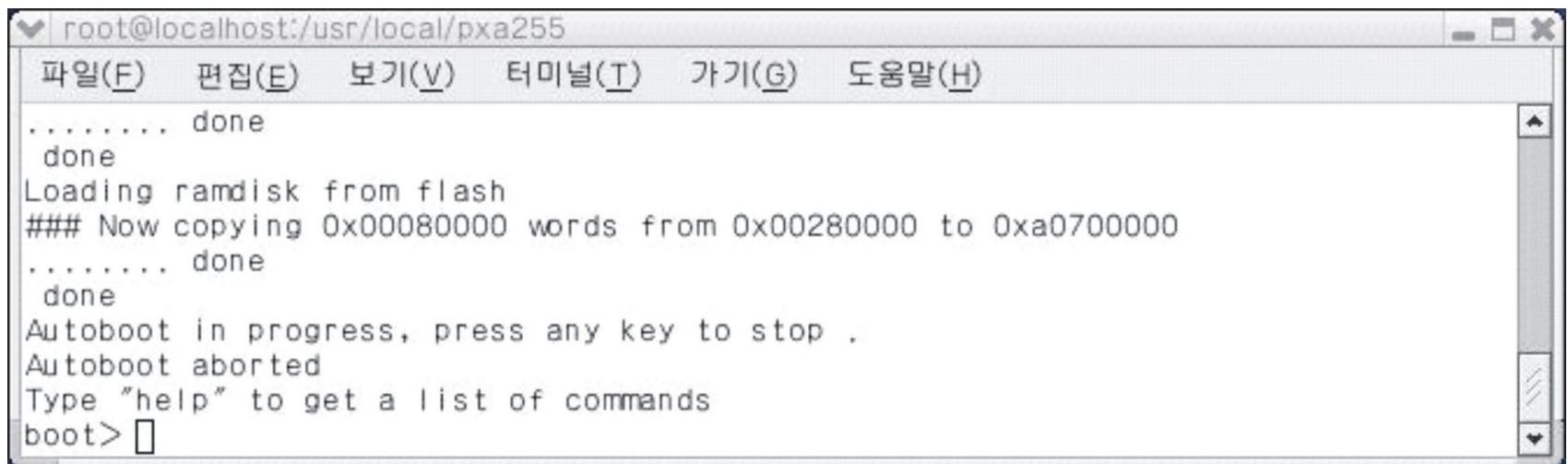
```
# minicom -s  
  
- minicom 접근 -  
Serial Port Setup  
Save setup as dfl Exit
```



```
root@localhost:~  
파일(F) 편집(E) 보기(V) 터미널(T) 가기(G) 도움말(H)  
[root@huins nfs]$ ./test_app  
test_open start!!!  
read() ^C  
ioctl() ^C  
[root@huins nfs]$
```

임베디드 시스템의 소프트웨어 개발 환경 구축 (원격 접속)

- 임베디드 시스템(embedded system:PXA255-PRO) 제어 (원격 접속)
 1. blob으로 부팅
 2. minicom 을 실행하여 원격 접속
 3. blob command mode로 임베디드 시스템(embedded system:PXA255-PRO) booting



The screenshot shows a terminal window with the following text output:

```
root@localhost:/usr/local/pxa255
파일(F) 편집(E) 보기(V) 터미널(T) 가기(G) 도움말(H)
..... done
done
Loading ramdisk from flash
### Now copying 0x00080000 words from 0x00280000 to 0xa0700000
..... done
done
Autoboot in progress, press any key to stop .
Autoboot aborted
Type "help" to get a list of commands
boot> 
```

임베디드 시스템의 소프트웨어 개발 환경 구축 (원격 접속 및 주소설정)

- IP 어드레스를 '0.0.0.0'으로 설정

- Boot> status

- Boot> set ip_addr

- Bootp 실행

- Boot> btp

- bootptab 파일 만들기

- /etc/bootptab 설정

```
# vi /etc/bootptab
```

```
.default:W
:hd=/tftpboot:bf=null:W
:sm=255.255.255.0
pxa255_pro:ht=1:ha=0x0123456789ab:tc=.default:ip=192.168.1.83:
```

Target MAC 주소

Target IP 주소

- /etc/hosts 설정

- 192.168.1.82는 host의 IP

- 192.168.1.83은 target IP

127.0.0.1	localhost.localdomain	localhost
192.168.1.82	test-linuxs	
192.168.1.83	pxa255_pro	

임베디드 시스템의 소프트웨어 개발 환경 구축 (원격 접속 및 Bootp 데몬 설치)

- 임베디드 시스템(embedded system:PXA255-PRO) 제어 (원격 접속 및 Bootp 데몬 설치)
 - Bootp 데몬 설치

```
# mkdir /temp_cd/ Bootpd
# cd /temp_cd/Bootpd
# cp /mnt/cdrom/Application/bootpd
-2.4.tar.gz ./
# tar xvfz bootpd- 2.4.tar.gz
# cd bootpd-2.4
# pwd
# make clean
# make
# make install
```

- bootp 참조파일-
/etc/bootptab
/etc/hosts

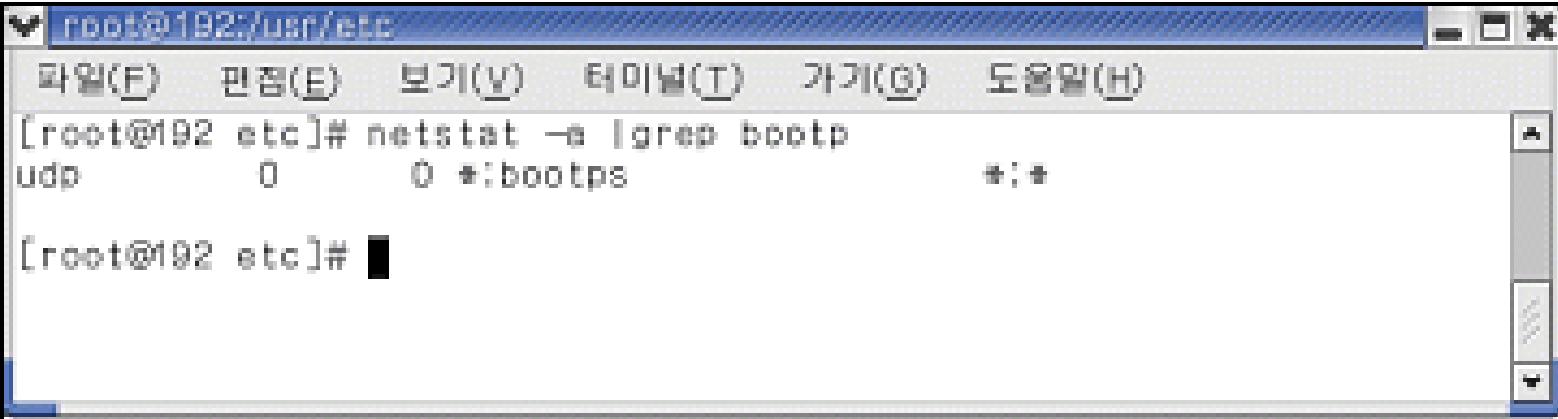
Aovlex-link



임베디드 시스템의 소프트웨어 개발 환경 구축

(원격 접속 및 Bootp 데몬 설치)

- 임베디드 시스템(embedded system:PXA255-PRO) 제어 (원격 접속 및 Bootp 실행 및 확인)
- Bootpd 실행 및 확인
 - # cd /usr/etc/
 - # ./bootpd&
 - # netstat -a | grep bootp



A screenshot of a terminal window titled "root@192:etc". The window shows the following command being run and its output:

```
[root@192 etc]# netstat -a |grep bootp
udp          0      0 *:bootps
[root@192 etc]#
```

임베디드 시스템의 소프트웨어 개발 환경 구축 (TFTP를 이용한 커널, 사용자 파일 전달)

임베디드 운영체계 및 사용자 파일 전송

Ethernet Cross Cable 이용

TFTP 또는 FTP 설정

```
# rpm -qa | grep tftp  
  
# mkdir /tftpboot  
  
# cp -rf  
    /temp_cd/Image/*  
    /tftpboot  
  
# vi /etc/xinetd.d/tftp  
  
# ntsysv  
  
# cd /etc/init.d  
  
# ./xinetd restart
```



임베디드 시스템의 소프트웨어 개발 환경 구축 (TFTP를 이용한 커널, 사용자 파일 전달)

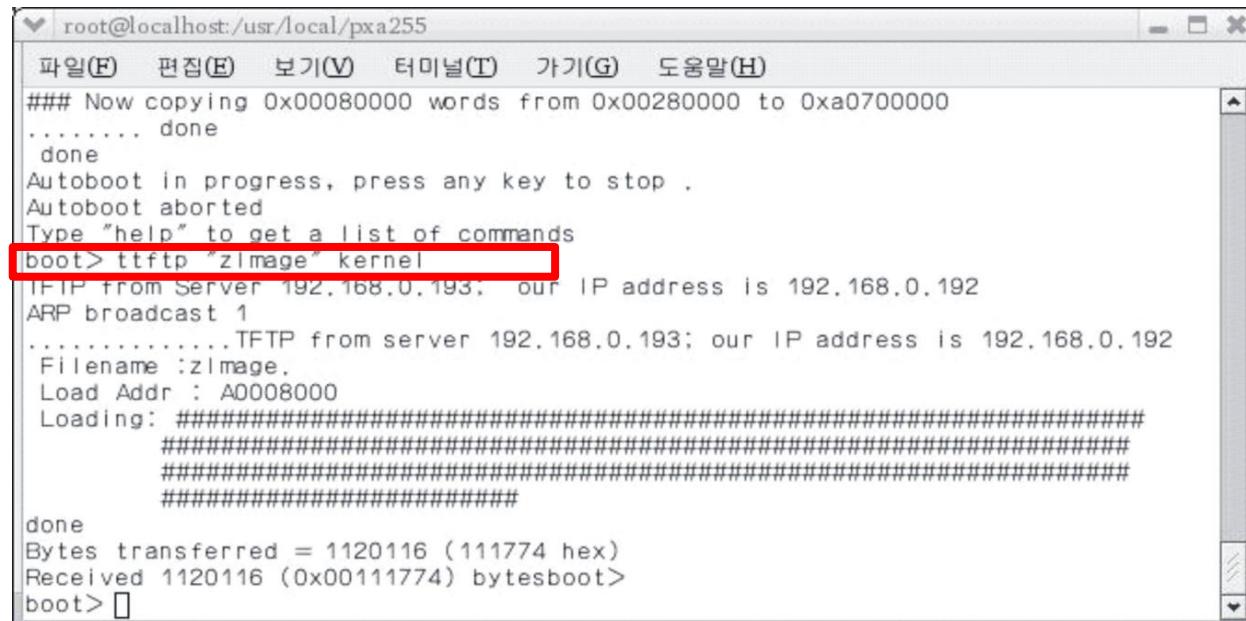
- 임베디드 시스템 보드(embedded system:PXA255-PRO)에서 커널 다운로드
 - Boot> tftp zImage kernel : SDRAM에 저장
 - Boot> flash kernel : flash memory에 저장
- 임베디드 시스템 보드(embedded system:PXA255-PRO)에서 파일 시스템 다운로드
 - Boot> tftp ramdisk.gz ramdisk
 - Boot> flash ramdisk
- 임베디드 시스템 보드(embedded system:PXA255-PRO)에서 유저 파일 시스템 다운로드
 - Boot> tftp usr_qt.jffs2 usr
 - Boot> flash usr

```
root@localhost:/usr/local/pxa255
파일(E) 편집(E) 보기(V) 터미널(T) 가기(G) 도움말(H)
### Now copying 0x00080000 words from 0x00280000 to 0xa0700000
..... done
done
Autoboot in progress, press any key to stop .
Autoboot aborted
Type "help" to get a list of commands
boot> ttftp "zImage" kernel
TFTP from Server 192.168.0.193; our IP address is 192.168.0.192
ARP broadcast 1
.....TFTP from server 192.168.0.193; our IP address is 192.168.
Filename :zImage,
Load Addr : A0008000
Loading: #####
```

임베디드 시스템의 소프트웨어 개발 환경 구축 (TFTP를 이용한 커널, 사용자 파일 전달)

- 커널 이미지 만들기
 - # tar xvfj linux-2.4.19-pxa255.tar.bz2
 - # make menuconfig
 - # make dep
 - # make clean
 - # make zImage

- ## • 커널 Download



임베디드 시스템의 소프트웨어 개발 환경 구축 (TFTP를 이용한 커널, 사용자 파일 전달)

- Booting linux kernel

The screenshot shows two terminal windows side-by-side. Both windows have a title bar 'root@localhost:/usr/local/pxa255' and a menu bar in Korean: 파일(F), 편집(E), 보기(V), 터미널(T), 가기(G), 도움말(H).
The left terminal window displays the kernel boot log:

```
000445dd (280029) words from 0xa0008000 to 0x00080000
Writing at 0x00191700
280029 words source image
280029 words written to flash
0 words skipped
0 erase operations
0 words scanned down
```

The command 'boot> boot' is entered at the bottom of this window, highlighted with a blue dashed oval.

The right terminal window shows the system booting up:

```
Starting network Starting network

Starting boa Starting Boa...
/etc/rc.d/rc3.d/S60boa: /usr/sbin/boa: No such file or directory
```

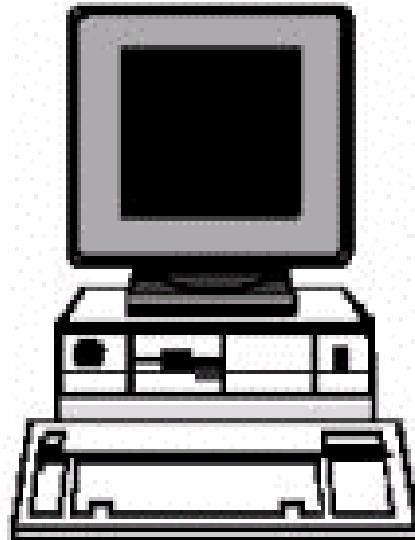

At the bottom of this window, the system prompts for a login:

```
Welcome to the Embedded Linux class!
huins login: root
[root@huins /root]$
```

임베디드 시스템의 소프트웨어 개발 환경 구축 (프로그램 설치 및 명령어)

호스트 컴퓨터

- 임베디드 S/W 컴파일
(Cross Compiler Toolchain 설치)



- 플래시 메모리에 Bootloader 탑재
(JTAG Fusing 프로그램 설치)

```
# jtag
jtag) cable parallel 0x378 PXA255
jtag) detect
jtag) flashmem 0 <파일명>
```

- 임베디드 시스템(embedded system:PXA255-PRO) 제어
(Minicom 설정)
- 주소 할당
(Bootp 데몬 설치)
- 커널, 사용자 파일 전달
(TFTP 설정)

2. Download
Serial Interface
or Ethernet
Interface

JTAG Interface
1.Fusing

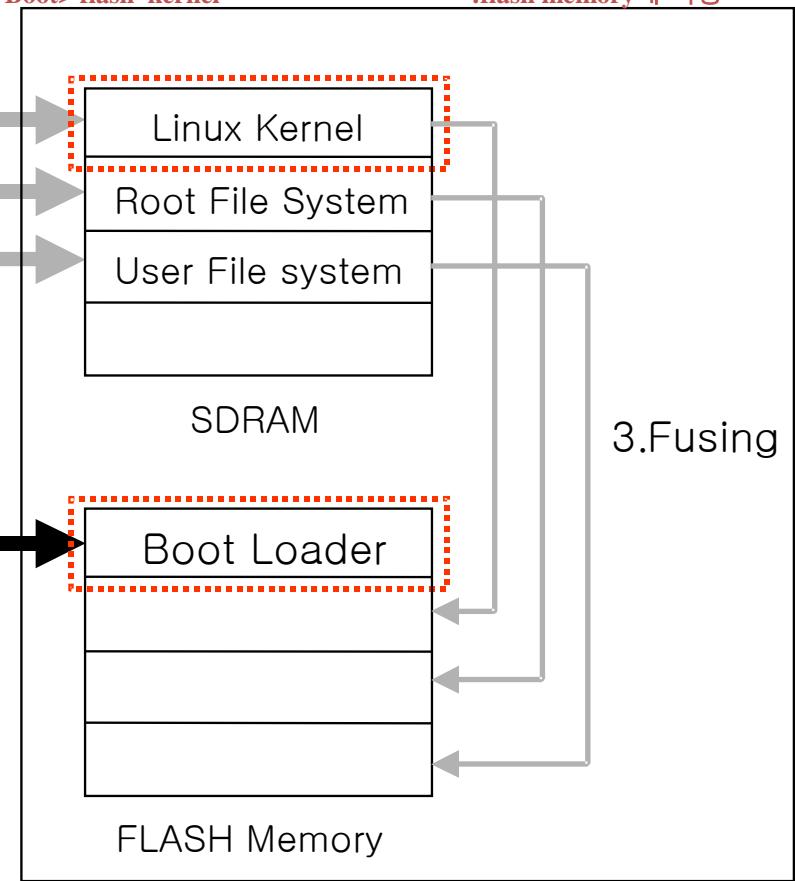
임베디드 시스템

-Boot> tftp zImage kernel

-Boot> flash kernel

:SDRAM에 저장

:flash memory에 저장



임베디드 시스템의 소프트웨어 개발 환경 구축 과정 요약

호스트 컴퓨터와
임베디드 시스템 연결

Cross Compiler Toolchain 설치

```
# cd /temp_cd/  
Toolchain  
# tar xvfj arm-cross-  
compiler.tar.bz2  
# cp -rf arm-linux  
/usr/local  
# vi  
/root/.bash_profile  
# source  
/root/.bash_profile
```

JTAG Fusing 프로그램 설치

```
# mkdir  
/temp_cd/JTAG  
# cd /temp_cd/JTAG  
# cp  
/mnt/media/cdrom/  
Application/pxa255-  
jtag.tar.gz ./  
# tar xfz pxa255-  
jtag.tar.gz  
# cd pxa255-jtag  
# cd jtag-0.4  
# ./configure  
# make  
# make install  
# cd /temp_cd  
/JTAG/jtag/bin  
# cp jtag /usr/bin  
# jtag
```

Minicom 설정

```
# minicom -s  
– minicom 접근 –  
Serial Port Setup  
Save setup as dfl Exit
```

Bootp 데몬 설치

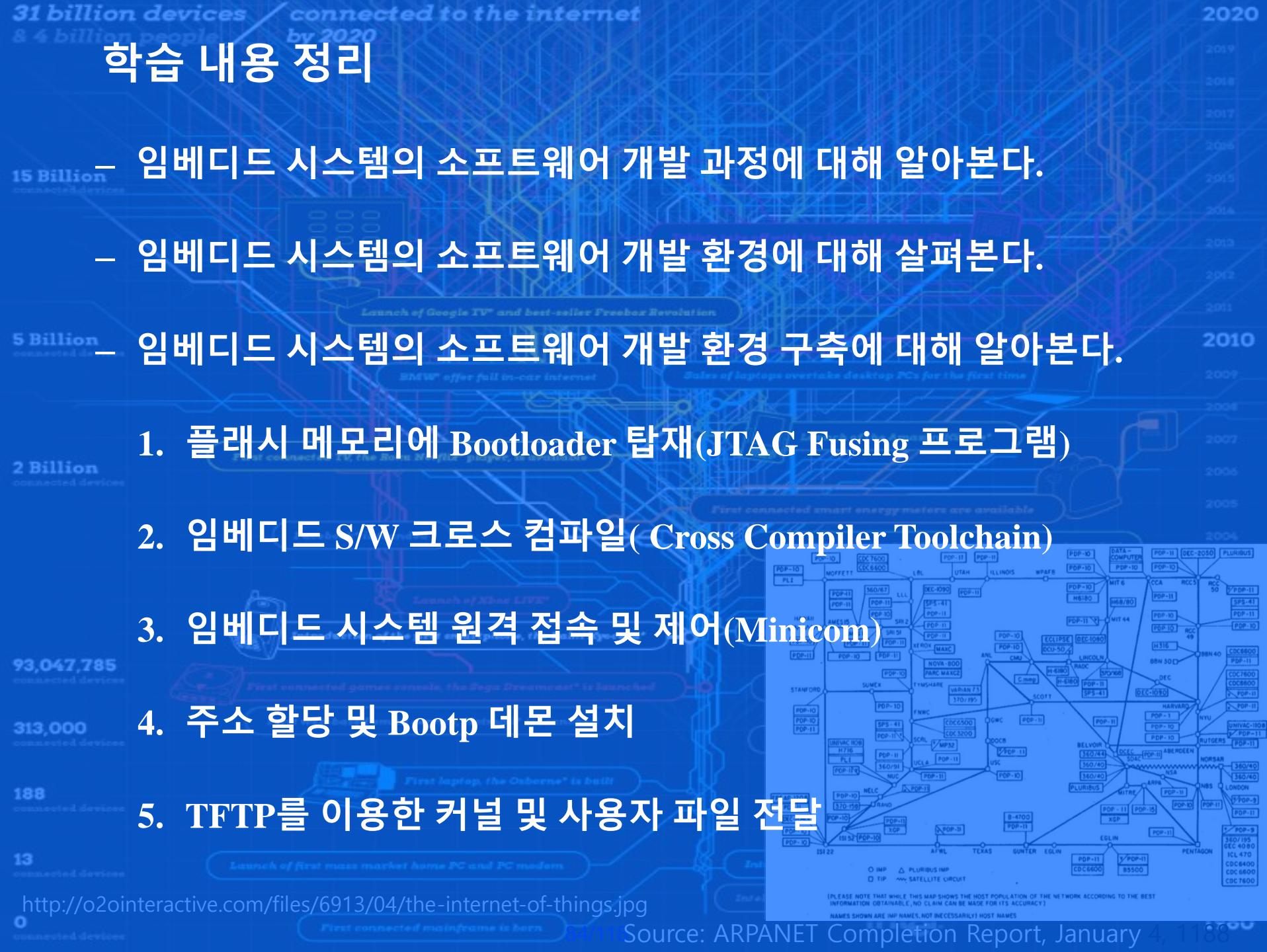
```
# mkdir /temp_cd/  
Bootpd  
# cd /temp_cd/  
Bootpd  
# cp -rf  
/mnt/cdrom/  
Application/bootpd  
-2.4.tar.gz ./  
# tar xvfz bootpd-  
2.4.tar.gz  
# cd bootpd-2.4  
# pwd  
# make clean  
# make  
# make install  
– bootp 참조파일 –  
/etc/bootptab  
/etc/hosts
```

TFTP 설정

```
# rpm -qa | grep tftp  
# mkdir /tftpboot  
# cp -rf  
/temp_cd/Image/*  
/tftpboot  
# vi /etc/xinetd.d/tftp  
# ntsysv  
# cd /etc/init.d  
# ./xinetd restart
```

A screenshot of a terminal window titled "root@localhost:~". The window shows a command-line interface with Korean menu options at the top: 파일(F), 편집(E), 보기(V), 터미널(T), 가기(G), 도움말(H). The terminal displays the following text:

```
[root@huins nfs]$. ./test_app  
test_open start!!!  
read() ^  
ioctl() ^  
[root@huins nfs]$
```



학습 내용 정리

- 임베디드 시스템의 소프트웨어 개발 과정에 대해 알아본다.
- 임베디드 시스템의 소프트웨어 개발 환경에 대해 살펴본다.
- 임베디드 시스템의 소프트웨어 개발 환경 구축에 대해 알아본다.
 1. 플래시 메모리에 Bootloader 탑재(JTAG Fusing 프로그램)
 2. 임베디드 S/W 크로스 컴파일(Cross Compiler Toolchain)
 3. 임베디드 시스템 원격 접속 및 제어(Minicom)
 4. 주소 할당 및 Bootp 데몬 설치
 5. TFTP를 이용한 커널 및 사용자 파일 전달



Thank you for listening