

임베디드시스템설계

라즈베리파이

Byungjin Ko

Department of Intelligent Robotics

임베디드 소프트웨어와 하드웨어

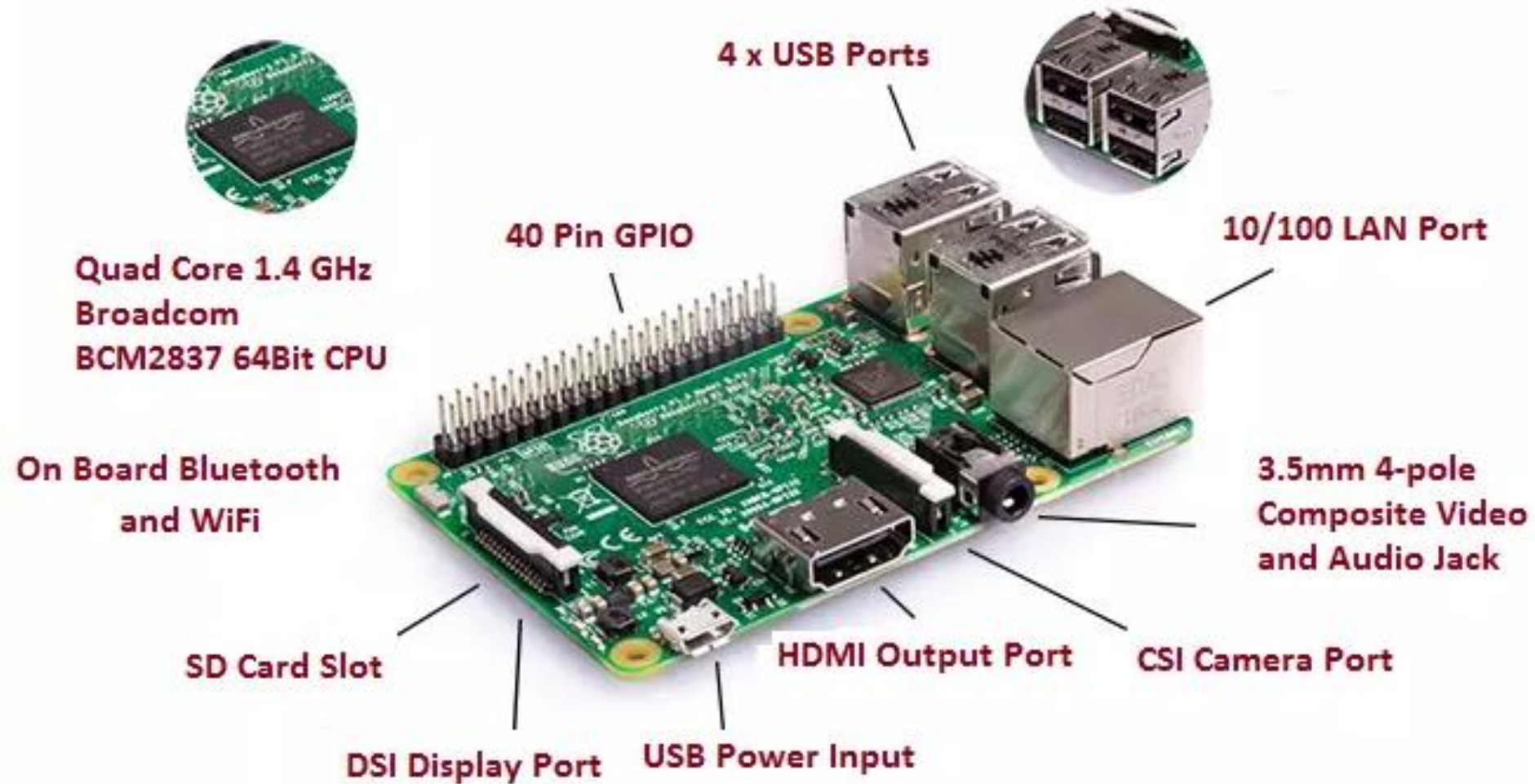
● 임베디드 소프트웨어

- 특정 하드웨어의 임베디드 즉, 하드웨어에 내장되어 목적에 적합한 특정 기능을 수행하는 소프트웨어

● 임베디드 하드웨어 구성요소

- 목적에 따라서 프로세스, 메모리, 입출력 장치, 주변 장치 등이 결정됨

라즈베리파이3 하드웨어 구성도



라즈베리파이의 프로세서

- 브로드컴의 BCM2837 SoC 탑재
- SoC (System on Chip)
 - 컴퓨터를 비롯한 여러 전자시스템에 주요 구성요소가 하나의 칩에 결합되어있는 집적회로 IC
- 라즈베리파이의 ARM SoC의 구성요소
 - 중앙처리장치(CPU)
 - 그래픽 처리장치(GPU)
 - 메모리
 - 타이머
 - GPIO
 - USB
 - I2C
 - SPI
 - UART
 - 기타 주변장치 입출력 회로

ARM 프로세서

- ARM의 최신 프로세서들은 Cortex라는 이름으로 명명
- Cortex의 3가지 종류
 - Cortex A(Application): 스마트폰 등 운영체제가 필요한 비교적 큰 규모의 장치에 사용
 - Cortex R(Realtime): 자동차, 산업용 제어장치 등 실시간 임베디드 시스템 서비스에 사용
 - Cortex M(Microcontroller): 소형, 저가, 저전력에 필요한 애플리케이션에 사용
- 라즈베리파이3는 Cortex A
 - ARMv8 Architecture, 64bit, 애플리케이션 프로세서(AP) A53버전

라즈베리파이3 기타 하드웨어

메모리

- 라즈베리파이 3은 메모리 시스템이 메모리 1GB, LPDDR2 SDRAM
- 라즈베리파이 3은 메모리를 SoC 칩 위에 적층 형태로 부착되어 있음

GPIO (General Purpose Input/Output)

- 입력이나 출력을 포함한 동작이 런타임 시에 사용자에게 의해 제어 가능한 디지털 신호 핀
- 외부장치들을 연결하여 제어 할 수 있는 핀

UART(Universal Asynchronous Receiver/Transmitter)

- 병렬 데이터 형태를 직렬 형태로 전환하여 데이터를 전송하는 하드웨어 (송수신기)

이더넷(Ethernet)

- 네트워크에 연결된 각 기기들이 고유의 MAC 주소를 가지고 이 주소를 이용해서 상호 간의 데이터를 주고받을 수 있도록 만들어진 것

USB(Universal Serial Bus)

- 직렬포트, 병렬 포트 등 다양한 기종의 연결 방식을 대체하기 위해서 만들어 짐

HDMI(High Definition Multimedia Interface)

- 영상과 음성을 통합한 단자 규격

부팅, 펌웨어, 부트로더

- 부팅에 필요한 프로그램, 운영체제, 데이터의 경우 micro SD card 에 저장하며 이를 이용해 부팅을 진행
 - 부팅에는 펌웨어, 부트로더가 작동함
- 펌웨어
 - 일반적으로 ROM에 저장된 하드웨어를 제어하는 소프트웨어 프로그램을 의미함
 - 소프트웨어와 하드웨어의 특성을 모두 가지고 있음
 - 하드웨어 제어를 위해 필요한 별도의 논리회로를 소프트웨어(펌웨어) 대체함으로써 간단하게 구현 가능
- 부트로더
 - 펌웨어의 일종으로 부팅되는 과정에서 실행
 - 커널을 시동시키기 전에 하드웨어를 준비하는 역할
 - 프로세서의 종류에 따라 부트로더가 달라지지만, 대중적으로 사용되어지는 오픈소스 부트로더가 있음

임베디드 시스템의 구성과 펌웨어

Case 1

- 일반적인 PC 형태
- 하드웨어 제어를 운영체제가 담당
- 어플리케이션은 운영체제로 부터 제어를 받음
- 비교적 규모가 큰 임베디드 시스템
- 부트로더가 펌웨어의 역할
- 예) 라즈베리파이

Application software

Boot loader

OS

Hardware

Case 2

- 운영체제 없음
- 어플리케이션 소프트웨어가 하드웨어를 직접 제어
- 단순한 기능, 비교적 규모가 작은 임베디드 시스템 구성
- 부트로더 없음, 펌웨어 있음
- 예) 아두이노

Application software

Hardware

부트로더의 기능

● 부트로더(bootloader)

- 운영 체제가 시동되기 이전에 미리 실행되면서 커널이 올바르게 시동되기 위해 필요한 모든 관련 작업을 마무리하는 역할
- 운영체제가 시스템에 로드되어 잘 동작시동시키기 위한 목적을 가진 프로그램

● 부트로더 기능

- 하드웨어 초기화
- 운영체제 동작 환경 구성
- 운영체제를 메모리로 로드

부트로더가 지원해야 하는 상세 기능

- 하드웨어 초기화
 - 메모리 설정, CPU clock 설정, GPIO 설정, Serial 설정, 네트워크의 MAC 주소 설정, 이더넷 포트 설정 등
- 메모리 복사
 - 커널 이미지, RAM disk image, 부트로더 등을 flash ROM에서 SDRAM으로 복사하는 기능
- 커널 부팅
 - 커널 이미지를 메모리로 복사해 압축을 풀고 실행시키면 부트로더의 할 일은 다하게 되며, 제어권은 커널이 가짐
- 명령어 지원
 - 사용자가 부트로더의 기능을 사용할 수 있도록 명령어 지원
- 디버깅 모드
 - 부트로더의 동작 상태를 모니터링하면서 에러 부분을 찾아낼 수 있으며 전문적인 디버깅을 위해서는 장비나 소프트웨어가 필요

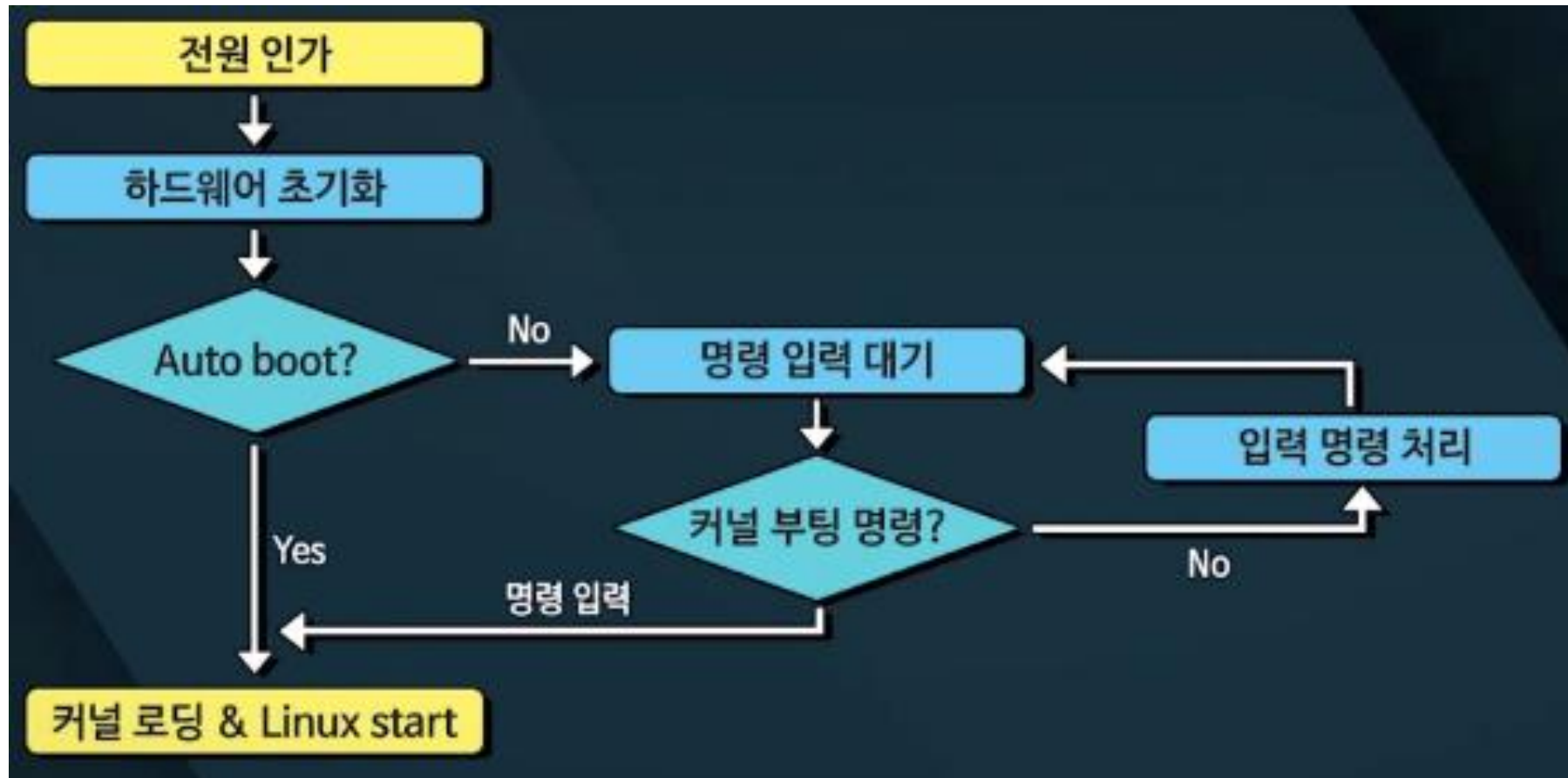
부트로더의 특징

- 시스템 하드웨어 의존성이 매우 강함
- 프로세서의 구조, 특징, 사용법 이해 필요
- 프로세서에 따른 명령어 사용법 이해 필요
 - 부트로더의 시작 부분이 어셈블리어로 작성되어있음
- 플래시메모리의 부트섹터에 저장되어 유지되어야 하므로 크기가 작아야 함
- 오픈 소스 소프트웨어가 많아 활용이 용이

부트로더 흐름

운영체제 부트 과정

- 전원이 인가되면 부팅에 사용되는 커널을 메모리에 올리고 실행



오픈소스 부트로더

- 모든 하드웨어를 지원하는 부트로더는 없으므로 부트로더 개발 시 오픈소스에서 지원해주는지 확인 필요
- 대표적인 오픈소스 부트로더
 - OpenBLT
 - GRUB (Grand Unified Bootloader)
 - PC 및 서버에서 사용
 - Arm 기반 U-Boot

U-Boot의 구조

U-Boot의 디렉토리 구조는 리눅스의 디렉토리 구조와 흡사

- `u-boot` : 최상위 디렉토리로 `Makefile`, `boards.cfg` 등의 주요 파일을 가지고 있으며, 모든 u-boot의 빌드 동작을 실행하는 디렉토리
- `arch` : ARM 등 프로세서 아키텍처별로 서로 다른 디렉토리를 구성하고 있으며, 각 프로세서 아키텍처에 따른 소스를 구성하는 디렉토리
- `board` : 각 제조사별 보드 관련 소스를 구현하는 디렉토리
- `common` : u-boot에서 공통적으로 사용되는 소스를 구현하는 디렉터리로 아키텍처 독립적인 코드, 환경 변수 등 일반적인 소스가 구현되어 있음
- `drivers` : 디바이스 제어를 위한 드라이버 프로그램이 있음 (gpio, i2c, pci, serial, sound, usb와 같은 외부 장치의 드라이버)
- `fs` : fat, yaffs2와 같이 u-boot에서 지원하는 파일 시스템 관련 코드
- `lib` : 모든 아키텍처와 관련된 라이브러리

U-boot의 주요 디렉토리와 파일

- 오픈소스 부트로더에 기능 수정이나 추가하는 방법은?
 - 기능에 따라 수정 해야하는 코드 부분, 추가 부분이 다름
- 수정을 위한 하드웨어 동작 관련의 디렉토리와 파일

디렉토리	파일	내용
u-boot top	Makefile	• U-boot 빌드에 사용하는 마스터 Makefile
	Board.cfg	• 지원하는 보드에 대한 옵션 지정 • 보드별 타겟 이름, 아키텍처, CPU정보, 보드이름, SoC이름
arch	Arm config.mk	• ARM 아키텍처에 대한 크로스컴파일러 및 빌드 옵션 지정 • 컴파일러는 CROSS_COMPILE 환경 변수에 지정
	cpu	• ARM 프로세서 디렉토리를 구현 • Armv7, armv8 등 프로세서 및 아키텍처 이름으로 구성
	include	• ARM프로세서 및 타겟 SoC 관련 헤더를 정의
	lib	• ARM 프로세서에서 일반적으로 사용되는 기능을 라이브러리로 구현한 디렉토리
board	제조사 명	• 제조사 보드 별로 특이한 사항들을 여기에 정의

U-boot의 주요 디렉토리와 파일

디렉토리	파일	내용
drivers	block	• 블록 디바이스를 구현
	mmc	• MMC 드라이버를 구현
	mtd	• NAND 등 메모리 장치를 블록디바이스로 사용하기 위한 MTD 드라이버 구현
	serial	• 시리얼 디바이스 드라이버가 구현
	usb	• USB 장치 드라이버 구현
	net	• 각종 네트워크 장치 드라이버 구현
include	Config 타겟.h	<ul style="list-style-type: none">• 타겟 보드가 동작하기 위한, 각종 하드웨어 및 소프트웨어 설정 값을 가짐• U-boot를 포팅하는데 있어 매우 중요한 정보를 가짐• Exynos 라든가 bcm으로 모델별로 정의

U-boot의 주요 디렉토리와 파일

Arch/Arm 디렉토리

디렉토리	파일	내용
Config.mk		크로스컴파일러 지정 / CROSS_COMPILE 환경변수
cpu	Armv7 Makefile	ARM arch v7 빌드용
	config.mk	ARM arch v7 빌드 옵션 지정
	u-boot.lds	U-boot 메모리 배치 정보 지정
	cpu.c	CPU 내부 제어, 캐시 제어 등
	start.S	U-boot의 시작 위치로 예외 처리 벡터를 비롯한 low-level 동작 구현
	보드명 clock.c	클록 제어
	gpio.c	GPIO 제어
	nand.c	NAND 플래시 제어
include	asm arch-타겟	<ul style="list-style-type: none">• Clock.h 클록 제어 헤더 정의• Cpu.h CPU 제어 헤더 정의• Gpio.h GPIO 제어 헤더 정의• Uart.h UART 제어 헤더 정의

U-boot의 주요 디렉토리와 파일

- 보드 디렉토리/제조사명 디렉토리/보드명 디렉토리

Makefile	• 보드 디렉토리 빌드
config.mk	• U-boot가 탑재되어 실행될 메모리 위치 지정 • CONFIG_SYS_TEXT_BASE 변수에 지정
u-boot.lds	• 타겟 보드용 u-boot 메모리 배치 정보 지정
lowlevel_init.S	• U-boot를 동작하기 위한 저수준의 코드를 구성 • CPU 초기화를 비롯하여 클럭 및 메모리 초기화 루팅을 호출하고, u-boot를 주메모리로 로딩하는 역할을 모두 담당
clock_init.S	• 타겟 보드에 적합하도록 SoC의 클럭을 초기화하는 루틴
mem_init.S	• 타겟 보드에 사용되는 메모리 장치의 초기화
보드명.c	• 보드 초기화 및 리눅스에서 사용되는 아키텍처 번호와 부트 옵션 위치 지정
setup.h	• 타겟 초기화에 필요한 헤더파일 정의

U-boot 기본환경 설정 파일

- u-boot는 라즈베리파이 부트로더 기본 환경 설정을 defconfig 파일로 제공하고 있음
- defconfig 파일을 이용해 원하는 설정을 변경할 수 있음



```
1 CONFIG_ARM=y
2 CONFIG_ARCH_CPU_INIT=y
3 CONFIG_ARCH_BCM283X=y
4 CONFIG_SYS_TEXT_BASE=0x00080000
5 CONFIG_TARGET_RPI_3=y
6 CONFIG_NR_DRAM_BANKS=1
7 CONFIG_ENV_SIZE=0x4000
8 CONFIG_DEFAULT_DEVICE_TREE="bcm2837-rpi-3-b"
9 CONFIG_SYS_PROMPT="U-Boot> "
10 CONFIG_SYS_LOAD_ADDR=0x1000000
11 CONFIG_DISTRO_DEFAULTS=y
12 CONFIG_HAS_CUSTOM_SYS_INIT_SP_ADDR=y
13 CONFIG_CUSTOM_SYS_INIT_SP_ADDR=0x7ffe40
14 CONFIG_OF_BOARD_SETUP=y
15 CONFIG_USE_PREBOOT=y
16 # CONFIG_DISPLAY_CPUINFO is not set
17 # CONFIG_DISPLAY_BOARDINFO is not set
18 CONFIG_MISC_INIT_R=y
19 CONFIG_FDT_SIMPLEFB=y
20 CONFIG_SYS_PBSIZE=1049
21 CONFIG_CMD_GPIO=y
22 CONFIG_CMD_MMC=y
23 CONFIG_CMD_USB=y
24 CONFIG_CMD_FS_UUID=y
25 CONFIG_OF_EMBED=y
26 CONFIG_ENV_FAT_DEVICE_AND_PART="0:1"
27 CONFIG_SYS_RELOC_GD_ENV_ADDR=y
28 CONFIG_ENV_VARS_UBOOT_RUNTIME_CONFIG=y
29 CONFIG_TFTP_TSIZE=y
30 CONFIG_BCM2835_GPIO=y
31 CONFIG_MMC_SDHCI=y
32 CONFIG_MMC_SDHCI_BCM2835=y
33 CONFIG_PHYLIB=y
34 CONFIG_PINCTRL=y
35 # CONFIG_PINCTRL_GENERIC is not set
36 # CONFIG_REQUIRE_SERIAL_CONSOLE is not set
37 CONFIG_SYSINFO=y
```

U-boot 명령어

- Cmd디렉토리에 명령어 정의 코드가 위치함
- 새로운 명령어 cmd디렉토리에 추가 가능

명령어	설명
bdinfo	board에 대한 정보를 화면에 출력 한다.
coninfo	console에 대한 device 정보를 화면에 출력 한다.
flinfo	Flash Memory에 대한 정보를 화면에 출력 한다.
iminfo	Image Header 정보를 화면에 출력 한다.
base	memory의 base address를 설정 하거나 화면에 출력 한다.
crc32	crc32를 계산 하거나 계산한 결과를 특정 메모리에 저장 한다.
cmp	memory의 내용을 비교 한다. (byte, word, long word)
cp	memory의 내용을 복사 한다. (byte, word, long word)
md	memory의 내용을 화면에 출력 한다. (byte, word, long word)
mm	memory의 내용을 변경 한다. (byte, word, long word)

U-boot 명령어

mtest	RAM read/write 테스트를 실행 한다.
mw	memory의 내용을 특정 값으로 채워 넣는다. (byte, word, long word)
nm	memory의 동일한 번지에 값만 바꿔서 넣는다. (byte, word, long word)
loop	memory의 특정 영역을 반복하여 실행 한다. 무한 loop 이기 때문에 실행 후 reset을 하여야 loop가 종료 된다. (byte, word, long word)
erase	flash의 내용을 지운다. 영역은 주소, 길이, sector, bank, flash 전체 등으로 줄 수 있다.
protect	flash의 write protect를 enable/disable 할 수 있다. 이것 역시 영역을 주소, 길이, sector, bank, flash 전체 등으로 줄 수 있다.
mtddpart	MTD partition table을 보거나 수정 하기 위해 사용 한다.
source	특정 주소에 있는 script를 실행 한다.
bootm	특정 memory의 주소에 있는 image로 부팅을 한다. image, ramdisk, kernel을 설정을 위한 sub command들이 있음.
go	특정 memory의 주소에 있는 application을 실행 한다.

U-boot 명령어

bootp	BOOTP/TFTP를 사용하여 Boot Image를 특정 메모리 주소에 다운로드 한다.
dhcp	DHCP/TFTP를 사용하여 Boot Image를 특정 메모리 주소에 다운로드 한다.
loadb	serial line(kermit mode)로 binary file을 특정 주소에 로드 한다.
loads	serial line을 사용하여 S-Record file을 특정 주소에 로드 한다.
rarp	RARP/TFTP를 사용하여 Boot Image를 특정 메모리 주소에 다운로드 한다.
tftpboot	TFTP를 사용하여 Boot Image를 특정 메모리 주소에 다운로드 한다.
printenv	환경변수를 화면에 출력 한다.
setenv	환경변수 설정
saveenv	환경변수를 flash에 저장한다.
run	환경변수의 값의 내용을 실행한다.
boot	bootcmd의 값을 사용하여 시스템 부팅을 시작한다. (run bootcmd)

Raspbian 이란?

- 라즈베리파이 전용 리눅스 운영체제
- 라즈베리파이의 부트로더와 커널 포함
- 마이크로 SD 카드에 라즈비안 이미지를 넣은 후 이를 이용해 라즈베리파이 부팅

hardware component
database인 dtb(device tree
block) 파일

부트로더 파일

커널 이미지 파일

이름	수정된 날짜	유형	크기
overlays	2018-03-13 오후 9:53	파일 폴더	
bcm2708-rpi-0-w.dtb	2018-03-13 오후 8:53	DTB 파일	16KB
bcm2708-rpi-b.dtb	2018-03-13 오후 8:53	DTB 파일	16KB
bcm2708-rpi-b-plus.dtb	2018-03-13 오후 8:53	DTB 파일	16KB
bcm2708-rpi-cm.dtb	2018-03-13 오후 8:53	DTB 파일	15KB
bcm2709-rpi-2-b.dtb	2018-03-13 오후 8:53	DTB 파일	17KB
bcm2710-rpi-3-b.dtb	2018-03-13 오후 8:53	DTB 파일	18KB
bcm2710-rpi-3-b-plus.dtb	2018-03-13 오후 8:52	DTB 파일	18KB
bcm2710-rpi-cm3.dtb	2018-03-13 오후 8:53	DTB 파일	17KB
bootcode	2018-03-13 오후 8:53	ALZip BIN File	51KB
cmdline		텍스트 문서	1KB
config	2018-05-14 오전 9:19	텍스트 문서	2KB
config.txt.save	2018-05-14 오전 9:17	SAVE 파일	2KB
COPYING.linux	2018-03-09 오후 6:28	LINUX 파일	19KB
fixup	2018-03-13 오후 8:53	DAT 파일	7KB
fixup_cd	2018-03-13 오후 8:53	DAT 파일	3KB
fixup_db	2018-03-13 오후 8:53	DAT 파일	10KB
fixup_x	2018-03-13 오후 8:53	DAT 파일	10KB
issue	2018-03-13 오후 11:16	텍스트 문서	1KB
kernel	2018-03-13 오후 8:53	ALZip IMG File	4,316KB
kernel7	2018-03-13 오후 8:53	ALZip IMG File	4,510KB
LICENCE.broadcom	2018-03-09 오후 6:28	BROADCOM 파일	2KB
LICENSE.oracle	2018-03-13 오후 11:16	ORACLE 파일	19KB
start.elf	2018-03-13 오후 8:53	ELF 파일	2,758KB
start_cd.elf	2018-03-13 오후 8:53	ELF 파일	656KB
start_db.elf	2018-03-13 오후 8:53	ELF 파일	4,850KB
start_x.elf	2018-03-13 오후 8:53	ELF 파일	3,819KB

환경설정 텍스트 파일

실습1

U-boot 컴파일

U-boot 소스 다운로드

- git clone 명령어를 이용해 U-Boot 소스 다운로드

```
ko@ko-virtual-machine: ~  
ko@ko-virtual-machine:~$ git clone git://git.denx.de/u-boot.git  
Command 'git' not found, but can be installed with:  
sudo apt install git  
ko@ko-virtual-machine:~$ sudo apt install git  
[sudo] password for ko: 
```

```
ko@ko-virtual-machine: ~  
ko@ko-virtual-machine:~$ git clone git://git.denx.de/u-boot.git  
Cloning into 'u-boot'...
```

U-boot 파일 확인

Makefile

- 소프트웨어 배포시에, 수정을 용이하게 하기 위해서 컴파일 과정과 소스 파일 사이에 관계들을 정의해 놓은 것

```
ko@ko-virtual-machine: ~/u-boot
ko@ko-virtual-machine:~$ pwd
/home/ko
ko@ko-virtual-machine:~$ ls
Desktop  Downloads  Pictures  snap      test_dir  Videos
Documents  Music      Public    Templates  u-boot
ko@ko-virtual-machine:~$ cd u-boot/
ko@ko-virtual-machine:~/u-boot$ ls
api      cmd      disk     env      Kbuild   MAINTAINERS  README
arch     common  doc      examples Kconfig   Makefile     scripts
board    config.mk  drivers  fs       lib       net          test
boot     configs  dts      include  Licenses  post         tools
ko@ko-virtual-machine:~/u-boot$
```

U-boot의 defconfig 파일

- 기본적인 환경설정
 - defconfig (default configuration)
- U-Boot 소스에서 Raspberry Pi3 부트로더 관련 파일 검색
 - 여러 파일이 검색되지만, 타겟 보드가 라즈베리파이 이므로 rpi_로 시작하는 파일을 찾으면 됨

The image displays three terminal windows illustrating the process of finding and editing the default configuration file for Raspberry Pi 3 in U-Boot.

Left Terminal: Shows the command `find ./ -name *defconfig*` being executed in the `~/u-boot` directory. The output lists numerous configuration files for various boards, including `apalis_imx6_defconfig`, `giedl_defconfig`, and `sniper_defconfig`.

Middle Terminal: Shows the command `find ./ -name *defconfig*` being executed in the `~/u-boot` directory. The output lists numerous configuration files for various boards, including `apalis_imx6_defconfig`, `giedl_defconfig`, and `sniper_defconfig`.

Right Terminal: Shows the command `gedit configs/rpi_3_defconfig` being executed. The terminal window displays the content of the `rpi_3_defconfig` file, which is a list of configuration options for the Raspberry Pi 3, such as `CONFIG_ARM=y`, `CONFIG_ARCH_CPU_INIT=y`, and `CONFIG_ARCH_BCM283X=y`.

Cross compile 환경 설정

toolchain 설치

- 소프트웨어 개발에 사용되는 프로그래밍 도구의 집합
- Arm 32bit 크로스컴파일러 설치
 - 32 bit , 64 bit 에 따라 입력 내용이 달라짐

```
ko@ko-virtual-machine: ~/u-boot
ko@ko-virtual-machine:~/u-boot$ sudo apt-get install gcc-arm-linux-gnueabi
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following package was automatically installed and is no longer required:
systemd-hwe-hwdb
```

크로스컴파일러 환경변수 설정을 위해 설정 명령 입력

- 툴체인 사용을 위해 설정 필요

```
ko@ko-virtual-machine: ~/u-boot
ko@ko-virtual-machine:~/u-boot$ export CROSS_COMPILE=arm-linux-gnueabi-
ko@ko-virtual-machine:~/u-boot$
```

deconfig 파일 make 실행

- U-boot 환경 설정을 타겟 보드에 맞게 설정하기 위해 make 이용
 - make 실행시 환경 설정 파일 .config 히든파일이 생성됨

```
ko@ko-virtual-machine: ~/u-boot
ko@ko-virtual-machine:~$ cd u-boot/
ko@ko-virtual-machine:~/u-boot$ make rpi_3_32b_deconfig
Command 'make' not found, but can be installed with:
sudo apt install make          # version 4.3-4.1build1, or
sudo apt install make-guile    # version 4.3-4.1build1
ko@ko-virtual-machine:~/u-boot$ sudo apt install make
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

```
ko@ko-virtual-machine: ~/u-boot
ko@ko-virtual-machine:~/u-boot$ make rpi_3_32b_deconfig
/bin/sh: 1: gcc: not found
HOSTCC scripts/basic/fixdep
/bin/sh: 1: cc: not found
make[1]: *** [scripts/Makefile.host:95: scripts/basic/fixdep] Error 127
make: *** [Makefile:492: scripts_basic] Error 2
ko@ko-virtual-machine:~/u-boot$
```

deconfig 파일 make 실행

해결책

- sudo apt install을 이용해 gcc 설치

```
ko@ko-virtual-machine:~/u-boot$ sudo apt install gcc
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following package was automatically installed and is no longer required:
  systemd-bus-busd
```

추가 설치

```
ko@ko-virtual-machine: ~/u-boot
ko@ko-virtual-machine:~/u-boot$ make rpi_3_32b_defconfig
HOSTCC scripts/basic/fixdep
HOSTCC scripts/kconfig/conf.o
YACC scripts/kconfig/zconf.tab.c
/bin/sh: 1: bison: not found
make[1]: *** [scripts/Makefile.lib:222: scripts/kconfig/zconf.tab.c] Error 127
make: *** [Makefile:575: rpi_3_32b_defconfig] Error 2
ko@ko-virtual-machine:~/u-boot$ sudo apt install bison
Reading package lists... Done
Building dependency tree... Done
```

deconfig 파일 make 실행

추가 설치

```
ko@ko-virtual-machine: ~/u-boot
ko@ko-virtual-machine:~/u-boot$ make rpi_3_32b_defconfig
YACC    scripts/kconfig/zconf.tab.c
LEX     scripts/kconfig/zconf.lex.c
/bin/sh: 1: flex: not found
make[1]: *** [scripts/Makefile.lib:214: scripts/kconfig/zconf.lex.c] Error 127
make: *** [Makefile:575: rpi_3_32b_defconfig] Error 2
ko@ko-virtual-machine:~/u-boot$ sudo apt install flex
Reading package lists... Done
Building dependency tree...
```

make 성공

- 정상적으로 설정 성공

```
ko@ko-virtual-machine: ~/u-boot
ko@ko-virtual-machine:~/u-boot$ make rpi_3_32b_defconfig
LEX     scripts/kconfig/zconf.lex.c
HOSTCC  scripts/kconfig/zconf.tab.o
HOSTLD  scripts/kconfig/conf
#
# configuration written to .config
#
ko@ko-virtual-machine:~/u-boot$
```


U-boot 빌드

- make 명령어로 부트로더 빌드

```
ko@ko-virtual-machine: ~/u-boot
ko@ko-virtual-machine:~/u-boot$ make
scripts/kconfig/conf  --syncconfig Kconfig
UPD      include/config.h
CFG      u-boot.cfg
GEN      include/autoconf.mk
```

- make과정 중 문제 발생 시

- libssl-dev 설치 후 make 다시 실행

```
In file included from tools/imagetool.h:24,
                  from tools/aisimage.c:7:
include/image.h:1264:12: fatal error: openssl/evp.h: No such file or directory
1264 | # include <openssl/evp.h>
      |           ^
compilation terminated.
make[1]: *** [scripts/Makefile.host:112: tools/aisimage.o] Error 1
make: *** [Makefile:1883: tools] Error 2
ko@ko-virtual-machine:~/u-boot$ sudo apt-get install libssl-dev
Reading package lists... Done
Building dependency tree... Done
```

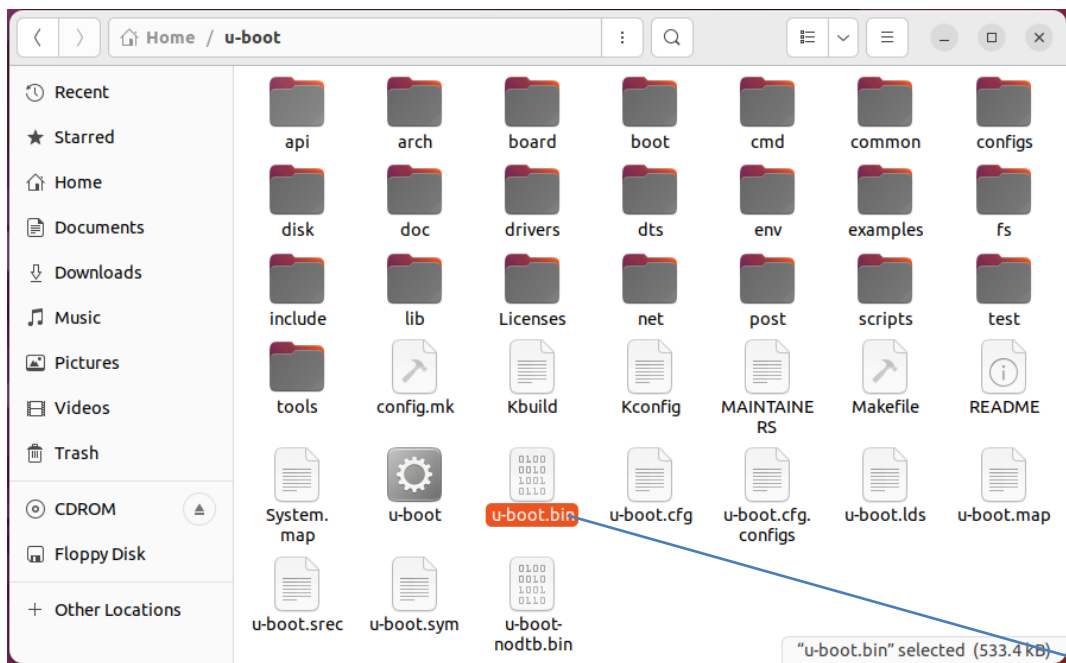

U-boot 빌드

- make 명령어 결과 u-boot.bin 파일이 생성되면 컴파일 성공

```
ko@ko-virtual-machine: ~/u-boot
ko@ko-virtual-machine:~/u-boot$ ls
api          disk         Kbuild       README       u-boot.cfg.configs
arch         doc          Kconfig      scripts      u-boot.lds
board        drivers      lib          System.map   u-boot.map
boot         dts          Licenses     test         u-boot-nodtb.bin
cmd          env          MAINTAINERS  tools        u-boot.srec
common       examples    Makefile     u-boot      u-boot.sym
config.mk    fs          net          u-boot.bin
configs      include     post         u-boot.cfg
ko@ko-virtual-machine:~/u-boot$
```

빌드 결과물 확인

● 생성된 u-boot.bin 파일을 Micro SD 카드에 붙여넣기



> OneDrive - Personal

> 내 PC

> 3D 개체

> 다운로드

> 동영상

> 문서

> 바탕 화면

> 사진

> 음악

> 로컬 디스크 (C:)

> USB 드라이브 (D:)

> boot (E:)

> USB 드라이브 (F:)

> USB 드라이브 (D:)

> 네트워크

config.txt

COPYING.linux

fixup.dat

fixup_cd.dat

fixup_db.dat

fixup_x.dat

fixup4.dat

fixup4cd.dat

fixup4db.dat

fixup4x.dat

issue.txt

kernel.img

kernel7.img

kernel7l.img

kernel8.img

LICENCE.broadcom

start.elf

start_cd.elf

start_db.elf

start_x.elf

start4.elf

start4cd.elf

start4db.elf

start4x.elf

u-boot.bin

2022-10-10 오전 11:23	텍스트 문서	2KB
2019-06-24 오후 3:21	LINUX 파일	19KB
2020-02-05 오후 3:25	DAT 파일	7KB
2020-02-05 오후 3:25	DAT 파일	3KB
2020-02-05 오후 3:25	DAT 파일	10KB
2020-02-05 오후 3:25	DAT 파일	10KB
2020-02-12 오후 1:33	DAT 파일	7KB
2020-02-12 오후 1:33	DAT 파일	4KB
2020-02-12 오후 1:30	DAT 파일	9KB
2020-02-12 오후 1:33	DAT 파일	9KB
2020-02-13 오후 4:16	텍스트 문서	1KB
2020-02-03 오후 12:50	ALZip IMG File	5,023KB
2020-02-03 오후 12:50	ALZip IMG File	5,298KB
2020-02-03 오후 12:50	ALZip IMG File	5,623KB
2020-02-03 오후 12:50	ALZip IMG File	13,205KB
2020-01-17 오전 10:01	BROADCOM 파일	2KB
2020-02-12 오후 1:33	ELF 파일	2,816KB
2020-02-12 오후 1:33	ELF 파일	675KB
2020-02-12 오후 1:33	ELF 파일	4,747KB
2020-02-12 오후 1:33	ELF 파일	3,709KB
2020-02-12 오후 1:33	ELF 파일	2,720KB
2020-02-12 오후 1:33	ELF 파일	766KB
2020-02-12 오후 1:33	ELF 파일	4,486KB
2020-02-12 오후 1:33	ELF 파일	3,464KB
2022-10-10 오후 3:12	ALZip BIN File	521KB

빌드 결과물 확인

- Micro SD 카드에 있는 config.txt 파일을 연 후 코드 가장 아랫부분에 kernel=u-boot.bin 입력

```
#dtoverlay=gpio-ir,gpio_pin=17
#dtoverlay=gpio-ir-tx,gpio_pin=18

# Additional overlays and parameters are documented /boot/overlays/README

# Enable audio (loads snd_bcm2835)
dtparam=audio=on

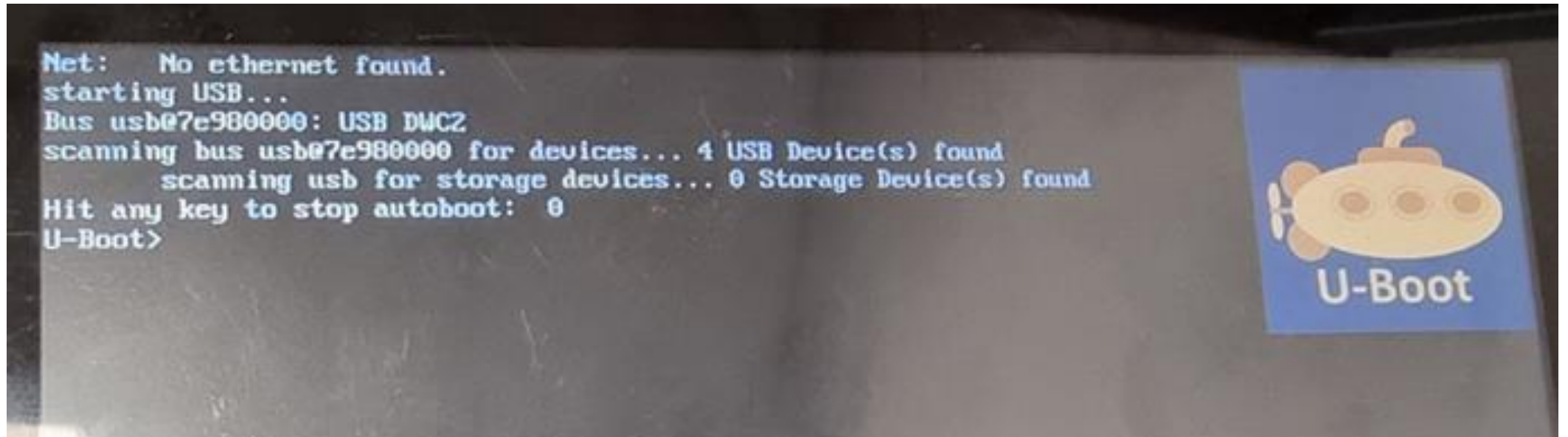
[pi4]
# Enable DRM VC4 V3D driver on top of the dispmanx display stack
dtoverlay=vc4-fkms-v3d
max_framebuffers=2

[all]
#dtoverlay=vc4-fkms-v3d
kernel=u-boot.bin
```

config.txt 파일 내용은 라즈베리언
버전마다 다를 수 있음

빌드 결과물 확인

- Micro SD 카드를 라즈베리파이에 삽입한 후 입력 후 3초 이내로 키보드 값 입력



실습2

U-boot 명령어 추가

명령어 프로그램 작성

- cmd 디렉토리에 새로 추가할 명령어 프로그램 작성
- 목표
 - hello 입력 시 hello HYU 출력
 - gedit을 이용해 hello.c 라는 모듈 작성을 통해 명령어 정의

```
ko@ko-virtual-machine: ~/u-boot/cmd
ko@ko-virtual-machine:~/u-boot$ cd cmd
ko@ko-virtual-machine:~/u-boot/cmd$ gedit hello.c
```

```
1 #include <common.h>
2 #include <command.h>
3 #include <fs.h>
4
5
6 static int say_hello(struct cmd_tbl *cmdtp, int flag, int argc, char* const argv[]){
7     printf("hello HYU\n");
8     return 0;
9 }
10
11 U_BOOT_CMD(hello, 1, 1, say_hello, "hello firmware command", "arg...");
12
```

Kconfig 파일 수정

- 명령어 정의 후 Kconfig 파일 수정해야 함
- Kconfig: 부트로더 컴파일 시 설정할 항목들을 서술해 놓은 파일
 - menuconfig 에서 확인 가능
- Kconfig 파일 열기

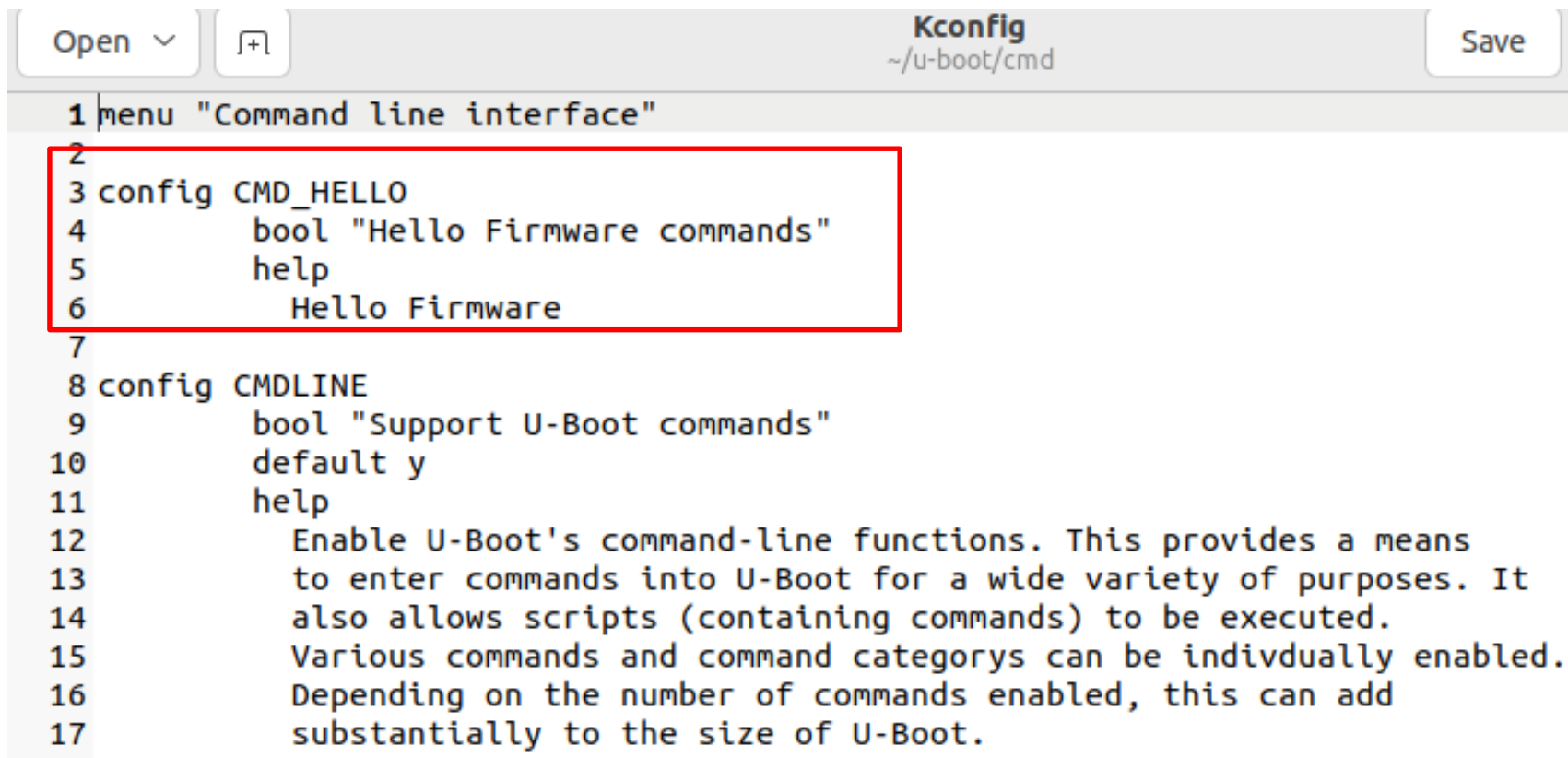
```
ko@ko-virtual-machine: ~/u-boot/cmd
ko@ko-virtual-machine:~/u-boot/cmd$ gedit Kconfig
```

```
Kconfig
~/u-boot/cmd

1 menu "Command line interface"
2
3 config CMDLINE
4     bool "Support U-Boot commands"
5     default y
6     help
7         Enable U-Boot's command-line functions. This provides a means
8         to enter commands into U-Boot for a wide variety of purposes. It
9         also allows scripts (containing commands) to be executed.
10        Various commands and command categorys can be individually enabled.
11        Depending on the number of commands enabled, this can add
12        substantially to the size of U-Boot.
13
14 config HUSH_PARSER
15     bool "Use hush shell"
16     depends on CMDLINE
17     help
18         This option enables the "hush" shell (from Busybox) as command line
19         interpreter, thus enabling powerful command line syntax like
20         if...then...else...fi conditionals or '&&' and '||'
21         constructs ("shell scripts").
22
23         If disabled, you get the old, much simpler behaviour with a somewhat
24         smaller memory footprint.
25
26 config CMDLINE_EDITING
27     bool "Enable command line editing"
28     depends on CMDLINE
29     default y
30     help
31         Enable editing and History functions for interactive command line
32         input operations
33
34 config CMDLINE_PS_SUPPORT
35     bool "Enable support for changing the command prompt string at run-time"
36     depends on HUSH_PARSER
37     help
```

Kconfig 파일 수정

🌐 Kconfig에 코드 추가



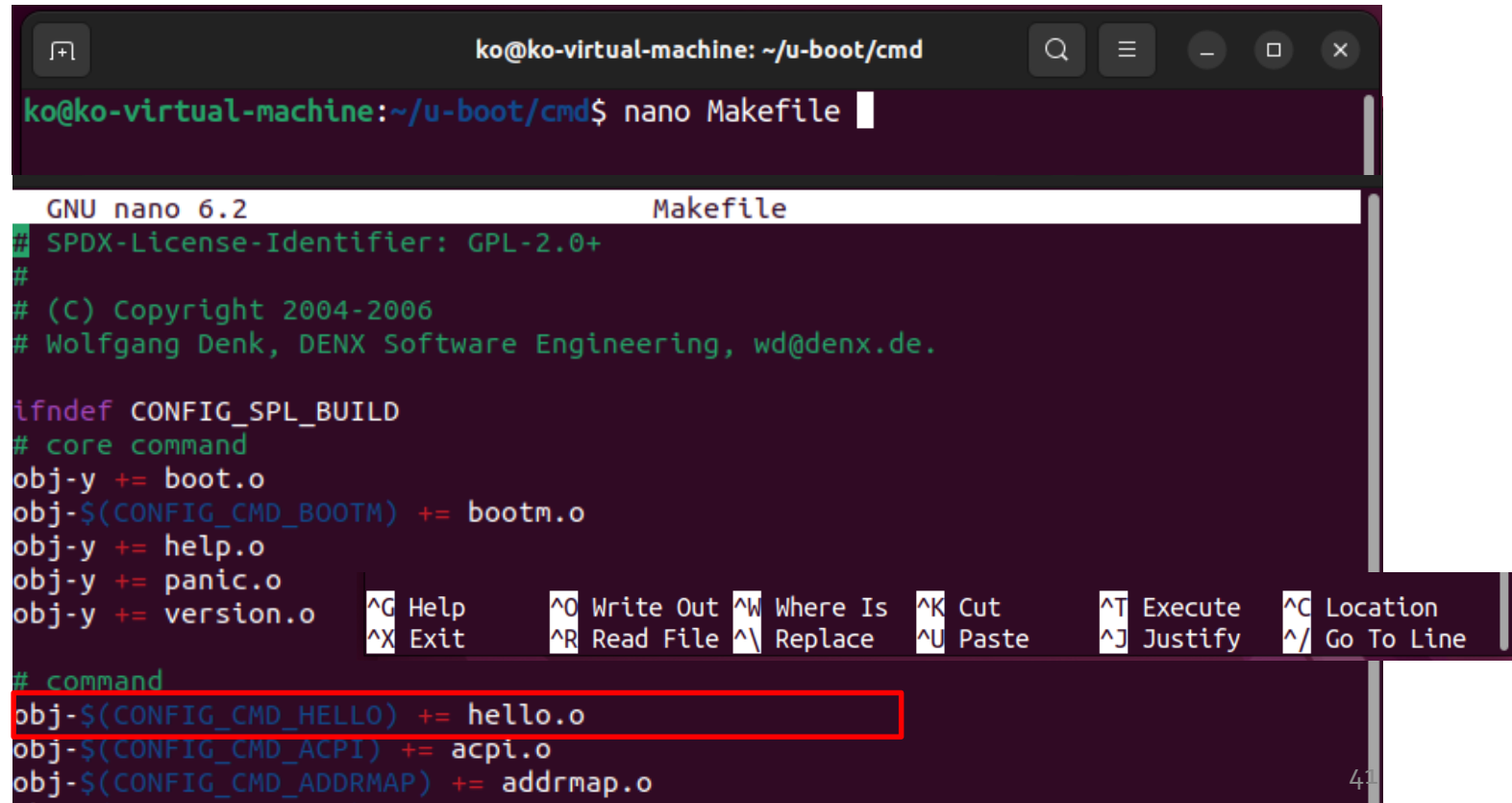
The screenshot shows a Kconfig editor window titled 'Kconfig' with the path '~ / u - boot / cmd'. The window has an 'Open' dropdown, a file icon, and a 'Save' button. The code is as follows:

```
1 menu "Command line interface"
2
3 config CMD_HELLO
4     bool "Hello Firmware commands"
5     help
6         Hello Firmware
7
8 config CMDLINE
9     bool "Support U-Boot commands"
10    default y
11    help
12        Enable U-Boot's command-line functions. This provides a means
13        to enter commands into U-Boot for a wide variety of purposes. It
14        also allows scripts (containing commands) to be executed.
15        Various commands and command categories can be individually enabled.
16        Depending on the number of commands enabled, this can add
17        substantially to the size of U-Boot.
```

A red rectangular box highlights the code block for `CMD_HELLO`, which includes the configuration name, a boolean option with a description, and a help text.

Makefile 수정

- Makefile : make에게 어떻게 컴파일할지, 어떤 명령을 수행할 것인지 알려주는 파일
- Makefile을 수정해서 menuconfig에서 빌드 가능하도록 설정
 - 빌드: 부트로더 이미지 만드는 과정
 - gedit, nano, vim 이용 가능



```
ko@ko-virtual-machine: ~/u-boot/cmd
ko@ko-virtual-machine:~/u-boot/cmd$ nano Makefile

GNU nano 6.2 Makefile
# SPDX-License-Identifier: GPL-2.0+
#
# (C) Copyright 2004-2006
# Wolfgang Denk, DENX Software Engineering, wd@denx.de.

ifndef CONFIG_SPL_BUILD
# core command
obj-y += boot.o
obj-$(CONFIG_CMD_BOOTM) += bootm.o
obj-y += help.o
obj-y += panic.o
obj-y += version.o
# command
obj-$(CONFIG_CMD_HELLO) += hello.o
obj-$(CONFIG_CMD_ACPI) += acpi.o
obj-$(CONFIG_CMD_ADDRMAP) += addrmap.o
```

생성된 명령어 추가

- menuconfig를 통해 새로 생성한 명령어 추가
 - fconfig는 기본적으로 작동하는 설정, menuconfig는 사용자가 직접 세부 설정 가능
 - make menuconfig 명령어를 통해 모든 디렉토리의 Kconfig의 내용을 확인하여 특정 module이 커널에 포함되도록 되어 있는지 확인 가능
 - Kconfig에서 추가한 hello 명령어를 선택함

```
ko@ko-virtual-machine: ~/u-boot
ko@ko-virtual-machine:~/u-boot/cmd$ cd ..
ko@ko-virtual-machine:~/u-boot$ make menuconfig
*
* Unable to find the ncurses package.
* Install ncurses (ncurses-devel or libncurses-dev
* depending on your distribution).
*
make[1]: *** [scripts/kconfig/Makefile:224: scripts/kconfig/.mconf-cfg] Error 1
make: *** [Makefile:575: menuconfig] Error 2
ko@ko-virtual-machine:~/u-boot$
```

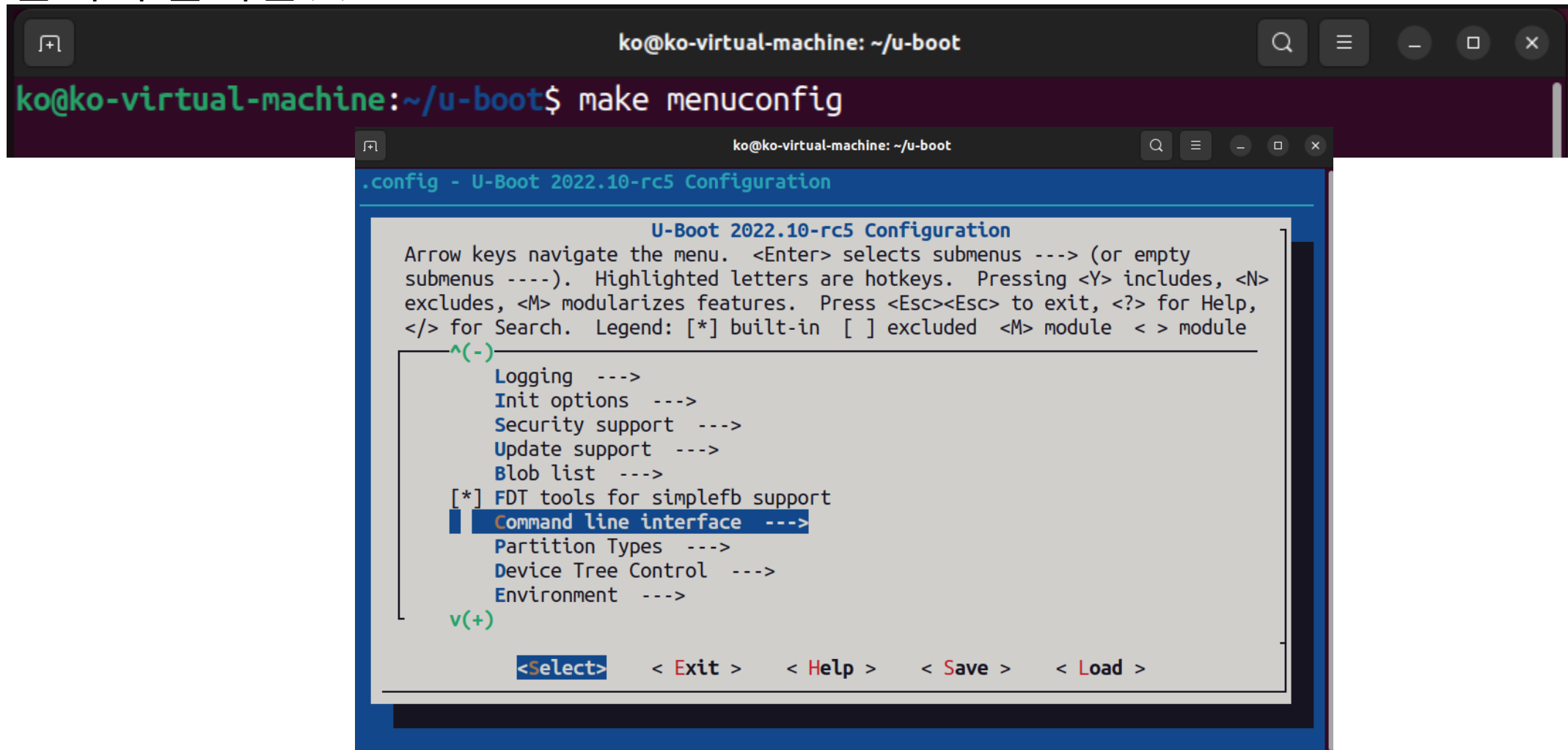
생성된 명령어 추가

문제점 해결

```
ko@ko-virtual-machine: ~/u-boot
ko@ko-virtual-machine:~/u-boot$ make menuconfig
*
* Unable to find the ncurses package.
* Install ncurses (ncurses-devel or libncurses-dev
* depending on your distribution).
*
make[1]: *** [scripts/kconfig/Makefile:224: scripts/kconfig/.mconf-cfg] Error 1
make: *** [Makefile:575: menuconfig] Error 2
ko@ko-virtual-machine:~/u-boot$ sudo apt-get install libncurses-dev
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following package was automatically installed and is no longer required:
```

생성된 명령어 추가

- make menuconfig 실행후 생성된 창에서 command line interface 에 엔터를 눌러서 들어갈것



```
ko@ko-virtual-machine: ~/u-boot
ko@ko-virtual-machine:~/u-boot$ make menuconfig

.config - U-Boot 2022.10-rc5 Configuration

U-Boot 2022.10-rc5 Configuration
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty
submenus ----). Highlighted letters are hotkeys. Pressing <Y> includes, <N>
excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help,
</> for Search. Legend: [*] built-in [ ] excluded <M> module < > module

^(-)
  Logging --->
  Init options --->
  Security support --->
  Update support --->
  Blob list --->
  [*] FDT tools for simplefb support
  Command line interface --->
  Partition Types --->
  Device Tree Control --->
  Environment --->
v(+)

<Select>  < Exit >  < Help >  < Save >  < Load >
```

생성된 명령어 추가

- Hello Firmware Commands 선택 후 (스페이스 바를 누르면 선택됨) 저장
- Save 는 화살표 방향 키를 이용하여 선택 가능

```
ko@ko-virtual-machine: ~/u-boot
.config - U-Boot 2022.10-rc5 Configuration
> Command line interface

Command line interface
Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty
submenus ----). Highlighted letters are hotkeys. Pressing <Y> includes, <N>
excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help,
</> for Search. Legend: [*] built-in [ ] excluded <M> module < > module

[*] Hello Firmware Commands
[*] Support U-Boot commands
  *- Use hush shell
  *- Enable command line editing
[ ] Enable support for changing the command prompt string at run-time
  *- Enable auto complete using TAB
  *- Enable long help messages
(U-Boot> ) Shell prompt
(> ) Hush shell secondary prompt
(16) Maximum number arguments accepted by commands
v(+)
```

<Select> <Exit> <Help> <Save> <Load>

```
ko@ko-virtual-machine: ~/u-boot
.config - U-Boot 2022.10-rc5 Configuration

Enter a filename to which this configuration
should be saved as an alternate. Leave blank to
abort.

.config
```

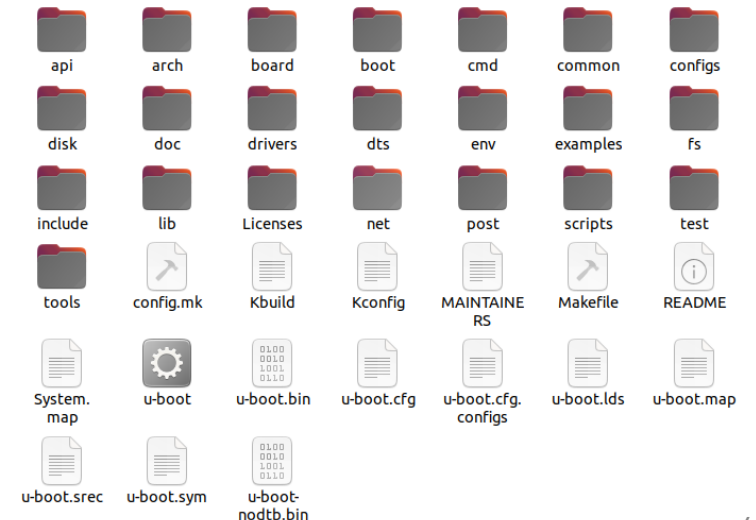
<Ok> <Help>

u-boot 컴파일

- export CROSS_COMPILE=arm-linux-gnueabi- 입력 후 make 입력

```
ko@ko-virtual-machine:~/u-boot$ export CROSS_COMPILE=arm-linux-gnueabi-
ko@ko-virtual-machine:~/u-boot$ make
scripts/kconfig/conf  --synconfig Kconfig
CFG      u-boot.cfg
GEN      include/autoconf.mk
GEN      include/autoconf.mk.dep
===== WARNING =====
```

- u-boot.bin 파일 생성 확인 후 Micro SD 카드에 복사



결과 화면 확인

- Micro SD카드 삽입 후 전원 넣은 뒤 3초 이내로 임의의 키 값 입력하여 U-Boot 쉘 활성화
- hello 입력하면 hello HYU 가 출력되게 됨

