Embedded System Software [CSE4116]

실습 1 주차

Department of Computer Science and Engineering, Sogang University, Seoul, South Korea

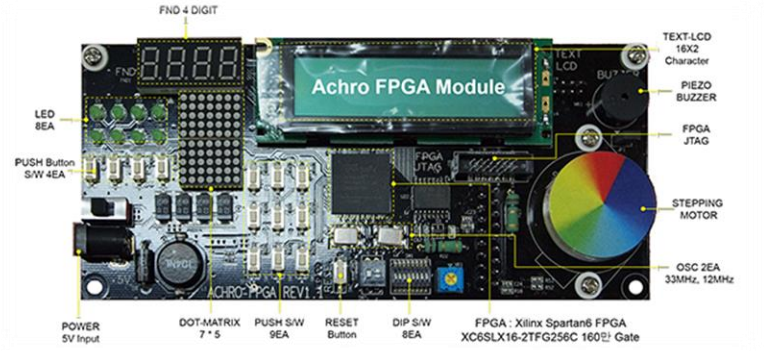
Data-Intensive and Computing and System Laboratory

1. Target Board

- Freescale I.MX6Q Cortex-A9 Dual core
- USB 2.0, SATA 1.0/2.0/3.0 Interface
- MIPI(Mobile Industry Processor Interface)
- EMMC 4.4 / T-FLASH
- WiFi / Bluetooth / GPS
- Ethernet 10/100M bps
- HDMI mirror screen
- 10 points touchscreen
- Flexcan

1.1. Overview

- LED 8
- DOT-MATRIX
- FND
- BUZZER
- DIP S/W
- PUSH S/W
- STEPPING MOTOR



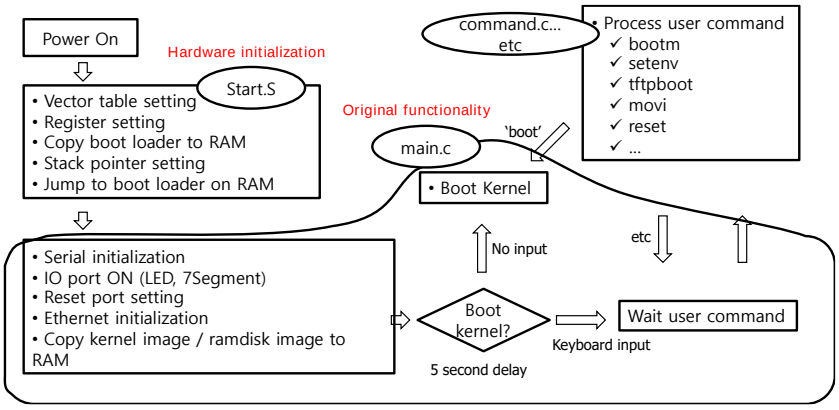
1.2. FPGA Module

1.3. Processor

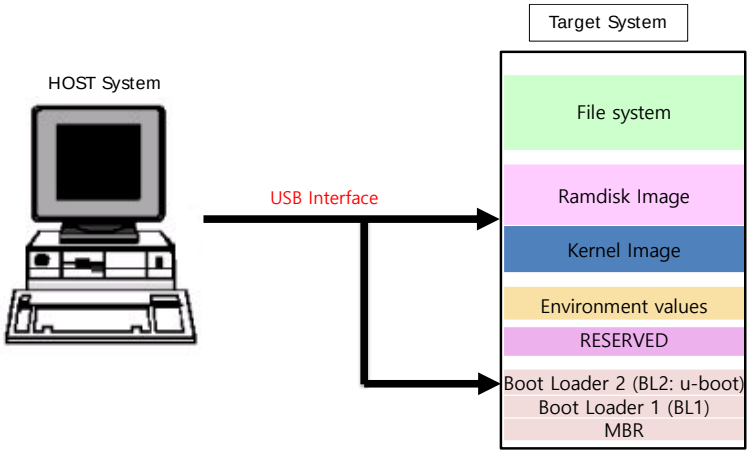
ARM Cortex-A9 기반의 듀얼코어 프로세서

2. Boot Loader

- 전원이 들어오면, 사용이 가능한 하드웨어를 초기화
- 커널을 메모리로 올려서 리눅스가 부팅될 수 있도록 해주는 역할

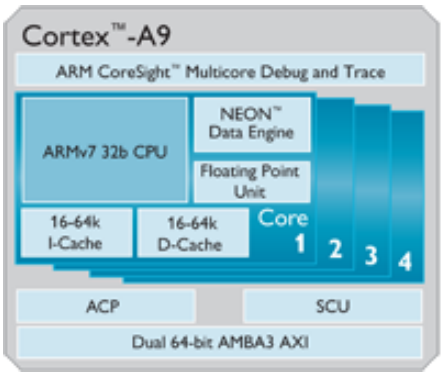


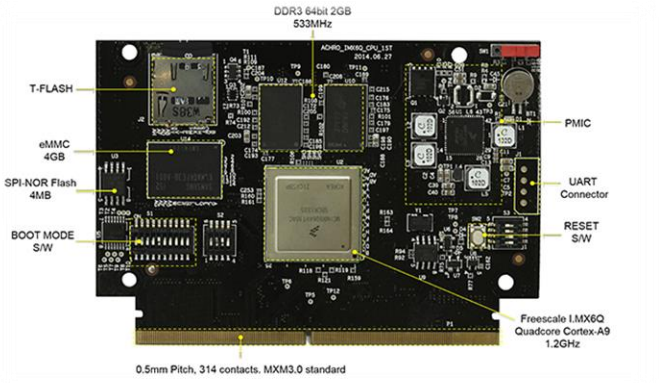
A. Boot Loader in Ahcro-imx



B. 커널의 기능

- 프로세스 관리(Process Management)
- 메모리 관리(Memory Management)
- 파일 시스템 관리(File System Management)
- 디바이스 관리(Device Management)
- 네트워크 관리(Network Management)





3. 환경 설정

3.1. Cross Compiler 설치

```
$  sudo dpkg-reconfigure -plow dash
```

→ No 선택

```
$  cd ~/Downloads
$  sudo mkdir /opt/toolchains
$  wget http://sources.buildroot.net/toolchain-external-codesourcery-arm/arm-2014.05-29-arm-none-linux-gnueabi-i686-pc-linux-gnu.tar.bz2
$  sudo tar -jxvf arm-2014.05-29-arm-none-linux-gnueabi-i686-pc-linux-gnu.tar.bz2 -C /opt/toolchains
```

- wget 위 사이트가 안될 시 아래에서도 다운 가능

```
$  wget --user=embe2024 --password=[칠판참고] ftp://dcclab.synology.me/embe2024/arm-2014.05-29-arm-none-linux-gnueabi-i686-pc-linux-gnu.tar.bz2
```

라이브러리 설치

```
$  sudo apt install libx11-dev libc6-i386 libc6-dev-i386
$  sudo apt install libstdc++6:i386 lib32z1 lib32ncurses5-dev
```

A. 환경변수 설정

- Vi 버전 업그레이드 및 .basrh 파일 편집

```
$  sudo apt install -y vim
$  vi ~/.bashrc
```

.bashrc 파일 맨 아래 다음 내용을 추가

```
export CROSS_COMPILE=arm-none-linux-gnueabi-
export PATH=/opt/toolchains/arm-2014.05/bin:$PATH
export ARCH=arm
```

- bashrc 갱신

```
$  source ~/.bashrc
```

- 설치 확인(임의로 hello.c 를 작성 후 Cross Compile 수행)

```
$  arm-none-linux-gnueabi-gcc -static -o hello hello.c
```

```
root@ubuntu:~# file ./hello
./hello: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV), statically linked, for GNU/Linux 2.6.16, not stripped
root@ubuntu:~# ./hello
bash: ./hello: cannot execute binary file: Exec format error
root@ubuntu:~#
```

→ 현재 환경에서 작동하지 않음

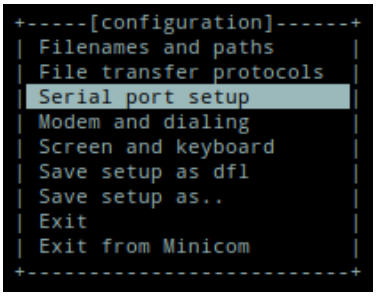
3.2. Minicom 설치

```
$  sudo apt install minicom
```

A. 환경설정

```
$  sudo minicom -s
```

- → Serial port setup
- (a 키 입력) 통신포트: /dev/ttyUSB0
- (f 키 입력) flow control: Hard → no
  Soft → no



- Save as setup as dfl 로 저장 후 Exit

3.3. Java Library 설치

```
$  sudo apt install git git-core gnupg flex bison gperf build-essential
$  sudo apt install zip curl libc6-dev libncurses5-dev:i386 x11proto-core-dev
$  sudo apt install libx11-dev:i386 libreadline6-dev:i386
$  sudo apt install libgl1-mesa-dev g++-multilib mingw-w64 tofrodos u-boot-tools
$  sudo apt install python-markdown python libxml2-utils xsltproc zlib1g-dev:i386
$  sudo apt install uuid uuid-dev zlib1g-dev liblz4-dev liblzo2-dev
```

3.4. Oracle JDK 설정

```
$  cd ~/Downloads
$  wget --user=embe2024 --password=[칠판참고] ftp://dcclab.synology.me/embe2024/jdk-6u45-linux-x64.bin
$  sh jdk-6u45-linux-x64.bin
$  sudo mkdir /usr/lib/jvm/
$  sudo mv jdk1.6.0_45 /usr/lib/jvm/java-6-oracle
```

A. 환경변수 설정

.bashrc 파일 맨 아래 다음 내용을 추가

```
export PATH=/usr/lib/jvm/java-6-oracle/jre/bin:$PATH
export PATH=/usr/lib/jvm/java-6-oracle/bin:$PATH
export JAVA_HOME=/usr/lib/jvm/java-6-oracle/jre/bin/java
export ANDROID_JAVA_HOME=/usr/lib/jvm/java-6-oracle
```

- bashrc 갱신 & Java version 확인

```
$  source ~/.bashrc
$  java -version
```

```
root@ubuntu:~# java -version
java version "1.6.0_45"
Java(TM) SE Runtime Environment (build 1.6.0_45-b06)
Java HotSpot(TM) 64-Bit Server VM (build 20.45-b01, mixed mode)
```

3.5. gcc/g++ Version Downgrade

```
$  sudo add-apt-repository 'deb http://archive.ubuntu.com/ubuntu/ trusty main'
$  sudo add-apt-repository 'deb http://archive.ubuntu.com/ubuntu/ trusty universe'
$  sudo apt install gcc-4.8 g++-4.8
$  sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-4.8 50 --slave /usr/bin/g++ g++ /usr/bin/g++-4.8
$  sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-9 10 --slave /usr/bin/g++ g++ /usr/bin/g++-9
```

- Version 확인

```
$  gcc --version
```

- Version 을 바꾸고 싶은 경우

```
$  sudo update-alternatives --config gcc
```

3.6. Fastboot tool 설치

```
$  sudo apt install android-tools-fastboot
```

4. 실습

4.1. SD 카드 초기화

보드에서 SD카드를 제거하고 카드 리더기에 SD카드를 장착하여 컴퓨터에 연결
(SD카드가 부러지지 않도록 주의)

디바이스 확인

```
$ lsblk
```

```
sdc           8:32   1  14.9G  0 disk
├─sdc1        8:33   1    32M  0 part
├─sdc2        8:34   1    32M  0 part
├─sdc3        8:35   1     1K  0 part
├─sdc4        8:36   1   6.3G  0 part /media/hh/ACHROIMX_DATA
├─sdc5        8:37   1   512M  0 part /media/hh/57f8f4bc-abf4-655f-bf67-946fc0f9f25b
├─sdc6        8:38   1   512M  0 part /media/hh/ACHROIMX_CACHE
├─sdc7        8:39   1     8M  0 part /media/hh/ACHROIMX_DEVICE
└─sdc8        8:40   1     4M  0 part
```

(초기화 전)

SD카드가 어떤 디바이스인지 잘 살필 것! (위 예시에서만 sdc, 본인 PC 에서는 다를 수 있음)

- (주의) 아래 명령어로 SD 카드를 장착한 **리더기 이름**과 **용량(7.4G or 14.9G)**을 반드시 확인하여 초기화 명령어의 **[*]** 부분에 *device name(sdb, sdc 등)*을 정확하게 입력할 것 Ex) /dev/sdb

```
$ lsblk -o +model
```

```
embe@embe-desktop:~$ lsblk -o +model
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT                                MODEL
loop0    7:0    0     4K  1 loop /snap/bare/5
loop1    7:1    0  54.2M  1 loop /snap/snap-store/558
loop2    7:2    0 346.3M  1 loop /snap/gnome-3-38-2004/115
loop3    7:3    0    62M  1 loop /snap/core20/1611
loop4    7:4    0    47M  1 loop /snap/snapd/16292
loop5    7:5    0  91.7M  1 loop /snap/gtk-common-themes/1535
sda      8:0    0 238.5G  0 disk                                          Samsung_SSD_850_PRO_256GB
├─sda1   8:1    0   512M  0 part /boot/efi
└─sda2   8:2    0   238G  0 part /
sdb      8:16   1   7.4G  0 disk                                          STORAGE_DEVICE
├─sdb1   8:17   1   3.6G  0 part /media/embe/Ubuntu 20.04.5 LTS amd64
├─sdb2   8:18   1   3.9M  0 part
└─sdb3   8:19   1   3.8G  0 part /media/embe/writable
```

초기화 수행

```
$  sudo umount /dev/sd[*]
$  sudo dd if=/dev/zero of=/dev/sd[*] bs=512 count=1 conv=notrunc
```

- 완료 후 리더기를 컴퓨터에서 제거했다가 다시 연결하여 초기화 상태를 확인

```
embe@embe-desktop:~$ lsblk
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
loop0    7:0    0     4K  1 loop /snap/bare/5
loop1    7:1    0    62M  1 loop /snap/core20/1611
loop2    7:2    0  54.2M  1 loop /snap/snap-store/558
loop3    7:3    0 346.3M  1 loop /snap/gnome-3-38-2004/119
loop4    7:4    0  63.3M  1 loop /snap/core20/1828
loop5    7:5    0 346.3M  1 loop /snap/gnome-3-38-2004/115
loop6    7:6    0  49.9M  1 loop /snap/snapd/18357
loop7    7:7    0  91.7M  1 loop /snap/gtk-common-themes/1535
loop8    7:8    0    46M  1 loop /snap/snap-store/638
sda      8:0    0 238.5G  0 disk
├─sda1   8:1    0   512M  0 part /boot/efi
└─sda2   8:2    0   238G  0 part /
sdb      8:16   1   7.4G  0 disk
embe@embe-desktop:~$
```

- 위와 같이 초기화가 되지 않을 경우 PC 재부팅 후 다시 시도

4.2. Work 디렉토리 생성

```
$  sudo mkdir /work/
$  sudo chown -R embe2024:embe2024 /work/
```

- 개인 노트북 계정을 사용시 위에서 embe 를 자신의 계정 이름으로 사용할 것

```
$  echo $USER
```

4.3. 부트로더 컴파일

```
$  cd ~/Downloads
$  wget --user=embe2024 --password=[칠판참고] ftp://dcclab.synology.me/embe2024/u-boot.tar.bz2
$  tar -xvf u-boot.tar.bz2 -C /work/
$  cd /work/u-boot/
$  make distclean
$  make clean
$  make achroimx_android_sd_config
$  make -j [number of cores]
```

- 코어 개수 확인 방법

```
$  grep -c processor /proc/cpuinfo
```

파티션 생성 및 부트로더 기록

```
$  sudo ./part_sd.sh sd[*]
$  sudo ./write_to_sd.sh sd[*]
```

→ 부트로더를 기록하면 부팅 및 minicom 연결이 가능해짐

4.4. 커널 컴파일

```
$  cd ~/Downloads
$  wget --user=embe2024 --password=[칠판참고] ftp://dcclab.synology.me/embe2024/kernel.tar.gz
$  tar -zxvf kernel.tar.gz -C /work/
$  cd /work/achroimx_kernel/
$  make clean
$  make achroimx_defconfig
$  make -j [number of cores]
```

- mkimage 관련 에러 발생 시

```
$  cp /work/u-boot/tools/mkimage /usr/bin
```

4.5. 파일시스템 컴파일

make 시간이 굉장히 오래 걸리므로 이미 make가 된 파일을 사용함.

```
$  cd ~/Downloads
$  wget --user=embe2024 --password=[칠판참고] ftp://dcclab.synology.me/embe2024/android.tar.gz
$  tar -zxvf android.tar.gz -C /work/
$  cd /work/android/
```

A. GNU make Build 버전 추가

Make를 위한 build 파일을 다음과 같이 추가하여 수정

```
$  vi build/core/main.mk
```

```
# Check for broken versions of make.
# (Allow any version under Cygwin since we don't actually build the platform there.)
ifeq (,$(findstring CYGWIN,$(shell uname -sm)))
ifeq (0,$(shell expr $$(echo $(MAKE_VERSION) | sed "s/[^0-9\.].*//") = 3.81))
ifeq (0,$(shell expr $$(echo $(MAKE_VERSION) | sed "s/[^0-9\.].*//") = 3.82))
ifeq (0,$(shell expr $$(echo $(MAKE_VERSION) | sed "s/[^0-9\.].*//") = 4.1))
ifeq (0,$(shell expr $$(echo $(MAKE_VERSION) | sed "s/[^0-9\.].*//") = 4.2.1))
$(warning ********************************************************************************)
$(warning *  You are using version $(MAKE_VERSION) of make.)
$(warning *  Android can only be built by versions 3.81 and 3.82.)
$(warning *  see https://source.android.com/source/download.html)
$(warning ********************************************************************************)
$(error stopping)
endif
endif
endif
endif
endif
```

B. 종속성 에러문제 해결을 위한 코드 수정

external/mtd-utils/mkfs.ubifs/mkfs.ubifs.h 파일에 sys/sysmacros.h 헤더파일 추가

```
$  vi external/mtd-utils/mkfs.ubifs/mkfs.ubifs.h
```

```
#include "crc32.h"
#include "defs.h"
#include "crc16.h"
#include "ubifs-media.h"
#include "ubifs.h"
#include "key.h"
#include "lpt.h"
#include "compr.h"

#include <sys/sysmacros.h>

/*
 * Compression flags are duplicated so that compr.c can compile without ubifs.h.
 * Here we make sure they are the same.
 */
#if MKFS_UBIFS_COMPR_NONE != UBIFS_COMPR_NONE
#error MKFS_UBIFS_COMPR_NONE != UBIFS_COMPR_NONE
#endif
#if MKFS_UBIFS_COMPR_LZO != UBIFS_COMPR_LZO
#error MKFS_UBIFS_COMPR_LZO != UBIFS_COMPR_LZO
"external/mtd-utils/mkfs.ubifs/mkfs.ubifs.h" 148L, 4429C written
```

C. **make_bootimg.sh 파일에서 KERNEL_ZIMAGE 경로 설정**
기존의 KERNEL_ZIMAGE 경로를 주석처리하고, 새로운 경로 추가

```
$ vi make_bootimg.sh
```

```
KERNEL_ZIMAGE="../achroimx_kernel/arch/arm/boot/zImage"
```

```
#KERNEL_ZIMAGE="../kernel/arch/arm/boot/zImage"
KERNEL_ZIMAGE="../achrolmx_kernel/arch/arm/boot/zImage"
```

이후에 Build 실행

```
$ ./build_android.sh
```

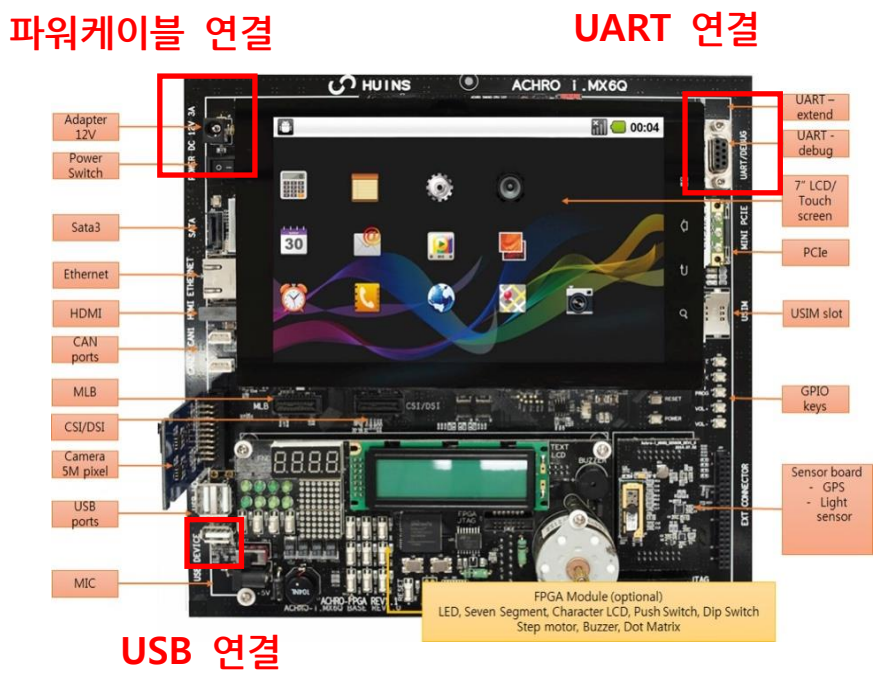
- Make 를 처음부터 할 시(생략)

```
$ cd ~/Downloads
$ wget --user=embe2024 --password=[칠판참고] ftp://dcclab.synology.me/embe2024/android.tar.gz
$ tar -zxvf android.tar.gz -C /work/
$ cd /work/android/
$ make clean
$ make clobber
$ sudo ./build_android.sh
```

4.6. 커널/파일시스템 기록 (/work/android 디렉토리에서 진행)

A. **Host 와 Target Device(보드) 연결**

위에서 부트로더를 기록한 SD 카드를 보드에 장착 후,
PC 와 보드를 두 케이블 (USB, UART)을 통해 연결

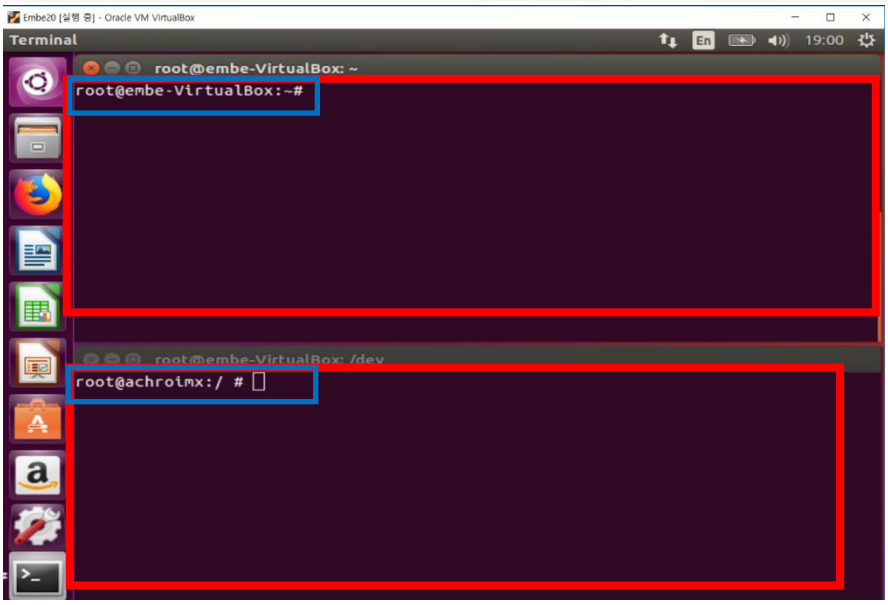


보드의 파워케이블을 연결 후 보드 전원 ON

B. **Minicom**

아래와 같이 두개의 터미널을 띄워 target 쪽에 minicom을 실행함.

```
$ sudo minicom
```



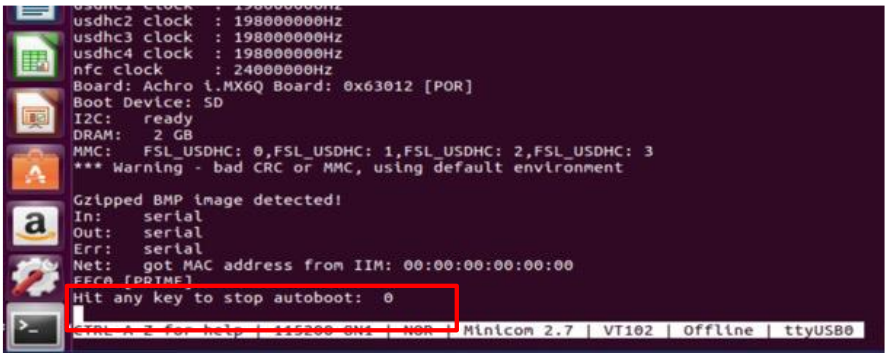
Host (PC)

Target (보드)

C. **u-boot 모드 진입 및 fastboot (On the Target Device(보드)-side)**

a. Minicom을 켜둔 상태에서 보드의 전원을 껐다 킴(or Reset 버튼)

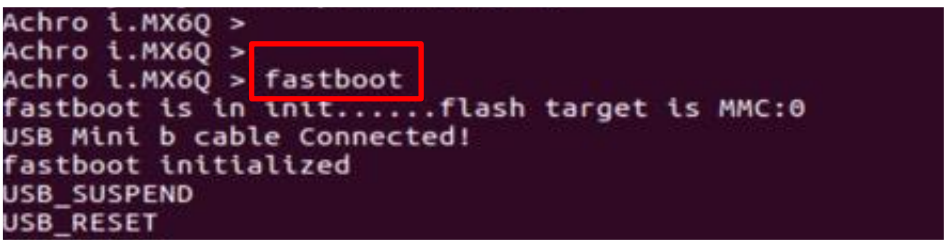
b. 부팅이 되기 전 Minicom에서 아무키나 입력하여 auto-boot을 종료



제한 초가 짧습니다.

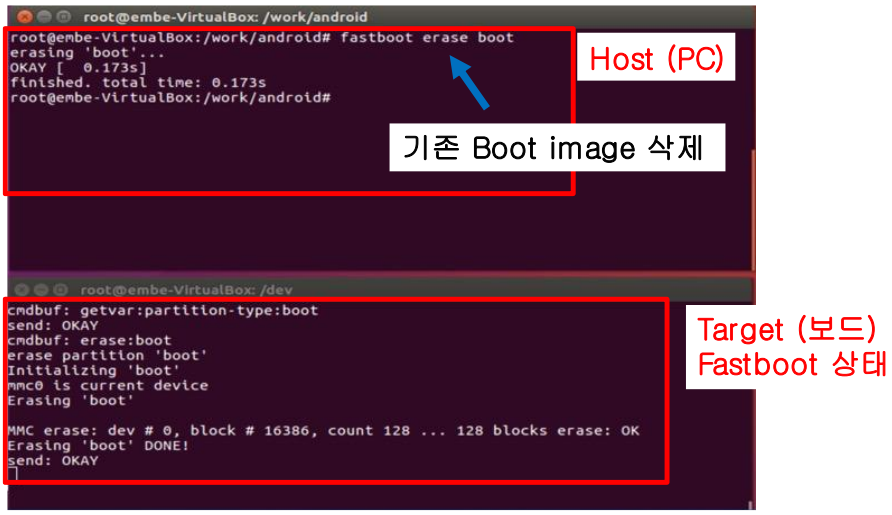
c. Shell창이 나오면 fastboot을 입력

```
$ fastboot
```

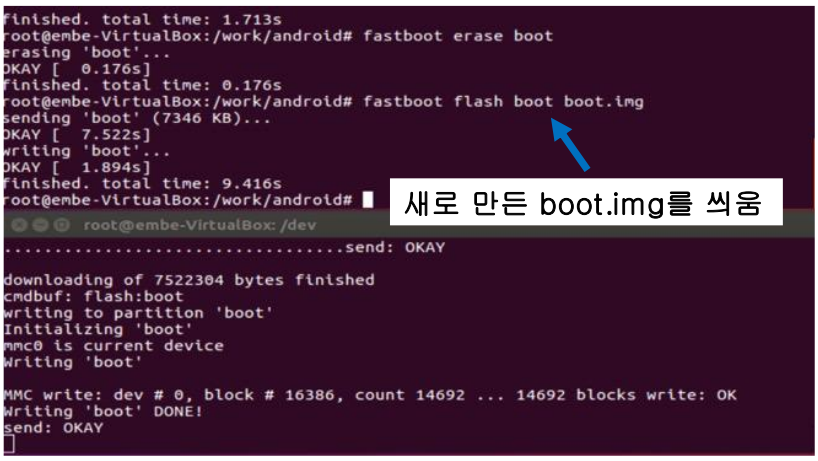


D. **커널 이미지 업데이트(On the host-side)**

```
$  fastboot erase boot
```




```
$  fastboot flash boot boot.img
```



E. **파일시스템 이미지 업데이트(On the host-side)**

```
$  fastboot erase system
$  fastboot flash system system.img
```

완료 후 (5 분 이상 소요) 보드 reboot

```
$  fastboot reboot
```

F. **안드로이드 리눅스 저장 공간 확보 (On the target board-side, in the Minicom)**

dmseg 가 출력되는 것을 무시하고 아래 명령어를 입력 (minicom에서 엔터 눌러보기)

```
$  mount -o rw,remount,size=6G /dev/mmcblk0p4 /data
```

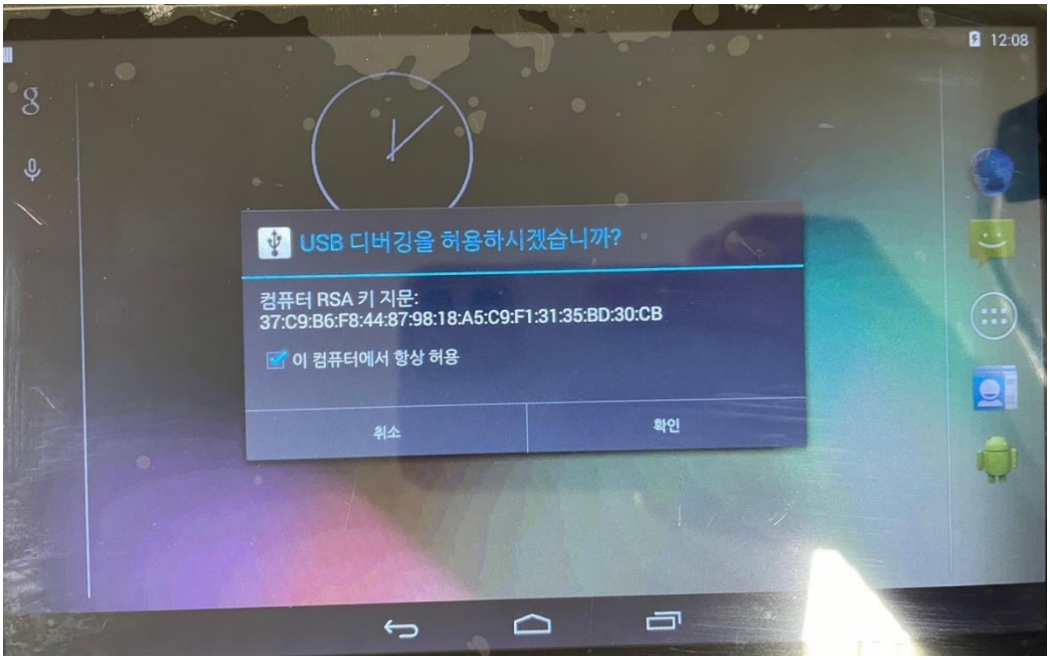
5. 실습 과제

호스트에서 “본인의 학번, 이름”을 출력하는 프로그램 작성

1. Cross-compile 을 통해서 binary 파일 생성
2. Binary 파일을 보드로 전송 (아래는 예시)

```
$ adb push [filename] /data/local/tmp
```

이때 반드시 화면에서 다음과 같이 USB 디버깅을 허용해야함.



(참고) [PROG] 버튼을 누르면 화면이 켜짐.

3. minicom 을 통해 보드에서 binary 파일 실행
4. 출력 화면 검사 요청