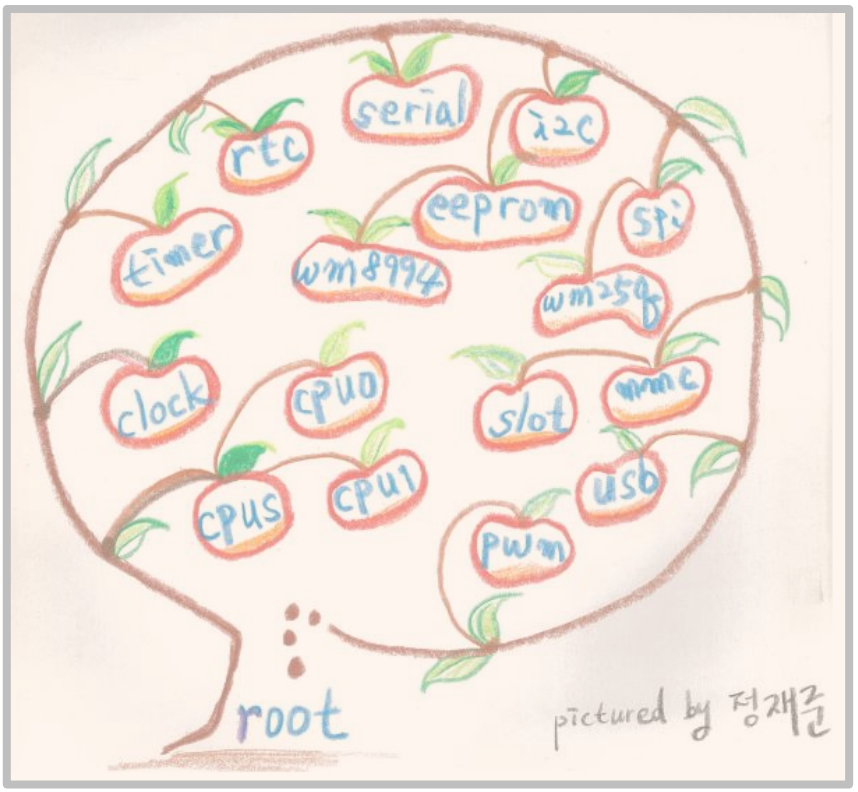
Device Tree 상세분석

in Linux Kernel 4.0

First Edition



커널연구회(www.kernel.bz)

정재준(rgbi3307@nate.com)

저작권

Device Tree 상세분석 in Linux Kernel 4.0

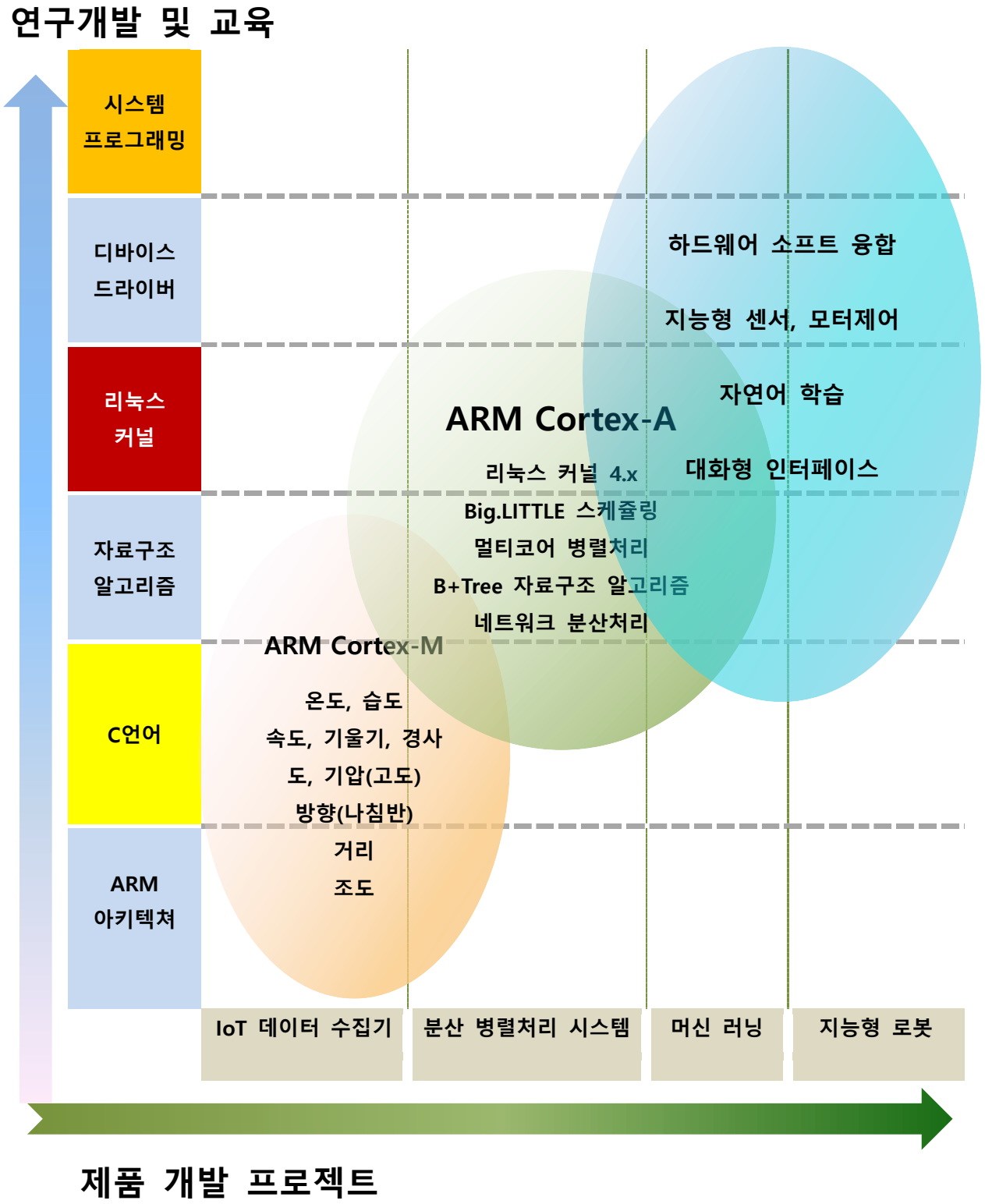
저자 정재준

Published by 커널연구회(www.kernel.bz).
서울 금천구 두산로 70 현대지식산업센터 A동 26층 2611호

발행인: 정재준
발행처: 커널연구회
출판사등록번호: 제2011-75호
출판사등록일: 2011년 09월 27일
전화및팩스: 031-594-5307
ISBN: 978-89-97750-06-1

초판발행일: 2015년 08월 07일

커널연구회 로드맵


연구개발 및 교육
시스템 프로그래밍
디바이스 드라이버
리눅스 커널
자료구조 알고리즘
C언어
ARM 아키텍쳐
하드웨어 소프트 융합
지능형 센서, 모터제어
자연어 학습
대화형 인터페이스
ARM Cortex-A
리눅스 커널 4.x
Big.LITTLE 스케쥴링
멀티코어 병렬처리
B+Tree 자료구조 알고리즘
네트워크 분산처리
ARM Cortex-M
온도, 습도
속도, 기울기, 경사도, 기압(고도)
방향(나침반)
거리
조도
IoT 데이터 수집기
분산 병렬처리 시스템
머신 러닝
지능형 로봇
제품 개발 프로젝트

커널연구회 교육과정

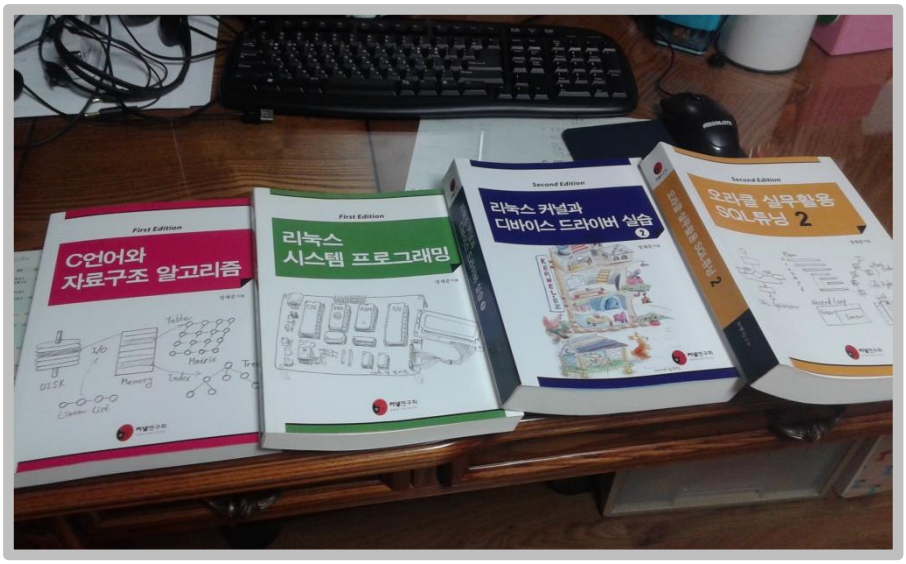
	교육과정명	일수	시간	교육비	환급 (할인)	교육일시
1	C언어와 자료구조 알고리즘	4일	32시간	88만원	일반인50% 대학생60%	매월 **첫째주**(월~목) 09:00 ~ 18:00
2	리눅스 시스템 프로그래밍	4일	32시간	88만원	일반인50% 대학생60%	매월 **둘째주**(월~목) 09:00 ~ 18:00
3	ARM 아키텍쳐, 펌웨어 실습	4일	32시간	99만원	일반인50% 대학생60%	매월 **세째주**(월~목) 09:00 ~ 18:00
4	리눅스 커널 및 드라이버 실습	4일	32시간	99만원	일반인50% 대학생60%	매월 **넷째주**(월~목) 09:00 ~ 18:00
	* 모든 교육과정에 대하여 커널연구회에서 집필한 교재를 제공합니다. *** 대학생은 모든 교육과정에 대하여 교육비 60% 할인해 드립니다.**					

♣ 좀더 자세한 내용들은 커널연구회(www.kernel.bz) 웹사이트를 참조하세요.

저자 소개

정재준 (rgbi3307@nate.com) / 커널연구회(www.kernel.bz)

저자는 학창시절 마이크로프로세서 제어 기술을 배웠으며 리눅스 커널을 연구하고 있다. 15년 이상 쌓아온 실무 경험을 바탕으로 "C언어와 자료구조 알고리즘", "리눅스 시스템 프로그래밍", "리눅스 커널과 디바이스드라이버 실습2", "오라클 실무활용 SQL튜닝2"등의 책을 집필하고, 월간임베디드월드 잡지에 다수의 글을 기고 하였다. 또한 "맞춤형 문장 자동 번역 시스템 및 이를 위한 데이터베이스 구축방법 (The System for the customized automatic sentence translation and database construction method)" 라는 내용으로 프로그래밍을 하여 특허청에 특허출원 하였다. 최근에는 서울시 버스와 지하철 교통카드 요금결재 단말기에 들어가는 리눅스 커널과 디바이스 드라이버 개발 프로젝트를 성공적으로 수행했고 여러가지 임베디드 제품을 개발했다. 저자는 스탠포드대학교의 John L. Hennessy 교수의 저서 "Computer Organization and Design" 책을 읽고 깊은 감명을 받았으며, 컴퓨터구조와 자료구조 알고리즘 효율성 연구를 통한 기술서적 집필에 노력하고 있다. 저자는 온라인 상에서 커널연구회(http://www.kernel.bz/) 웹사이트를 운영하며 연구개발, 교육, 관련기술 공유 등을 위해 노력하고 있다.



♣ 저자가 집필한 책들

(목록단락)
Device Tree 상세분석 in Linux Kernel 4.0
First Edition

문서 표준

(제목1)제#장 대제목 맑은고딕 20 진하게

본문 내용(contents) **폰트는** 맑은고딕 크기 10

(제목2)#.# 중제목 맑은고딕 16 진하게

본문 내용(contents) **폰트는** 맑은고딕 크기 10

(캡션1)맑은고딕 14 진하게

본문 내용(contents) **폰트는** 맑은고딕 크기 10

(캡션2)맑은고딕 12 진하게

본문 내용(contents) **폰트는** 맑은고딕 크기 10

(캡션3)맑은고딕 11 진하게

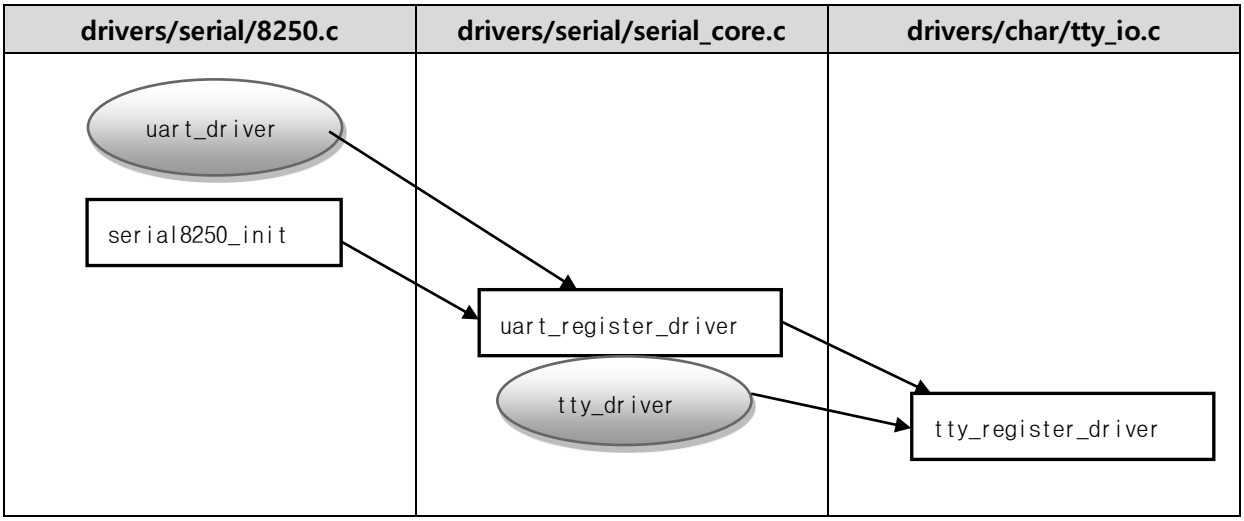
본문 내용(contents) **폰트는** 맑은고딕 크기 10

(캡션4)맑은고딕 10 진하게

본문 내용(contents) 폰트는 맑은고딕 크기 10
(강한참조) 본문 내용(contents) 폰트는 맑은고딕 크기 10
(약한참조) 본문 내용(CONTENTS) 폰트는 맑은고딕 크기 10

소스 흐름을 설명하기 위한 Flow Diagram

소스 코드 폰드는 돋움체 크기 9



위에서 타원형은 구조체를 나타내고 사각형은 함수를 의미한다. 화살표 연결은 구조체와 함수간의 데이터 흐름을 나타낸다.

소스를 상세 설명하기 위한 테이블

소스 코드 폰드는 돋움체 크기 9

함수원형	int uart_register_driver(struct uart_driver *drv)	
설명	uart core 계층에 드라이버 등록	
매개변수	drv: 드라이버 구조체	
소스파일	drivers/serial/serial_core.c	
구조체	struct uart_driver struct uart_state struct tty_driver struct tty_port	_1 _2 _3 _4

소스	int uart_register_driver(struct uart_driver *drv)
	{
	struct tty_driver *normal;
	int i, retval;
	
	BUG_ON(drv->state);
	
	//중간생략...
	
	if (retval >= 0)
	return retval;
	
	put_tty_driver(normal);
	out_kfree:
	kfree(drv->state);
	out:
	return -ENOMEM;
		
		_5
		
		_6

	}	

_번호는 소스코드를 상세하게 설명하기 위한 색인 번호.
색인 번호별로 소스 코드 설명.

함수 요약 설명 테이블

소스 코드 폰드는 돋움체 크기 9

함수명		설명
serial8250_init()		소스파일: drivers/serial/8250.c
1	uart_register_driver()	uart core 계층에 드라이버 등록
2	platform_device_alloc()	
3	platform_device_add()	
4	serial8250_register_ports()	
5	platform_driver_register()	
6	platform_device_del()	
7	platform_device_put()	
8	uart_unregister_driver()	

번호는 함수를 설명하기 위한 색인 번호.
함수 색인 번호별로 소스 코드 설명.

목차

내용

1. 작업환경 및 소스경로

♣ 필자의 주말농장에서 찍은 코스모스

1.1 개요

부트로더는 리눅스 커널을 타켓 보드의 RAM에 복사한 다음에 커널로 진입(jump)하는 역할을 한다. 커널은 CPU을 설정하고 가상 메모리를 초기화 하는 여러가지 작업들을 콘솔에 표시한다. 이러한 커널의 동작들은 장치들의 정보를 레지스터에 쓰면서 수행된다. 프로세서 코어들의 종류가 많고 접근할 수 있는 메모리 용량도 다양한 상황에서 커널은 어떻게 레지스터들의 주소를 알 수 있을까? 쉽게 직관적으로 해결할 수 있는 방법은 커널 소스안에 플렛폼 의존적(맞춤형)인 부트 루틴을 만드는 것이다. 이것은 커널 파라미터들을 사용하여 설정할 수 있다. 이러한 방식은 모든것이 고정되어 있을때는 좋은 해법이 될 수 있지만, 장치들이 가변적으로 변경된다면 커널은 실행 시간(run-time)에 변경된 장치들을 알아내야 한다.

ARM 아키텍쳐는 점점 다양화 되고 있어 리눅스 커뮤니티에서 논란이 많아지고 있다. ARM 보드마다 자신만의 외부 장치들을 제각각 장착하고 있어서, 커널 안에 특별한 환경설정 파라미터를 패치하거나 헤더 파일들이 제멋대로 생겨나고 있다. 이로 인해서 커널 소스가 난잡해지고 유지보수가 힘들어져서 커널을 분석하는데 점점 어려움을 느끼고 있는 실정이다. 또한, 커널 소스가 특정 보드상의 특정 칩에 맞게끔 컴파일되어 실행 바이너리가 생성됨으로 인해서 모든 ARM 프로세서에 호환되지 못하는 상황이 벌어지고 있다. 그러나 PC에서는 BIOS가

있어서 하드웨어 장치가 변경되면 자동적으로 검출하여 동작하도록 해주는 작업이 쉽다. ARM 프로세서는 BIOS가 없어서 리눅스 커널이 모든 역할을 다해주고 있어서 리눅스 커널을 믿는 수밖에 없다. 그래서 리눅스 커널에 도입된 해결책이 디바이스 트리(device tree)이며, Open Firmware (abbreviated OF) 혹은 Flattened Device Tree (FDT)라고 언급되기도 한다.

디바이스 트리는 강력한 규정이라기 보다 엄격한 관례라는 관점으로 접근하는 것이 바람직하다. 예를들면, 디바이스 트리안의 경로와 파라미터는 어떻게 정하는 것이 좋은지 정해진 관례를 따라가는 것이다. 정해진 API가 특정 데이터에 접근하도록 표준화된 트리 구조체를 사용하도록 한다. 주변장치들의 버스, 주소, 인터럽트, 변수들은 정해진 관례를 지켜서 디바이스 트리에 표현한다. 디바이스 트리는 커널의 일부분으로서 하드웨어 정보를 추가, 제거, 운반하는 장소 역할을 한다.

디바이스 트리를 심도있게 학습하기 전에 먼저 아래에서 작업환경과 디바이스 트리 소스경로를 확인하여 기본적인 배경 지식을 요약 정리한다.

1.2 작업환경

Device Tree는 리눅스 커널 3.1(2011-10-24)부터 소스에 적용되었다. 그런데, 필자가 이글을 집필할 때 리눅스 커널 4.0이 배포되었으므로 가장 최신 소스인 linux-4.0을 가지고 Device Tree을 설명한다. 하드웨어 아키텍쳐는 삼성 Exynos5420(Arndale Octa Board)을 선택하여 실습 했으며 리눅스 Ubuntu 12.04 배포판(64비트 버전)에서 크로스 컴파일러 "arm-eabi-"를 사용하여 소스를 빌드했다.

이곳에서는 이들을 간략히 설명하고, 좀더 자세한 내용들은 "제4장 Device Tree 실습"에서 개발환경 구축과 소스 빌드 방법들을 자세히 설명한다.

Linux Kernel Source Version: linux-4.0

리눅스 커널 4.0 특징

참조:
https://lkml.org/lkml/2015/4/12/178
http://liliputing.com/2015/04/linux-4-0-kernel-released-for-what-its-worth.html

2015년 4월 12일(일) 15:41
Ima Sheep와 리너스 토발즈는 LKML 메일링을 통해서
Linux 3.19.4 배포이후 다음버전을 Linux 3.20으로 하지 않고 Linux 4.0 으로 결정했습니다.

Linux 4.0의 특징을 요약하면,

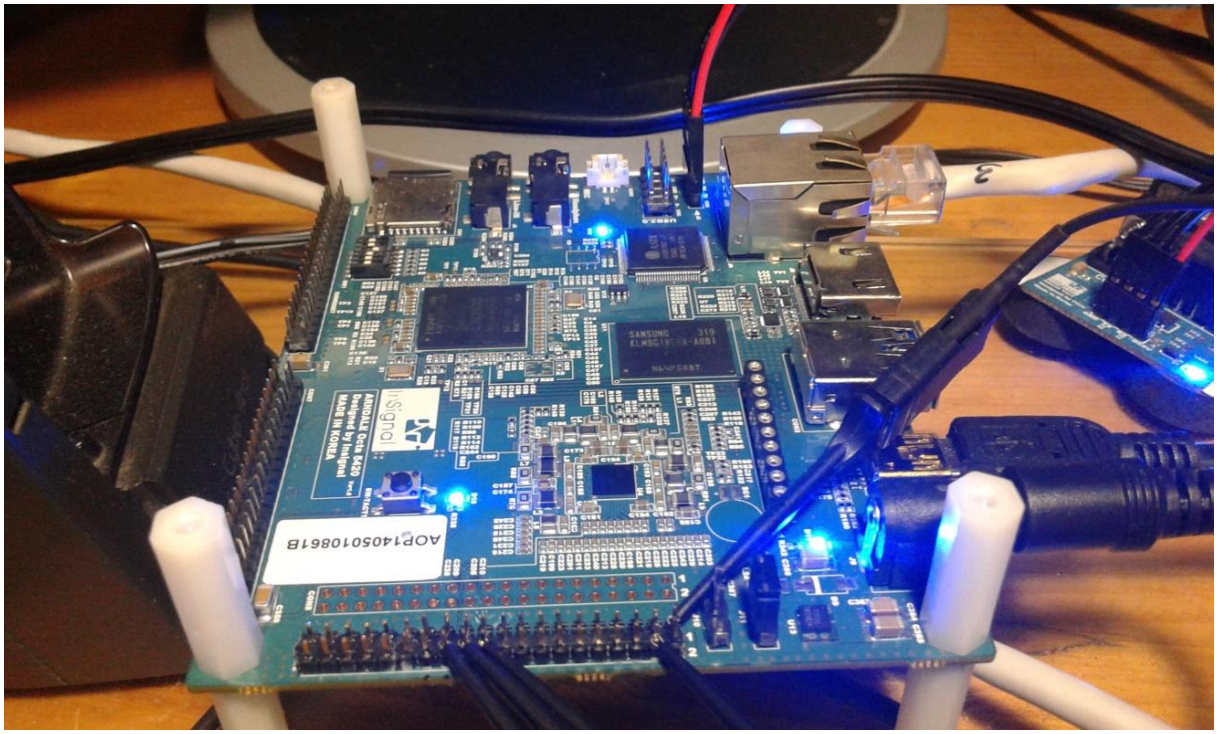
Live kernel 패치 기능으로 시스템을 리부팅하지 않아도 커널이 업데이트 됨.
AMD Radeon 드라이버가 DisplayPort를 통하여 audio을 지원함.
Intel의 Skylake 프로세서를 지원함.

Linux 4.0 버전은 새로운 기능들을 많이 추가한 것이 아니라
안정화에 초점을 맞춘 stable release라고 언급하고 있습니다.

Linux 4.1 버전에서 좀더 많은 변화가 있을 것이라고 예고하고 있습니다.

Linux 4.0 download:
https://www.kernel.org/pub/linux/kernel/v4.x/

Hardware Architecture: Samsung Exynos5420 (Arndale Octa Board)



위의 보드는 ARM Cortex-A15 1.8GHz 코어 4개와 Cortex-A7 1.2GHz 코어 4개가 합쳐진 big.LITTLE 아키텍쳐 구조로 되어 있으며, 파이러스텍(http://www.pyrustek.com/kr/)에서 판매하고 있다.

크로스 컴파일러 버전은 다음과 같이 확인한다.

Cross Compiler

```
$ arm-eabi-gcc --version
arm-eabi-gcc (GCC) 4.6.x-google 20120106 (prerelease)
Copyright (C) 2011 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

이글은 linux-4.0 커널 소스와 Documentation 경로에 있는 문서들을 참조하여 집필한다. Device Tree는 커널 소스의 Documentation/devicetree 경로에 설명되어 있다. 아울러 Device Tree의 문법적인 내용들은 아래 사이트를 참조했다.

http://devicetree.org/Device_Tree_Usage

디바이스 트리에 대해서 사전 지식이 있는 독자분들은 제2장의 내용들은 가볍고 확인하고 제3장과 4장으로 바로가서 실습해 볼 수 있다.

1.3 Device Tree 소스경로

Device Tree 소스(스크립트)는 커널의 arch/arm/boot/dts/ 경로에 있다. 소스 스크립트 파일은 아키텍쳐별로 구성되어 있다. 삼성 Exynos5에 관한 스크립트 파일은 다음과 같이 확인할 수 있다.

arch/arm/boot/dts/

```
$ cd arch/arm/boot/dts

$ ll exynos5*
-rw-rw-r-- 1 jungjj jungjj 12670 Feb 11 16:01 exynos5250-arndale.dts
-rw-rw-r-- 1 jungjj jungjj 18304 Feb 11 16:01 exynos5250.dtsi
-rw-rw-r-- 1 jungjj jungjj 16324 Feb 11 16:01 exynos5250-pinctrl.dtsi
-rw-rw-r-- 1 jungjj jungjj  8737 Feb 11 16:01 exynos5250-smdk5250.dts
-rw-rw-r-- 1 jungjj jungjj 13126 Feb 11 16:01 exynos5250-snow.dts
-rw-rw-r-- 1 jungjj jungjj  7604 Feb 11 16:01 exynos5260.dtsi
-rw-rw-r-- 1 jungjj jungjj 11172 Feb 11 16:01 exynos5260-pinctrl.dtsi
-rw-rw-r-- 1 jungjj jungjj  1873 Feb 11 16:01 exynos5260-xyref5260.dts
-rw-rw-r-- 1 jungjj jungjj  5463 Feb 11 16:01 exynos5410.dtsi
-rw-rw-r-- 1 jungjj jungjj  1400 Feb 11 16:01 exynos5410-smdk5410.dts
-rw-rw-r-- 1 jungjj jungjj  9160 Feb 11 16:01 exynos5420-arndale-octa.dts
-rw-rw-r-- 1 jungjj jungjj 21756 Feb 11 16:01 exynos5420.dtsi
-rw-rw-r-- 1 jungjj jungjj 19229 Feb 11 16:01 exynos5420-peach-pit.dts
-rw-rw-r-- 1 jungjj jungjj 14024 Feb 11 16:01 exynos5420-pinctrl.dtsi
-rw-rw-r-- 1 jungjj jungjj  9710 Feb 11 16:01 exynos5420-smdk5420.dts
-rw-rw-r-- 1 jungjj jungjj  7093 Feb 11 16:01 exynos5440.dtsi
-rw-rw-r-- 1 jungjj jungjj   871 Feb 11 16:01 exynos5440-sd5v1.dts
-rw-rw-r-- 1 jungjj jungjj  1559 Feb 11 16:01 exynos5440-ssdk5440.dts
```

```
-rw-rw-r-- 1 jungjj jungjj   684 Feb 11 16:01 exynos5800.dtsi
-rw-rw-r-- 1 jungjj jungjj 18961 Feb 11 16:01 exynos5800-peach-pi.dts
-rw-rw-r-- 1 jungjj jungjj  2796 Feb 11 16:01 exynos5.dtsi
```

.dtsi 파일은 System On Chip(SoC) 레벨에서 정의한 인클루드 파일이고, .dts 파일은 보드 레벨에서 정의한 스크립트 파일이다. dts 파일은 XML처럼 데이터의 구성을 문법적으로 기술한 것이다.

dts 스크립트를 바이너리 파일(.dtb)로 컴파일하는 Device Tree Compiler 소스는 커널의 scripts/dtc 경로에 있다. dtb 파일은 커널을 빌드할 때 아래의 Makefile 정보를 참조하여 생성된다. arch/arm/boot/dts/Makefile 에서 EXYNOS에 관련된 것만 확인해 보면 다음과 같다.

arch/arm/boot/dts/Makefile

```
dtb-$(CONFIG_ARCH_EXYNOS) += exynos4210-origen.dtb ₩
        exynos4210-smdkv310.dtb ₩
        exynos4210-trats.dtb ₩
        exynos4210-universal_c210.dtb ₩
        exynos4412-odroidu3.dtb ₩
        exynos4412-odroidx.dtb ₩
        exynos4412-odroidx2.dtb ₩
        exynos4412-origen.dtb ₩
        exynos4412-smdk4412.dtb ₩
        exynos4412-tiny4412.dtb ₩
        exynos4412-trats2.dtb ₩
        exynos5250-arndale.dtb ₩
        exynos5250-smdk5250.dtb ₩
        exynos5250-snow.dtb ₩
        exynos5260-xyref5260.dtb ₩
        exynos5410-smdk5410.dtb ₩
        exynos5420-arndale-octa.dtb ₩
        exynos5420-peach-pit.dtb ₩
        exynos5420-smdk5420.dtb ₩
        exynos5440-sd5v1.dtb ₩
        exynos5440-ssdk5440.dtb ₩
        exynos5800-peach-pi.dtb
```

dtb 파일은 부팅할 때 부트로더(u-boot)에 의해서 적재되고 커널이 파싱한다. u-boot에서 부팅 명령어인 bootz에 dtb가 있는 메모리 주소를 다음과 같이 알려준다.

bootz kernel_addr ramdisk_addr dtb_addr
bootz 0x20008000 0x21000000

커널에는 kernel_entry() 함수에 dtb_addr를 다음과 같이 전달 한다.

OLD: kernel_entry(0, mach_id, atag_addr)

NEW: kernel_entry(0, mach_id, dtb_addr)

커널 소스를 빌드할때는 menuconfig에서 아래 내용이 선택되어 있는지 확인한다.

Boot options → Flattened Device Tree support

커널을 부팅한 이후에는 /proc/device-tree 경로에서 다음과 같이 장치 설정정보들을 확인할 수 있다.

/proc/device-tree

```
# cd /proc/device-tree/
# pwd
/proc/device-tree
# ls -a
#address-cells                  mmc@12210000
#size-cells                     mmc@12220000
.                               model
..                              name
adc@12D10000                    phy@12100000
aliases                         phy@12130000
amba                            phy@12500000
audss-clock-controller@3810000  pinctrl@03860000
cci@10d20000                    pinctrl@13400000
chipid@10000000                 pinctrl@13410000
chosen                          pinctrl@14000000
clock-controller@10010000       pinctrl@14010000
codec@11000000                  power-domain@10044000
compatible                      power-domain@10044020
cpus                            power-domain@10044060
dp-controller@145B0000          power-domain@10044120
dsi@14500000                    pwm@12dd0000
fimd@14400000                   rtc@101E0000
firmware@02073000               serial@12C00000
fixed-rate-clocks               serial@12C10000
gpio_keys                       serial@12C20000
hdmi@14530000                   serial@12C30000
hdmiphy@145D0000                spi@12d20000
i2c@12C60000                    spi@12d30000
i2c@12C70000                    spi@12d40000
i2c@12C80000                    sss@10830000
i2c@12C90000                    syscon@10050000
i2c@12CA0000                    sysram@02020000
i2c@12CB0000                    system-controller@10040000
i2c@12CC0000                    tmu@10060000
i2c@12CD0000                    tmu@10064000
i2c@12E00000                    tmu@10068000
i2c@12E10000                    tmu@1006c000
i2c@12E20000                    tmu@100a0000
i2s@03830000                    usb@12000000
i2s@12D60000                    usb@12110000
```

```
i2s@12D70000                        usb@12120000
interrupt-controller@10440000       usb@12400000
interrupt-controller@10481000       video-phy@10040714
interrupt-parent                    video-phy@10040728
mct@101C0000                        video-scaler@13e00000
memory                              video-scaler@13e10000
mixer@14450000                      watchdog@101D0000
mmc@12200000
```

지금까지 Device Tree에 대한 개념을 간단히 요약하여 설명했다. 이제부터는 Device Tree 소스구조와 이것을 어떻게 빌드하여 커널에 적재후 부팅하는지 자세히 알아 보도록 하자. Device Tree에 대해서 사전 지식이 있는 독자분들은 제2장의 내용은 가볍게 확인하고 제3장부터 실습할 수 있다.

2. Device Tree 소스 분석

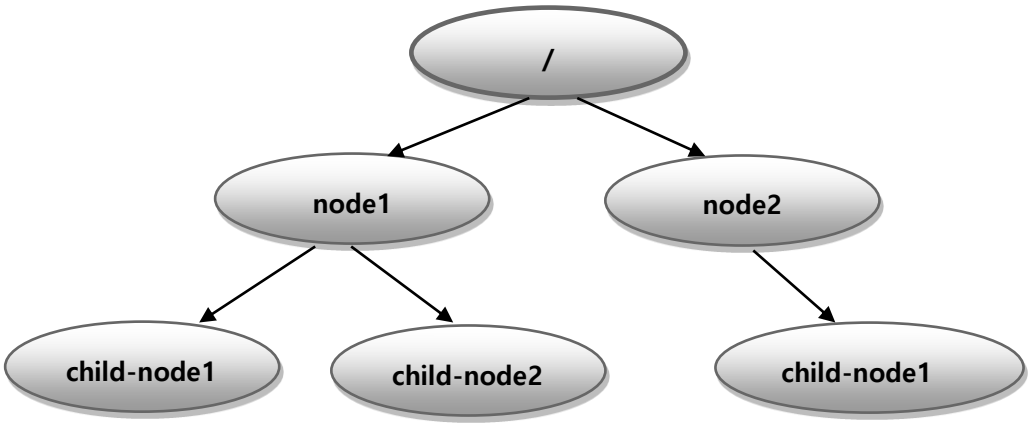
♣ 필자의 주말농장에서 찍은 장미꽃

2.1 DTS 기본 문법

아래 예제는 DTS 기본 문법을 개념적으로 표현한 것이다.

```
/ {
    node1 {
        a-string-property = "A string";
        a-string-list-property = "first string", "second string";
        a-byte-data-property = [0x01 0x23 0x34 0x56];
        child-node1 {
            first-child-property;
            second-child-property = <1>;
            a-string-property = "Hello, world";
        };
        child-node2 {
        };
    };
    node2 {
        an-empty-property;
        a-cell-property = <1 2 3 4>; /* each number (cell) is a uint32 */
        child-node1 {
        };
    };
};
```

/는 루트 노드를 의미한다. 노드는 {}로 범위를 규정한다. 루트 노드는 node1, node2라는 child 노드를 가지고, node1는 또다시 child-node1, child-node2을 가지고, node2는 child-node1를 가진다. 위의 노드를 트리구조로 표현하면 다음과 같다.



{}안에 노드 속성들을 표현하며 다음과 같은 종류들이 있다.

문자열: ""을 사용하여 다음과 같이 표현한다.

string-property = "a string";

Cells: <>을 사용하여 다음과 같이 32비트 부호없는 정수로 표현한다.

cell-property = <0xbeef 123 0xabcd1234>;

이진 데이터: []을 사용하여 다음과 같이 표현한다.

binary-property = [0x01 0x23 0x45 0x67];

혼합된 데이터: ,을 사용하여 다음과 같이 표현한다.

mixed-property = "a string", [0x01 0x23 0x45 0x67], <0x12345678>;

문자열 리스트: ,을 사용하여 다음과 같이 표현한다.

string-list = "red fish", "blue fish";

((중간생략: 책의 전체 내용은 시중 서점에서 구매 가능합니다))

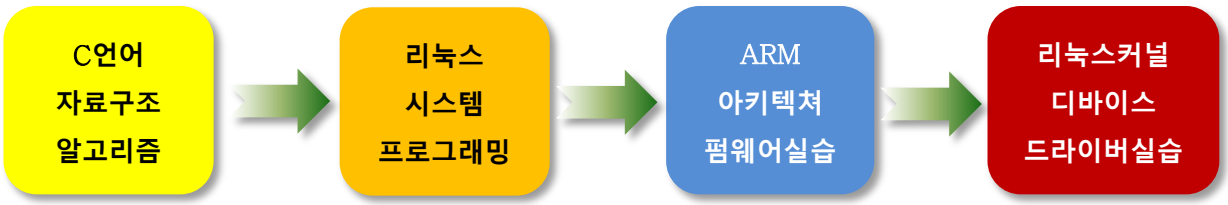
부록3. 커널연구회 교육과정

A3.1 교육 일정

	교육과정명	일수	시간	교육비	환급 (할인)	교육일시
1	C언어와 자료구조 알고리즘	4일	32시간	88만원	일반인50% 대학생60%	매월 **첫째주**(월~목) 09:00 ~ 18:00
2	리눅스 시스템 프로그래밍	4일	32시간	88만원	일반인50% 대학생60%	매월 **둘째주**(월~목) 09:00 ~ 18:00
3	ARM 아키텍쳐, 펌웨어 실습	4일	32시간	99만원	일반인50% 대학생60%	매월 **세째주**(월~목) 09:00 ~ 18:00
4	리눅스 커널 및 드라이버 실습	4일	32시간	99만원	일반인50% 대학생60%	매월 **넷째주**(월~목) 09:00 ~ 18:00
	* 모든 교육과정에 대하여 커널연구회에서 집필한 교재를 제공합니다. *** 대학생은 모든 교육과정에 대하여 교육비 60% 할인해 드립니다.**					

♣ 좀더 자세한 내용들은 커널연구회(www.kernel.bz) 웹사이트를 참조하세요.

A3.2 교육 로드맵

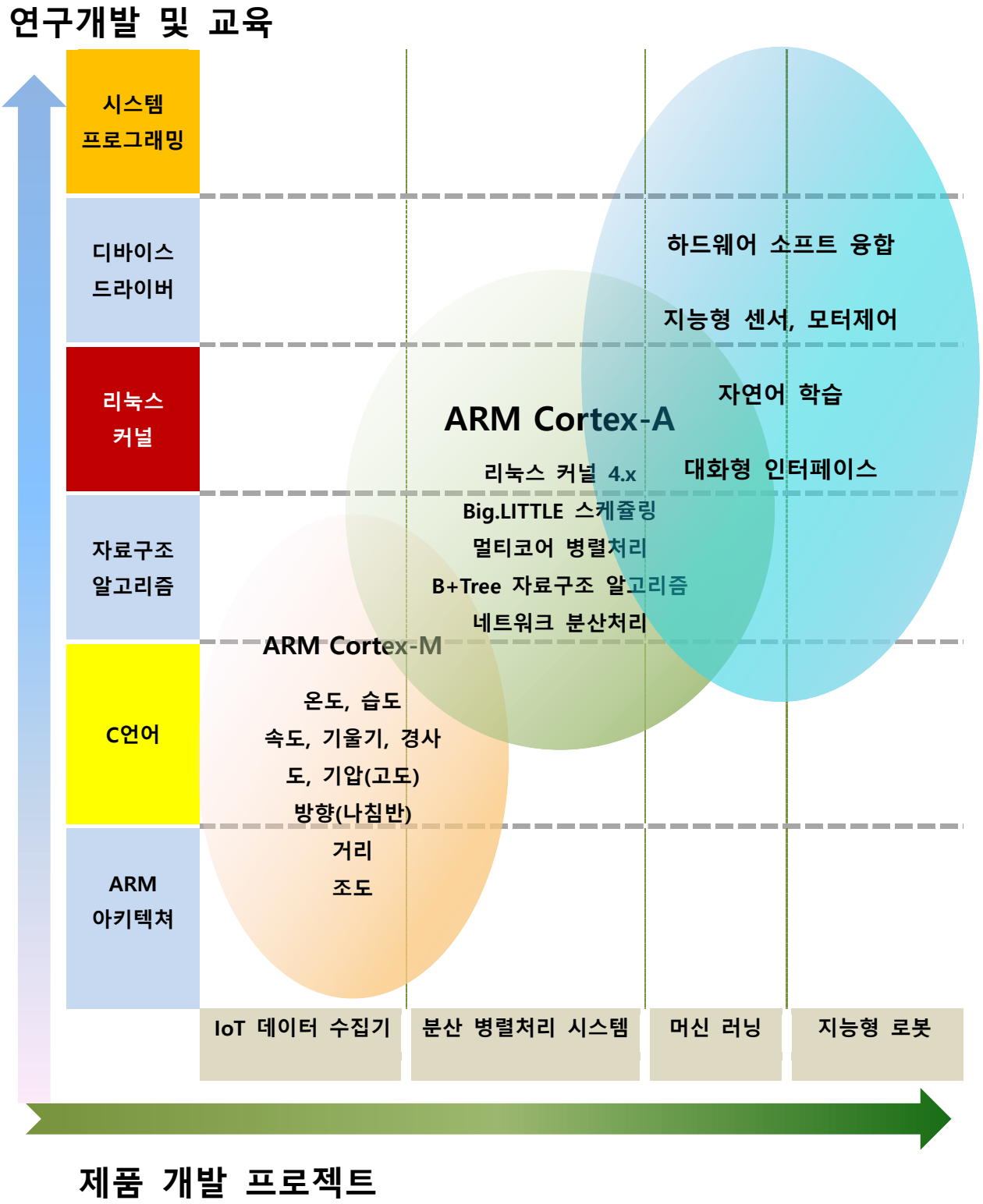


A3.3 상세 교육 내용 테이블

	기초 지식	실무 실습	심화 학습
C언어	형태, 연산자, 표현 제어흐름, 문장과 블록 조건판단, 반복문, 분기 함수, 영역(scope) 매크로, 전처리기	포인터와 주소, 함수, 배열 주소연산 포인터 배열 함수 포인터 복잡한 포인터 해석	구조체 함수, 배열, 포인터 객체 지향 코딩 모듈화, 라이브러화 대형 프로젝트 설계 코딩
자료구조 알고리즘	반복, 재귀, 포인터 정렬 알고리즘 Bubble, Insertion, Merge, Quick, Heap	Linked List 구현 Stack, Queue 실습 Hashing 실습 Binary Search Tree	Red-Black Tree 구현 B-Tree 구현
리눅스 시스템 프로그램	파일 입출력 이해 open, read, write, sync seek, select, poll, epoll mmap	Process 이해 및 실습 fork, exec 실습 zombi, daemon, wait thread , mutex 실습	address, malloc, mapping signal 실습 timers 실습 네트워크 프로그래밍
ARM 아키텍쳐	ARM Cortex-M 이해 ARM Cortex-A 이해 메모리 맵 어드레싱 이해	UART, I2C 인터페이스 I2S, SPI 인터페이스 ADC, USB 인터페이스 RTC, TFT-LCD 인터페이스	SRAM, Flash 메모리 이해 BlueTooth, WiFi 이해 Modem 무선통신 이해
리눅스 커널 이론	커널 자료구조 이해 커널 알고리즘 효율성 Process 이해 문맥 교환 이해 쓰레드 이해	프로세스 스케쥴링 이해 프로세스 스케쥴러 분석 인터럽트 이해 Top/Bottom Halves 이해	동기화 이론 및 분석 보호영역과 경쟁조건 이해 락킹(Locking) 이해 락킹(Locking) 분석 동기화 방법들 분석
리눅스 커널 디바이스 실습	리눅스 커널소스 빌드 램디스크 포팅 부팅 과정 실습 Device Tree 이해 Device Tree 실습	GPIO 드라이버 실습 시리얼(UART) 드라이버 USB 드라이버 분석 UART 드라이버 상세분석	I2C, SPI 장치 실습 USB 장치 실습 ADC 장치 실습 통신 장치 실습 (BlueTooth, WiFi)
응용 프로젝트	IoT 데이터 수집기 작성 온도, 습도, 속도, 기울기 경사, 기압(고도) 방향(나침반), 조도	분산 병렬처리 시스템 임베디드 클러스터 시스템 네트워크 분산처리 빅데이터 수집 서버	머신 러닝 자연어 학습 대화형 인터페이스 지능형 로봇

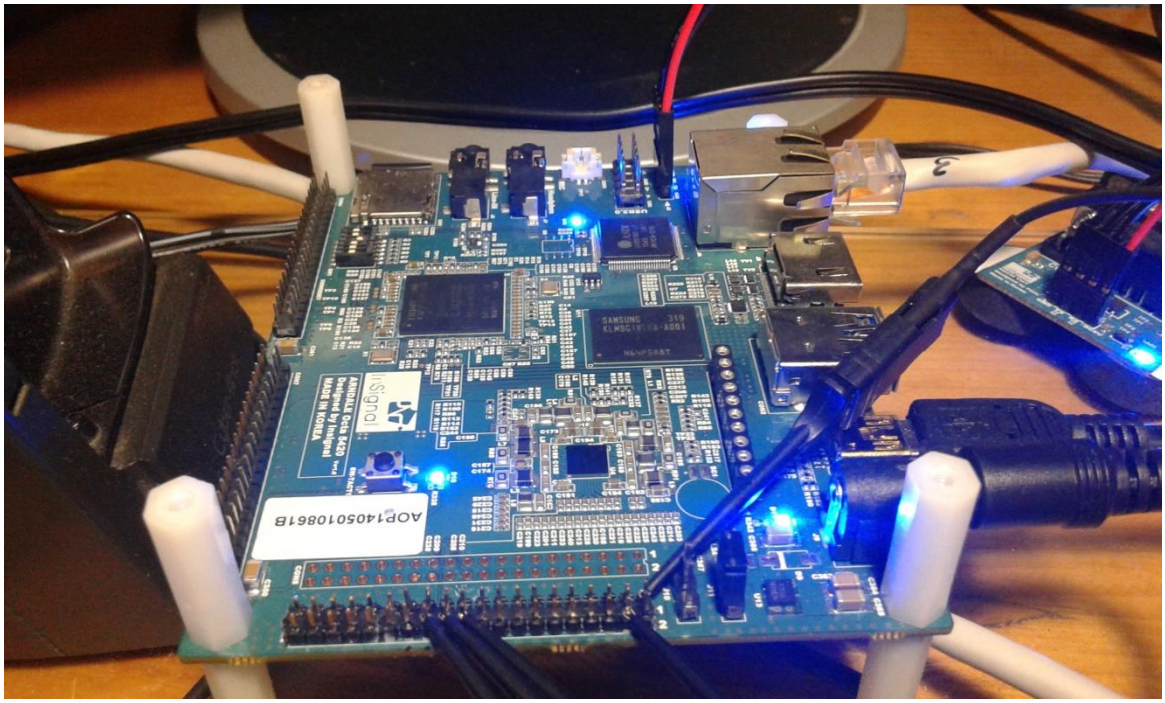
♣ 좀더 자세한 내용들은 커널연구회(www.kernel.bz) 웹사이트를 참조하세요.

A3.4 커널연구회 로드맵

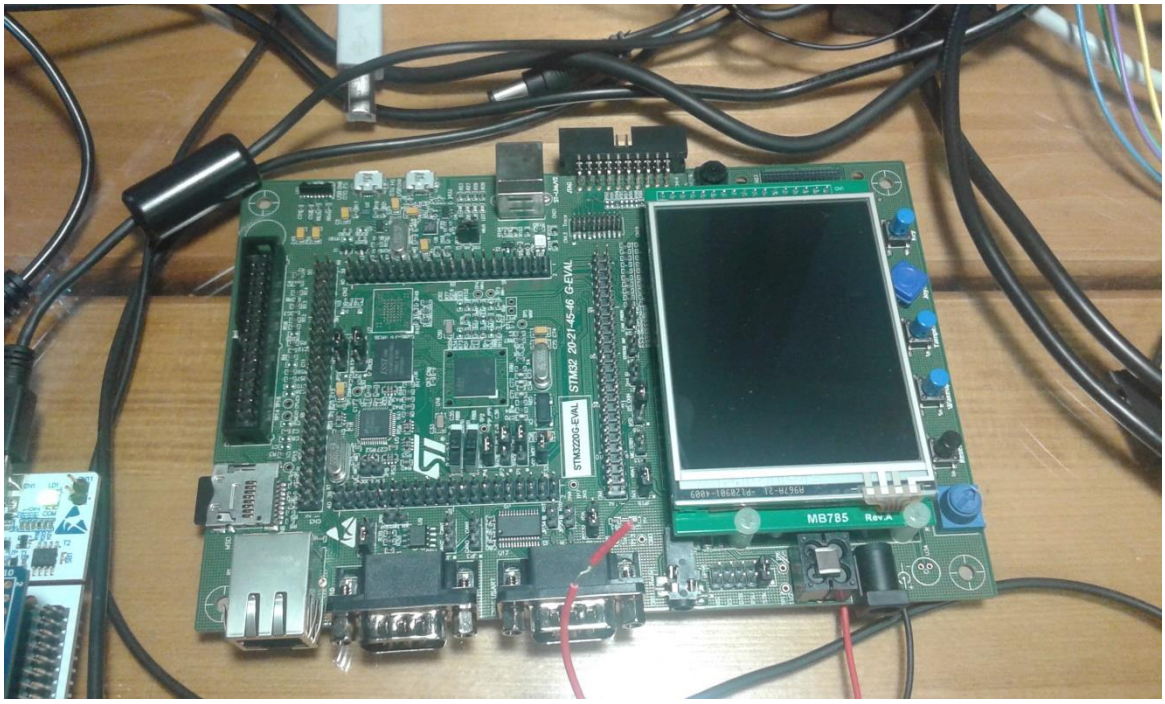


A3.5 실습용 임베디드 보드

ARM Cortex-A15, A7 (삼성 Exynos5420)



ARM Cortex-M3 (STM32F2)



A3.6 커널연구회 교육학원 위치(약도)

- **주소:** 서울시 금천구 두산로 70 (독산동 291-1)
  현대지식산업센터 A동 26층 2611호

- **교통: 지하철 1호선 독산역 1번출구**
  (도보 11분, 이동거리 708미터)

♣ 기타 내용들은 커널연구회(www.kernel.bz) 웹사이트를 참조하시기 바랍니다.
♣ 감사합니다.