

# Lab 1. Embedded Board and Development Environment Setup

## 1. Purpose

Lab 1의 목적은 PC-based environment에서 embedded system 설계를 위해 cross-compiling system을 구축하고 테스트하는데 있다. 이를 위해 다양한 측면들을 확인하고 테스트한다. PC에서는 Ubuntu Linux 환경을, Lab에서 활용하는 embedded board인 BeagleBone에서는 Debian Linux 환경을 사용하며 C code build system(CMake), cross-compile, NFS, 그리고 GUI(VSCode)등을 구현해보고 테스트한다.

- Bash Shell과 Linux 사용에 익숙해진다.
- SSH(network, NFS)를 이용하여 Linux 설정을 완료한다.
- C code의 compilin과 corss-compiling 과정을 이해한다.
- 기본적인 debugging을 연습한다.

## 2. Experiment Sequence

본 lab은 세가지 problem으로 구성되어 있다. 전체적인 lab의 내용은 cross-compiling system을 구축하고 C++ language를 이용해 embedded appilication example을 구현하고 실행하는 것이다. 아래는 구현하게 될 세가지 problem이다.

### (1) Problem 1A

Lab PC에 cross-development system을 구축한다. "Hello World!" 코드를 작성하고 cross-compile한다. NFS와 SSH를 사용하여 embedded system에 program을 다운로드하고 실행한다.

### (2) Problem 1B

Countdown program: 500Hz의 timed loop을 사용하여 10부터 0까지 countdown하는 프로그램을 작성하고 실행한다.

### (3) Problem 1C

Reaction timer program: 두개의 키보드 input 사이의 time interval을 측정하는 프로그램을 작성하고 실행한다.

- 1) 프로그램이 실행되면 컴퓨터가 "Type 's' key to start timer."라는 메시지를 출력한다.
- 2) 's' 키를 누를 경우 timer가 시작되며, 그렇지 않을 경우 시작되지 않는다.
- 3) Timer가 시작되면, "Type 'e' key to end timer."라는 메시지를 출력한다.
- 4) 'e' 키를 누를 경우 timer가 종료되며, timer's execution time과 loop counter가 output으로 나오게 된다.

### 3. Experiment Results

#### Preparation

Beaglebone Black을 이용한 cross-development system을 구축하기 위해 선행되어야 할 기본적인 세팅 및 설정들을 하였다. 가장 먼저 PC에 설정된 ip 주소를 확인하고 정상적으로 network connection이 이루어지는 확인하기 위해 ping 명령어를 보냈으며 정상적으로 연결되어 있는 것을 확인했다. 이후, Debian OS가 설치되어 있는 SD카드를 Beaglebone Black에 삽입하고 이를 PC와 연결하여 부팅하였다.

```

kaist@kaist-n5: ~
File Edit View Search Terminal Help
kaist@kaist-n5:~$ ifconfig
eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 143.248.141.124 netmask 255.255.255.0 broadcast 143.248.141.255
    inet6 fe80::32c8:da47:46e4:409e prefixlen 64 scopeid 0x20<link>
    inet6 fe80::ee:d4f2:71e1:f050 prefixlen 64 scopeid 0x20<link>
    inet6 fe80::37a3:1618:1a20:cc55 prefixlen 64 scopeid 0x20<link>
    ether b4:2e:99:c8:a4:71 txqueuelen 1000 (Ethernet)
    RX packets 578 bytes 272244 (272.2 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 162 bytes 15828 (15.8 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
    device interrupt 16 memory 0x55200000-55220000

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 179 bytes 18785 (18.7 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 179 bytes 18785 (18.7 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

kaist@kaist-n5:~$ ping 143.248.141.124 -c 5
PING 143.248.141.124 (143.248.141.124) 56(84) bytes of data.
64 bytes from 143.248.141.124: icmp_seq=1 ttl=64 time=0.034 ms
64 bytes from 143.248.141.124: icmp_seq=2 ttl=64 time=0.085 ms
64 bytes from 143.248.141.124: icmp_seq=3 ttl=64 time=0.039 ms
64 bytes from 143.248.141.124: icmp_seq=4 ttl=64 time=0.039 ms
64 bytes from 143.248.141.124: icmp_seq=5 ttl=64 time=0.040 ms

--- 143.248.141.124 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4085ms
rtt min/avg/max/mdev = 0.034/0.047/0.085/0.019 ms
kaist@kaist-n5:~$

```

그림 1. PC의 network connection 상태 확인

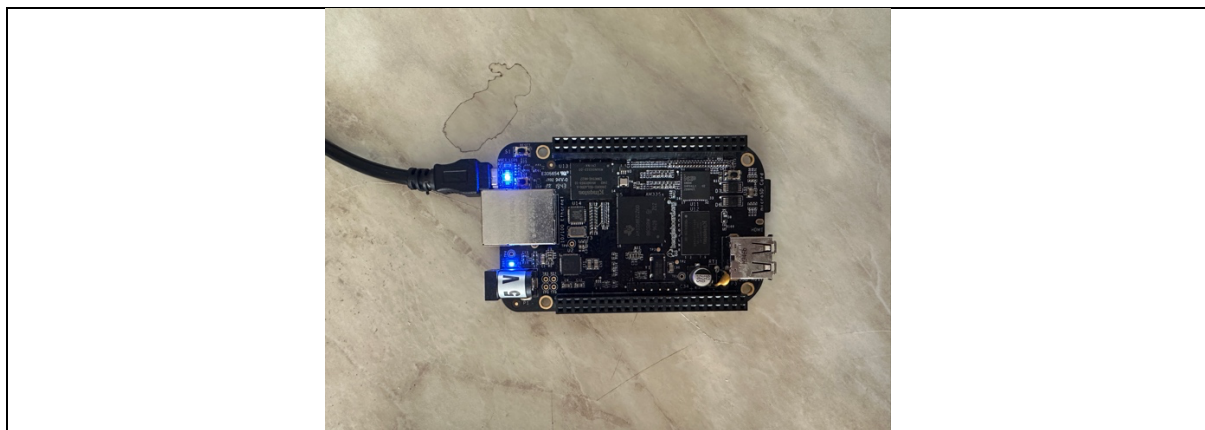
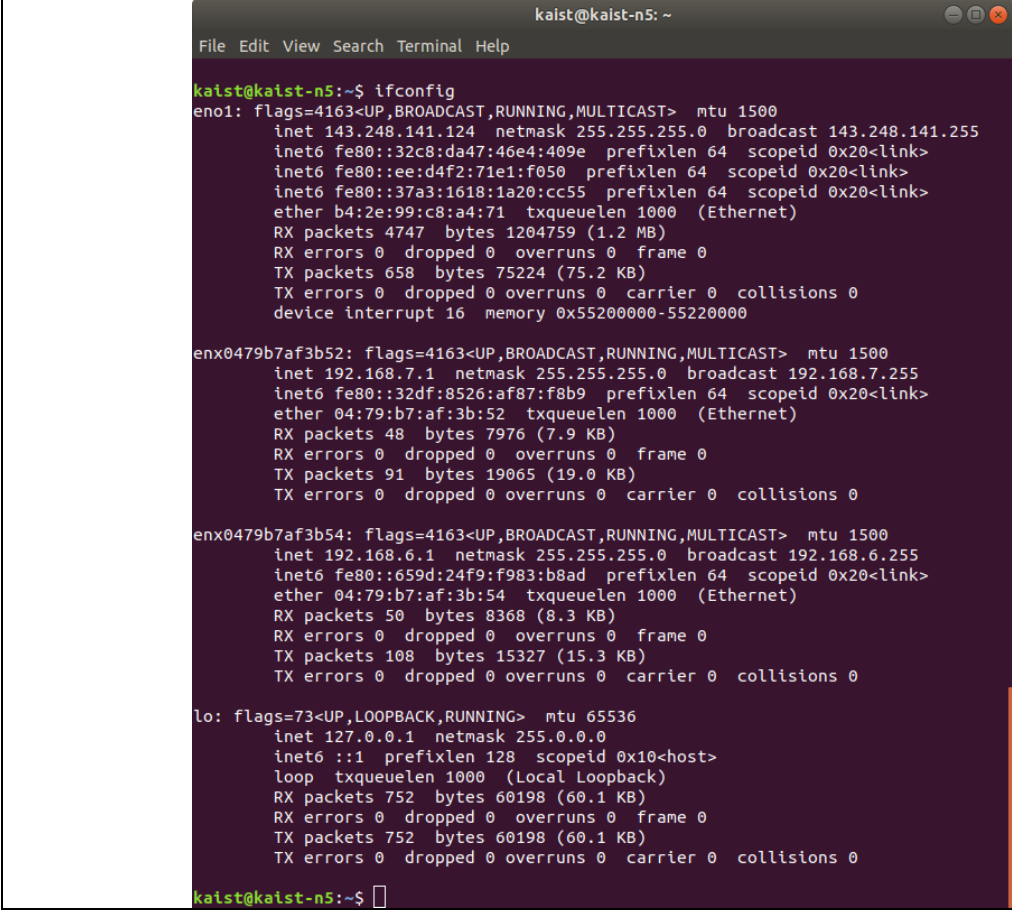


그림 2. Debian OS가 설치되어 있는 SD카드를 이용해 부팅한 Beaglebone Black

다시 한번 ifconfig 명령어를 이용하여 network connection을 확인해 본 결과 새로운 network adapter가 나타난 것을 확인할 수 있었다. Beaglebone black에 할당된 local network에 SSH(Secure Shell Command)를 이용하여 연결하였다.



```

kaist@kaist-n5: ~
File Edit View Search Terminal Help

kaist@kaist-n5:~$ ifconfig
eno1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 143.248.141.124 netmask 255.255.255.0 broadcast 143.248.141.255
    inet6 fe80::32c8:da47:46e4:409e prefixlen 64 scopeid 0x20<link>
    inet6 fe80::ee:d4f2:71e1:f050 prefixlen 64 scopeid 0x20<link>
    inet6 fe80::37a3:1618:1a20:cc55 prefixlen 64 scopeid 0x20<link>
    ether b4:2e:99:c8:a4:71 txqueuelen 1000 (Ethernet)
    RX packets 4747 bytes 1204759 (1.2 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 658 bytes 75224 (75.2 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
    device interrupt 16 memory 0x55200000-55220000

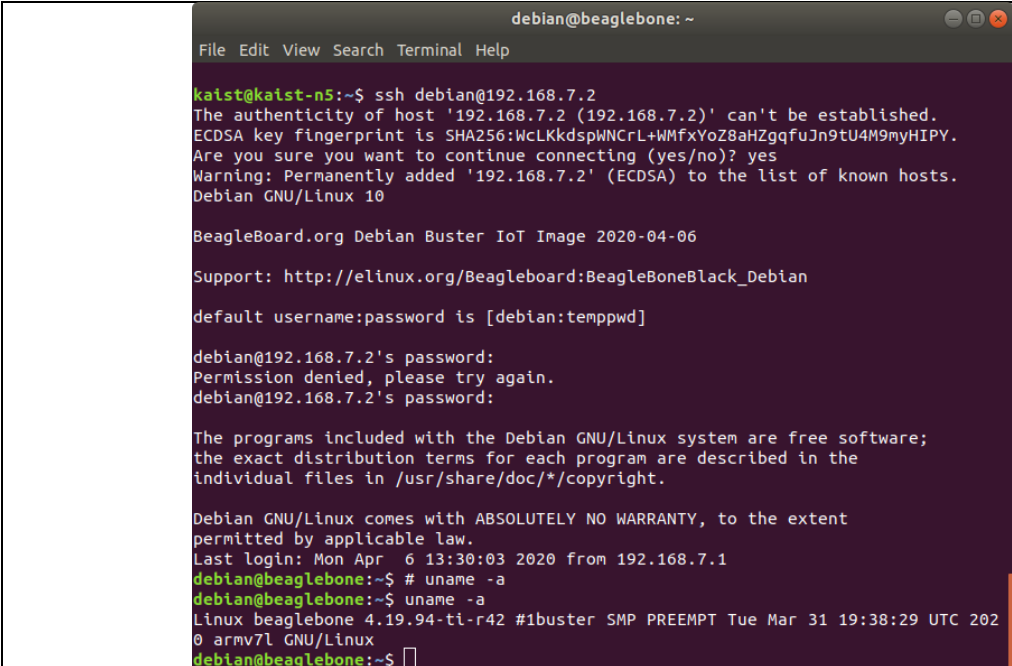
enx0479b7af3b52: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.7.1 netmask 255.255.255.0 broadcast 192.168.7.255
    inet6 fe80::32df:8526:af87:f8b9 prefixlen 64 scopeid 0x20<link>
    ether 04:79:b7:af:3b:52 txqueuelen 1000 (Ethernet)
    RX packets 48 bytes 7976 (7.9 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 91 bytes 19065 (19.0 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

enx0479b7af3b54: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.6.1 netmask 255.255.255.0 broadcast 192.168.6.255
    inet6 fe80::659d:24f9:f983:b8ad prefixlen 64 scopeid 0x20<link>
    ether 04:79:b7:af:3b:54 txqueuelen 1000 (Ethernet)
    RX packets 50 bytes 8368 (8.3 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 108 bytes 15327 (15.3 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 752 bytes 60198 (60.1 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 752 bytes 60198 (60.1 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

kaist@kaist-n5:~$
  
```

그림 3. Ifconfig 명령어를 통해 확인한 새로운 network adapter



```

debian@beaglebone: ~
File Edit View Search Terminal Help

kaist@kaist-n5:~$ ssh debian@192.168.7.2
The authenticity of host '192.168.7.2 (192.168.7.2)' can't be established.
ECDSA key fingerprint is SHA256:WcLKkdsphNCRl+WMfxYoZ8aHZgqfuJn9tU4M9myHIPY.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.7.2' (ECDSA) to the list of known hosts.
Debian GNU/Linux 10

BeagleBoard.org Debian Buster IoT Image 2020-04-06

Support: http://elinux.org/Beagleboard:BeagleBoneBlack_Debian

default username:password is [debian:tempwd]

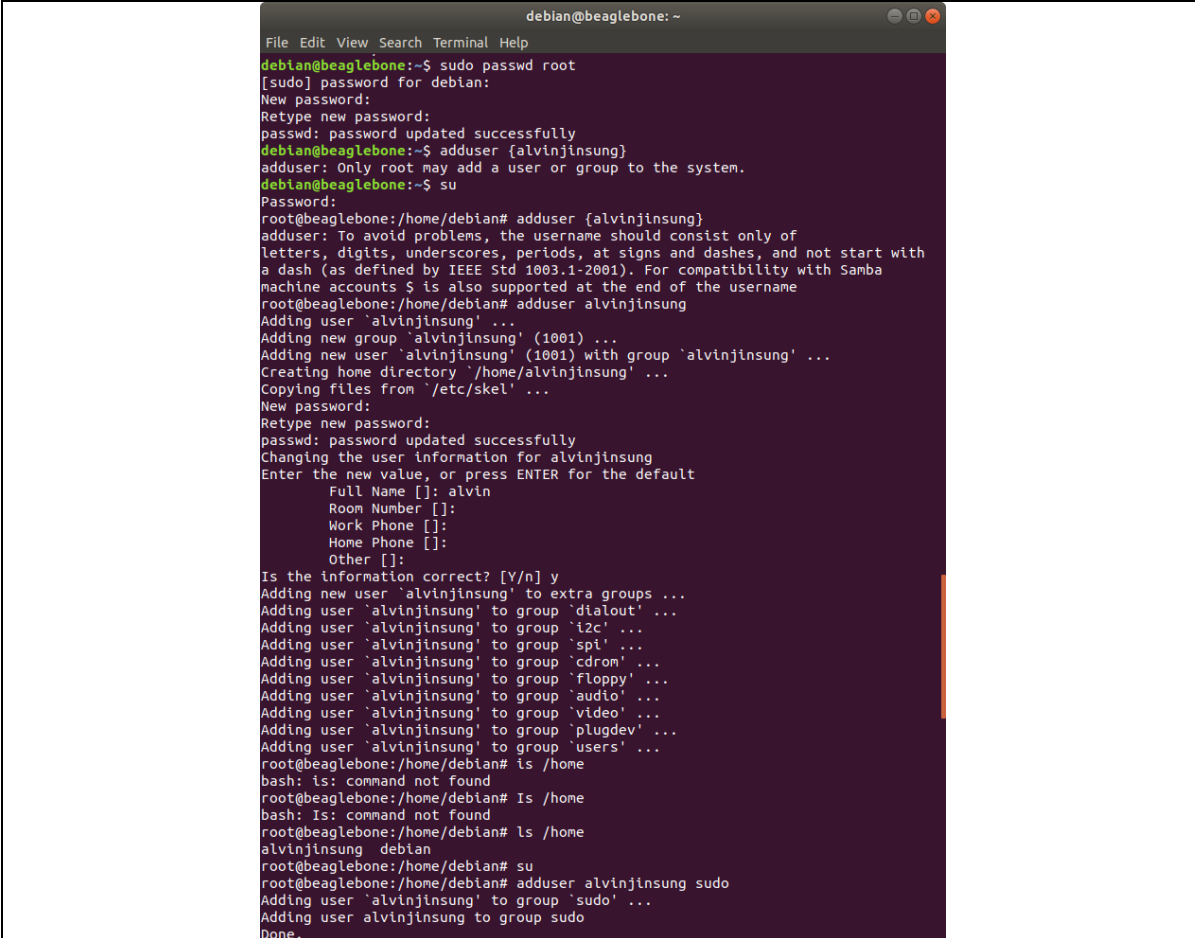
debian@192.168.7.2's password:
Permission denied, please try again.
debian@192.168.7.2's password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Apr  6 13:30:03 2020 from 192.168.7.1
debian@beaglebone:~$ # uname -a
debian@beaglebone:~$ uname -a
Linux beaglebone 4.19.94-ti-r42 #1buster SMP PREEMPT Tue Mar 31 19:38:29 UTC 2020
0 armv7l GNU/Linux
debian@beaglebone:~$
  
```

그림 4. SSH를 이용하여 Beaglebone black에 연결한 모습

Beaglebone debian에 superuser를 추가하였고 Beaglebone 상에서 network connection을 확인하여 ip를 정상적으로 할당 받았다는 것을 확인하였다.

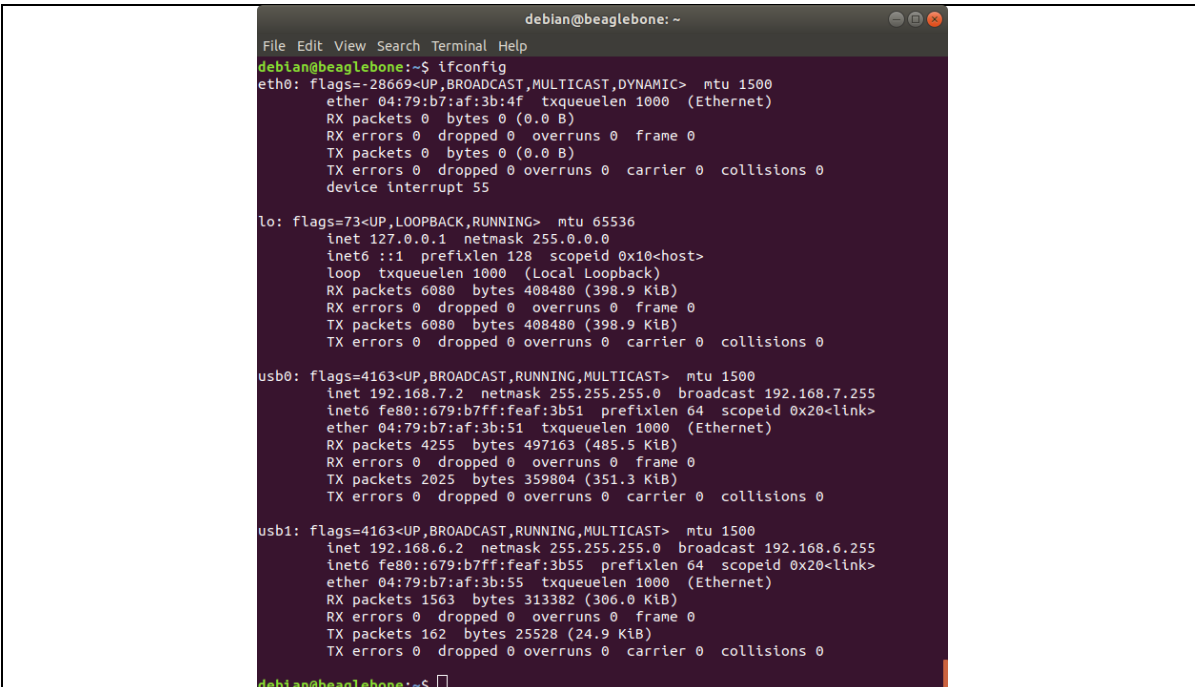


```

debian@beaglebone: ~
File Edit View Search Terminal Help
debian@beaglebone:~$ sudo passwd root
[sudo] password for debian:
New password:
Retype new password:
passwd: password updated successfully
debian@beaglebone:~$ adduser {alvinjinsung}
adduser: Only root may add a user or group to the system.
debian@beaglebone:~$ su
Password:
root@beaglebone:/home/debian# adduser {alvinjinsung}
adduser: To avoid problems, the username should consist only of
letters, digits, underscores, periods, at signs and dashes, and not start with
a dash (as defined by IEEE Std 1003.1-2001). For compatibility with Samba
machine accounts $ is also supported at the end of the username
root@beaglebone:/home/debian# adduser alvinjinsung
Adding user 'alvinjinsung' ...
Adding new group 'alvinjinsung' (1001) ...
Adding new user 'alvinjinsung' (1001) with group 'alvinjinsung' ...
Creating home directory '/home/alvinjinsung' ...
Copying files from '/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for alvinjinsung
Enter the new value, or press ENTER for the default
    Full Name []: alvin
    Room Number []:
    Work Phone []:
    Home Phone []:
    Other []:
Is the information correct? [Y/n] y
Adding new user 'alvinjinsung' to extra groups ...
Adding user 'alvinjinsung' to group 'dialout' ...
Adding user 'alvinjinsung' to group 'i2c' ...
Adding user 'alvinjinsung' to group 'spi' ...
Adding user 'alvinjinsung' to group 'cdrom' ...
Adding user 'alvinjinsung' to group 'floppy' ...
Adding user 'alvinjinsung' to group 'audio' ...
Adding user 'alvinjinsung' to group 'video' ...
Adding user 'alvinjinsung' to group 'plugdev' ...
Adding user 'alvinjinsung' to group 'users' ...
root@beaglebone:/home/debian# is /home
bash: is: command not found
root@beaglebone:/home/debian# Is /home
bash: Is: command not found
root@beaglebone:/home/debian# ls /home
alvinjinsung  debian
root@beaglebone:/home/debian# su
root@beaglebone:/home/debian# adduser alvinjinsung sudo
Adding user 'alvinjinsung' to group 'sudo' ...
Adding user alvinjinsung to group sudo
Done.

```

그림 5. Beaglebone debian에 새로운 superuser 추가



```

debian@beaglebone: ~
File Edit View Search Terminal Help
debian@beaglebone:~$ ifconfig
eth0: flags=28669<UP,BROADCAST,MULTICAST,DYNAMIC> mtu 1500
    ether 04:79:b7:af:3b:4f txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
    device interrupt 55

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 6080 bytes 408480 (398.9 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 6080 bytes 408480 (398.9 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

usb0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.7.2 netmask 255.255.255.0 broadcast 192.168.7.255
    inet6 fe80::679:b7ff:feaf:3b51 prefixlen 64 scopeid 0x20<link>
    ether 04:79:b7:af:3b:51 txqueuelen 1000 (Ethernet)
    RX packets 4255 bytes 497163 (485.5 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 2025 bytes 359804 (351.3 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

usb1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.6.2 netmask 255.255.255.0 broadcast 192.168.6.255
    inet6 fe80::679:b7ff:feaf:3b55 prefixlen 64 scopeid 0x20<link>
    ether 04:79:b7:af:3b:55 txqueuelen 1000 (Ethernet)
    RX packets 1563 bytes 313382 (306.0 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 162 bytes 25528 (24.9 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

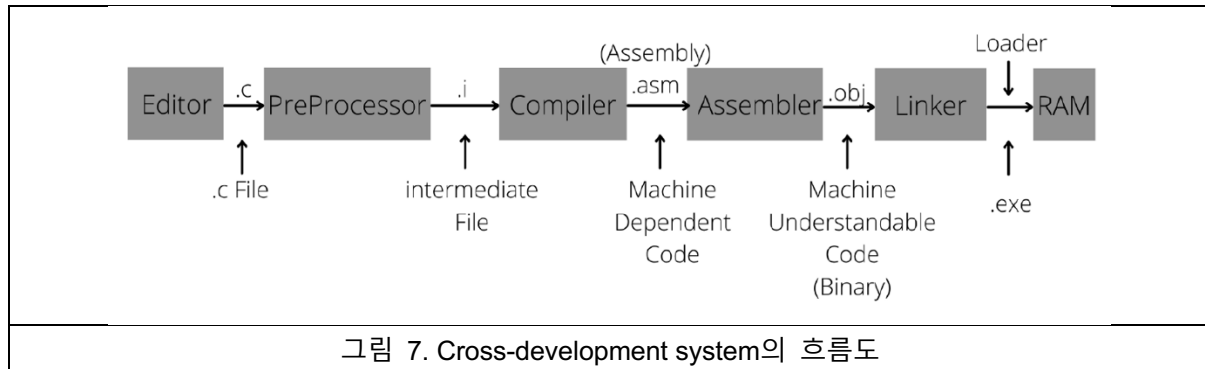
debian@beaglebone:~$

```

그림 6. Beaglebone에서 확인한 network connection

### Problem 1A. Cross-Development System

Intel 베이스의 Ubuntu PC 환경과 arm 베이스 Debian Beaglebone Black 환경의 cross-development system을 구축한다. Cross development system이란 target embedded board에서 실행할 software 개발을 위한 software development system을 의미한다. 적절한 cross-development software가 설치되고 활용되어야 하며, editor, compiler, linker, debugger 등을 포함한다. Compiler는 Pentium binary code가 아닌 우리의 target system에 해당하는 ARM cortex binary code를 생성한다. 전체적인 development process의 흐름은 아래와 같다.



Cross-development system 구축을 위해 우선 ARM에서 실행 가능한 cross-compiler를 PC에서 설치하였다.

```

kaist@kaist-n5: ~
File Edit View Search Terminal Help
kaist@kaist-n5:~$ arm-linux-gnueabi-gcc --version
arm-linux-gnueabi-gcc (Ubuntu/Linaro 7.5.0-3ubuntu1-18.04) 7.5.0
Copyright (C) 2017 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.
kaist@kaist-n5:~$
  
```

그림 8. PC에 설치된 cross-compiler

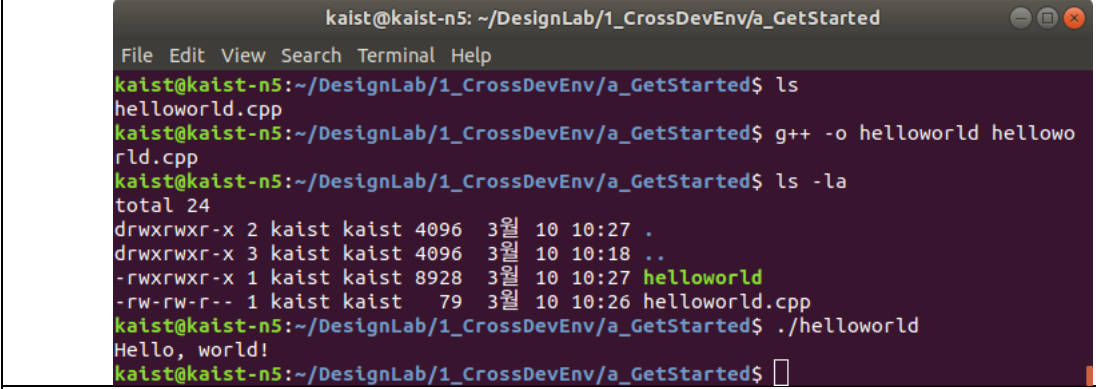
이후 cross-compiler를 통해 compile한 프로그램이 잘 실행되는지를 체크하기 위해 간단한 hello world 프로그램을 작성하여 실험해보았다. 우선 PC에서 g++를 통해 native-compile하여 PC에서 문제 없이 실행되는 것을 확인한 후 arm-linux-gnueabi-g++를 통해 cross-compile한 것이 우리의 target system인 Beaglebone Black에서 잘 실행되는 것을 확인하였다. Secure copy SCP를 이용하여 파일을 전송하였다.

```

helloworld.cpp
~/DesignLab/1_CrossDevEnv/a_GetStarted
#include <iostream>
void main()
{
    std::cout << "Hello, world!" << std::endl;
}
  
```

그림 9. helloworld.cpp 파일



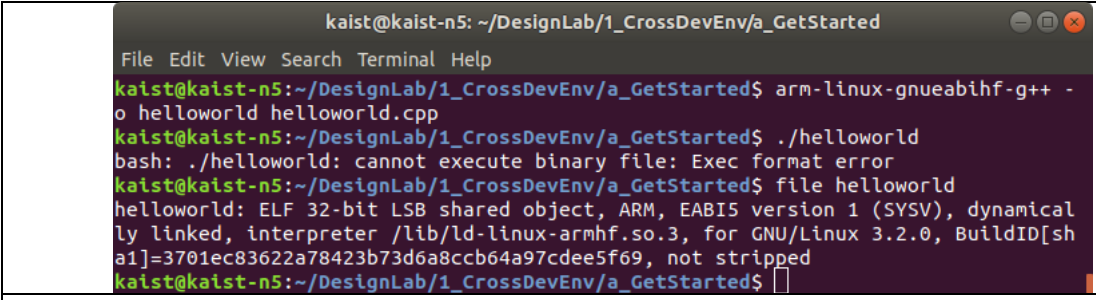


```

kaist@kaist-n5: ~/DesignLab/1_CrossDevEnv/a_GetStarted
File Edit View Search Terminal Help
kaist@kaist-n5:~/DesignLab/1_CrossDevEnv/a_GetStarted$ ls
helloworld.cpp
kaist@kaist-n5:~/DesignLab/1_CrossDevEnv/a_GetStarted$ g++ -o helloworld helloworld.cpp
kaist@kaist-n5:~/DesignLab/1_CrossDevEnv/a_GetStarted$ ls -la
total 24
drwxrwxr-x 2 kaist kaist 4096 3월 10 10:27 .
drwxrwxr-x 3 kaist kaist 4096 3월 10 10:18 ..
-rwxrwxr-x 1 kaist kaist 8928 3월 10 10:27 helloworld
-rw-rw-r-- 1 kaist kaist 79 3월 10 10:26 helloworld.cpp
kaist@kaist-n5:~/DesignLab/1_CrossDevEnv/a_GetStarted$ ./helloworld
Hello, world!
kaist@kaist-n5:~/DesignLab/1_CrossDevEnv/a_GetStarted$

```

그림 10. g++를 이용해 PC에서 native-compile한 후 PC에서 실행한 모습

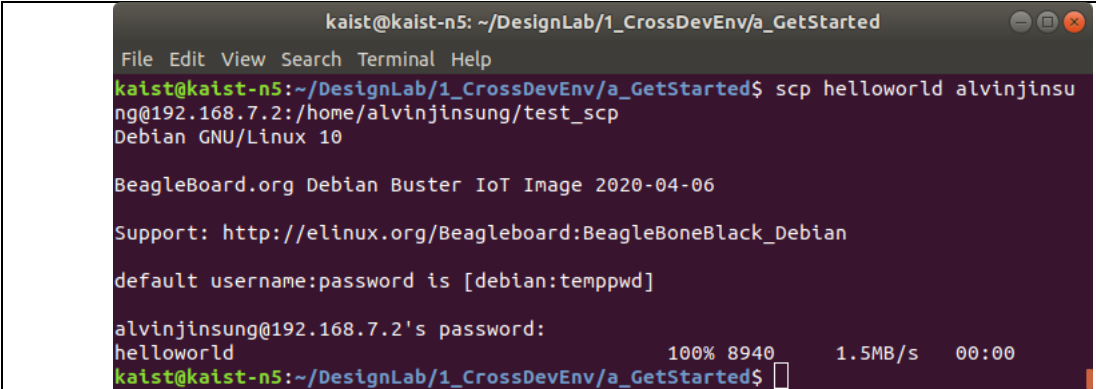


```

kaist@kaist-n5: ~/DesignLab/1_CrossDevEnv/a_GetStarted
File Edit View Search Terminal Help
kaist@kaist-n5:~/DesignLab/1_CrossDevEnv/a_GetStarted$ arm-linux-gnueabi-g++ -o helloworld helloworld.cpp
kaist@kaist-n5:~/DesignLab/1_CrossDevEnv/a_GetStarted$ ./helloworld
bash: ./helloworld: cannot execute binary file: Exec format error
kaist@kaist-n5:~/DesignLab/1_CrossDevEnv/a_GetStarted$ file helloworld
helloworld: ELF 32-bit LSB shared object, ARM, EABI5 version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux-armhf.so.3, for GNU/Linux 3.2.0, BuildID[sha1]=3701ec83622a78423b73d6a8ccb64a97cdee5f69, not stripped
kaist@kaist-n5:~/DesignLab/1_CrossDevEnv/a_GetStarted$

```

그림 11. arm-linux-gnueabi-g++를 이용해 PC에서 cross-compile한 후 PC에서 실행한 모습 (ARM-CPU에서 실행되도록 cross-compile했기에 오류나는 것이 정상)



```

kaist@kaist-n5: ~/DesignLab/1_CrossDevEnv/a_GetStarted
File Edit View Search Terminal Help
kaist@kaist-n5:~/DesignLab/1_CrossDevEnv/a_GetStarted$ scp helloworld alvinjinsung@192.168.7.2:/home/alvinjinsung/test_scp
Debian GNU/Linux 10

BeagleBoard.org Debian Buster IoT Image 2020-04-06

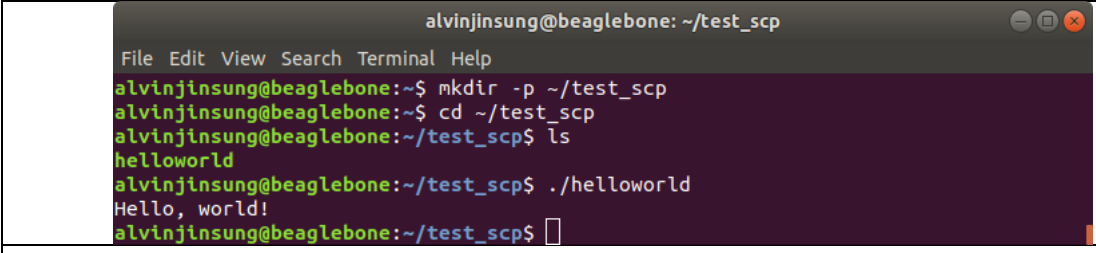
Support: http://elinux.org/Beagleboard:BeagleBoneBlack_Debian

default username:password is [debian:tempwd]

alvinjinsung@192.168.7.2's password:
helloworld 100% 8940 1.5MB/s 00:00
kaist@kaist-n5:~/DesignLab/1_CrossDevEnv/a_GetStarted$

```

그림 12. Cross-compile한 실행 파일을 scp를 이용해 PC에서 Beaglebone Black으로 전송



```

alvinjinsung@beaglebone: ~/test_scp
File Edit View Search Terminal Help
alvinjinsung@beaglebone:~$ mkdir -p ~/test_scp
alvinjinsung@beaglebone:~$ cd ~/test_scp
alvinjinsung@beaglebone:~/test_scp$ ls
helloworld
alvinjinsung@beaglebone:~/test_scp$ ./helloworld
Hello, world!
alvinjinsung@beaglebone:~/test_scp$

```

그림 13. Cross-compile된 파일이 Beaglebone Black에서 정상적으로 작동하는 모습

이후 이 cross-development system을 더욱 편하게 사용하기 위해 NFS를 설치 및 작동시켰다. NFS란 distributed file system protocol로 client computer의 user가 네트워크 상에 있는 파일들에 액세스할 수 있게 해준다. 앞의 예시와 같이 scp를 이용하여 매번 파일을 옮길 필요없이 client

computer에서 nfs로 연결된 네트워크 상의 파일들 바로 확인 가능하고 활용할 수 있게 해주는 것이다. 이를 위해 PC에 nfs server를 설치하고 nano 명령어를 활용하여 /etc/exports에서 연결 가능한 hosts를 추가해주었다. 본 실험에서는 Beaglebone Black이 할당 받은 IP 주소를 추가하였다. 변경된 /etc/exports를 반영하기 위해 sudo exportfs -a 커맨드를 실행하였으며 PC NFS server를 실행하였다.

```

kaist@kaist-n5: ~/DesignLab/1_CrossDevEnv/a_GetStarted
File Edit View Search Terminal Help
GNU nano 2.9.3 /etc/exports
# /etc/exports: the access control list for filesystems which may be exported
# to NFS clients.  See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes      hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_sub$
#
# Example for NFSv4:
# /srv/nfs4       gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
# /srv/nfs4/homes gss/krb5i(rw,sync,no_subtree_check)
#
# nfs for Bone Ubuntu - Robot Manipulator
/home/kaist 192.168.7.2(rw,sync,no_root_squash,no_subtree_check)
  
```

그림 14. 수정된 /etc/exports 파일(192.168.7.2: Beaglebone Black의 IP 주소)

```

kaist@kaist-n5: ~/DesignLab/1_CrossDevEnv/a_GetStarted
File Edit View Search Terminal Help
kaist@kaist-n5:~/DesignLab/1_CrossDevEnv/a_GetStarted$ sudo /etc/init.d/nfs-kern
el-server start
[ ok ] Starting nfs-kernel-server (via systemctl): nfs-kernel-server.service.
kaist@kaist-n5:~/DesignLab/1_CrossDevEnv/a_GetStarted$
  
```

그림 15. PC NFS server를 실행한 모습

이 다음 Beaglebone Black에는 nfs client를 설치해주었다. Beaglebone Black의 mount point를 지정했으며 sudo mount {pc\_ip}:{pc\_directory} {beaglebone\_directory} 명령어를 이용해 nfs client를 실행시켰다. 이후 편의를 위해 ~/bone\_nfs\_client.sh를 nano 명령어를 사용해 편집하여 실행시켰다.

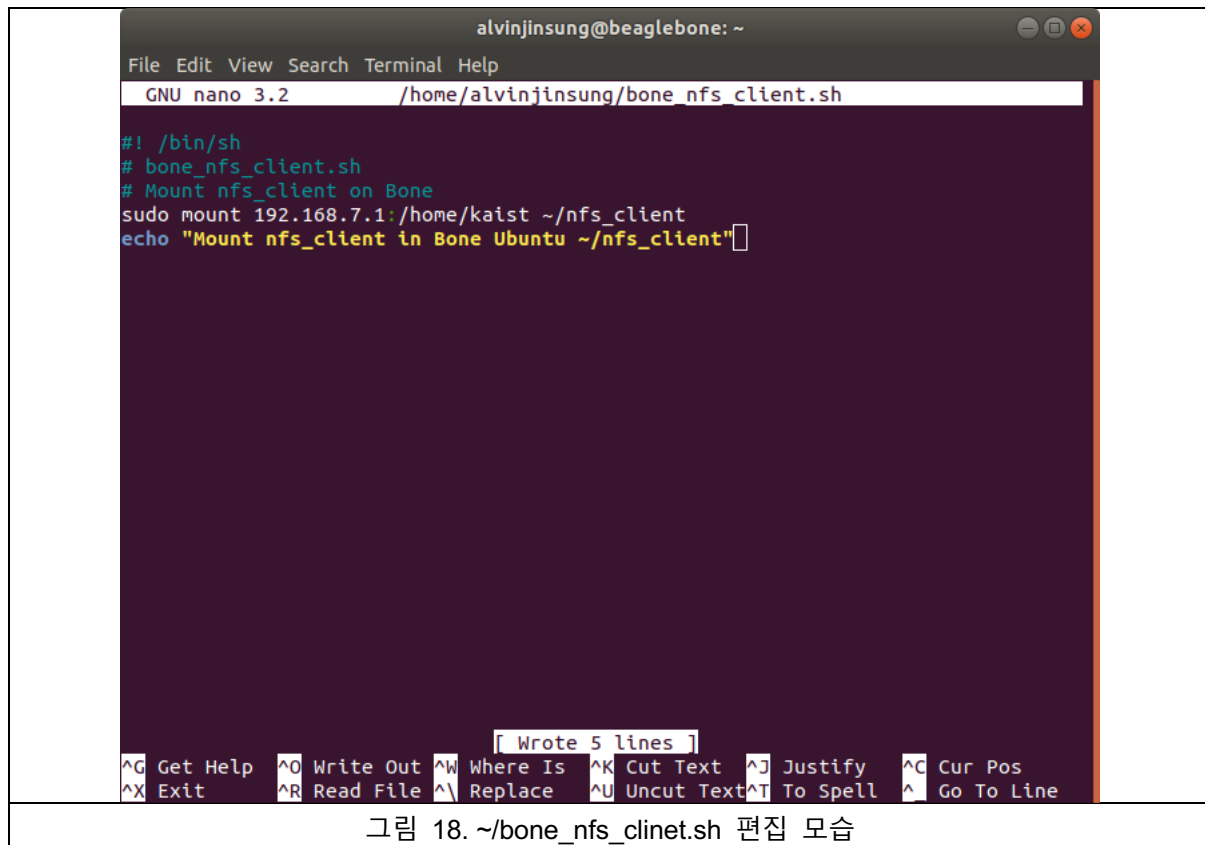
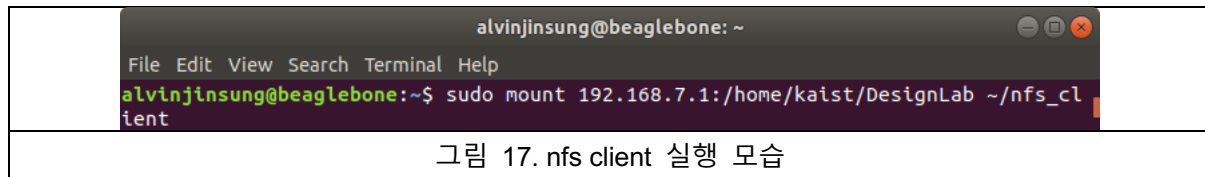
```

alvinjinsung@beaglebone: ~
File Edit View Search Terminal Help
alvinjinsung@beaglebone:~$ sudo apt-get install nfs-common
We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

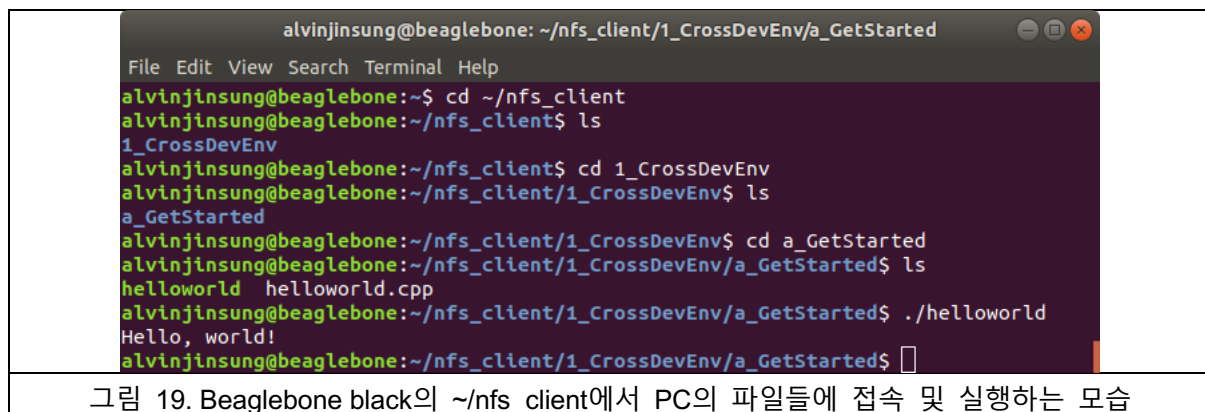
#1) Respect the privacy of others.
#2) Think before you type.
#3) With great power comes great responsibility.

[sudo] password for alvinjinsung:
Reading package lists... Done
Building dependency tree
Reading state information... Done
nfs-common is already the newest version (1:1.3.4-2.5+deb10u1).
0 upgraded, 0 newly installed, 0 to remove and 5 not upgraded.
alvinjinsung@beaglebone:~$
  
```

그림 16. Beaglebone Black nfs client 설치 모습



이제 Beaglebone Black의 ~/nfs\_client 내에서 nfs로 연결된 PC의 파일들을 볼 수 있는 것을 확인하였으며 cross-compile 되었던 helloworld 프로그램 역시 정상적으로 실행시킬 수 있는 것을 확인하였다.



위의 전 과정을 통해 성공적으로 Cross-development system을 구축하였고 NFS를 이용해 효율적으로 이를 활용 가능하게 만들었다.



**Problem 1B. Exercise1: Time looped program**

Chrono library를 활용해 time looped program을 작성하고 이를 native-compile과 cross-compile하여 PC와 Beaglebone Black에서 각각 실행시켜본다. <Chrono>는 C++ library로서 시간에 관한 다양한 함수를 제공한다. 현재 시간을 할당 받는 기본적인 함수는 아래와 같다.

```
std::chrono::system_clock::time_point now; // make a time point variable "now"
now = std::chrono::system_clock::now(); // assign current time into the variable
```

이 library를 활용하여 아래를 만족하는 counter program을 C++를 이용해 작성하였다.

## &lt;요구 조건&gt;

1. make a time point "prev" and save current time
2. in a while loop,
  - a. calculate elapsed time between current time and prev
  - b. if elapsed time  $\geq 1$  sec, print out counter and assign current time as prev
  - c. if counter is less than 0, break

```
int main(int argc, char *argv[])
{
    std::chrono::system_clock::time_point curr, prev;
    prev = std::chrono::system_clock::now();

    //string start_message = "Countdown !";
    //string end_message = "End";

    std::cout << "Countdown !" << std::endl; //start_message;

    int counter = 10;

    while (counter >= 0)
    {
        curr = std::chrono::system_clock::now();

        auto elapsed_time = std::chrono::duration_cast<std::chrono::microseconds>(curr - prev);
        if (elapsed_time.count() >= 1000000)
        {
            std::cout << counter << std::endl;
            counter = counter - 1;
            prev = curr;
        }
    }

    std::cout << "End" << std::endl; //end_message;

    return 0;
}
```

C++ Tab Width: 8 Ln 14, Col 1 INS

그림 20. C++를 이용해 작성한 counter program

위 코드를 PC에서 native-compile과 cross-compile하여 PC와 Beaglebone Black에서 각각 실행시켰다. 정상적으로 작동하는 것을 확인하였다.

```
kaist@kaist-n5: ~/DesignLab/20170699/1_CrossDevEnv
File Edit View Search Terminal Help
kaist@kaist-n5:~/DesignLab/20170699/1_CrossDevEnv$ ls
b_counter b_counter.cpp
kaist@kaist-n5:~/DesignLab/20170699/1_CrossDevEnv$ g++ -o b_counter b_counter.cpp
kaist@kaist-n5:~/DesignLab/20170699/1_CrossDevEnv$ ./b_counter
Countdown !
10
9
8
7
6
5
4
3
2
1
0
End
kaist@kaist-n5:~/DesignLab/20170699/1_CrossDevEnv$
```

그림 21. PC에서 native-compile 한 후 PC에서 실행한 모습

```

kaist@kaist-n5: ~/DesignLab/20170699/1_CrossDevEnv
File Edit View Search Terminal Help
kaist@kaist-n5:~/DesignLab/20170699/1_CrossDevEnv$ arm-linux-gnueabi-g++ -o b_
counter b_counter.cpp
kaist@kaist-n5:~/DesignLab/20170699/1_CrossDevEnv$ ./b_counter
bash: ./b_counter: cannot execute binary file: Exec format error
kaist@kaist-n5:~/DesignLab/20170699/1_CrossDevEnv$

```

그림 22. PC에서 cross-compile 한 후 PC에서 실행한 모습

```

alvinjinsung@beaglebone: ~/nfs_client/DesignLab/20170699/1_CrossDevEnv
File Edit View Search Terminal Help
alvinjinsung@beaglebone:~/nfs_client/DesignLab/20170699/1_CrossDevEnv$ ./b_count
er
Countdown !
10
9
8
7
6
5
4
3
2
1
0
End
alvinjinsung@beaglebone:~/nfs_client/DesignLab/20170699/1_CrossDevEnv$

```

그림 23. PC에서 cross-compile 한 후 Beaglebone Black에서 실행한 모습

### Problem 1C. Exercise2: User Input program

VSCode와 CMake를 이용한 개발 환경을 구축하고 debugger 사용법을 익힌다. 이를 이용해 user input을 인식하는 프로그램을 작성하고 PC와 Beaglebone Black에서 각각 실행해본다. 우선 VSCode를 설치하고 C/C++, C/C++ extension pack, CMake, CMake Tools를 Extensions에서 다운받았다. 그리고 CMake를 `sudo apt install cmake` 명령어를 활용하여 설치하였다. CMake는 cross-platform build management tool로 software project들을 각기 다른 platform과 compiler로 compile하고 linking해주는 과정을 자동화 시켜준다.

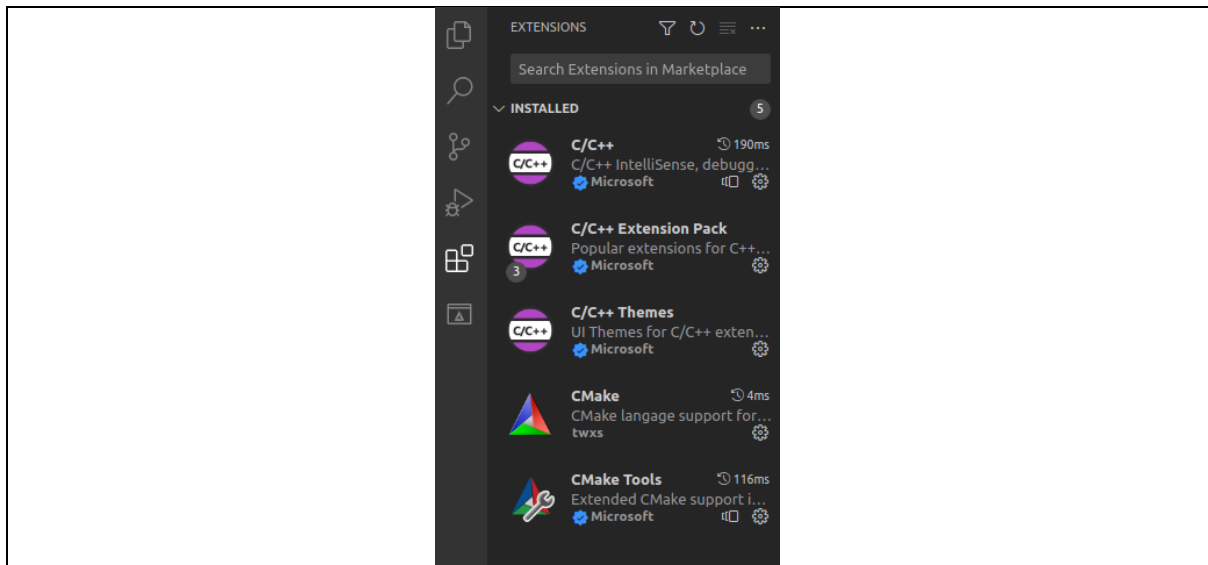
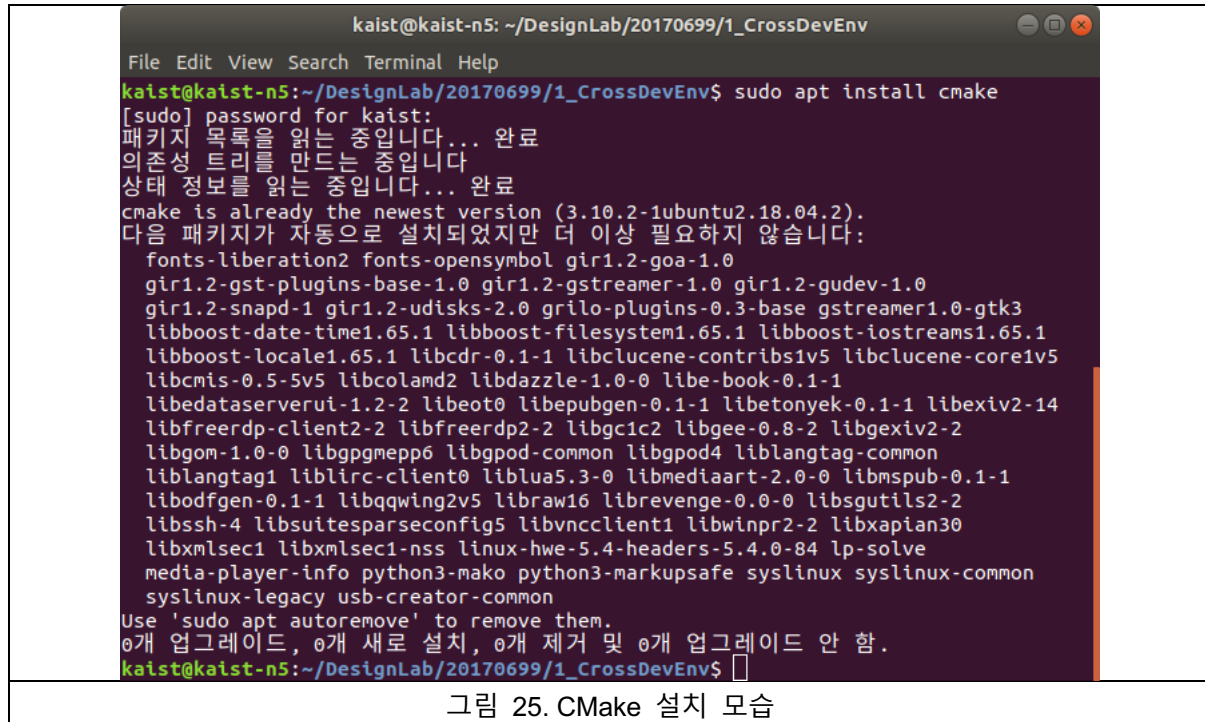


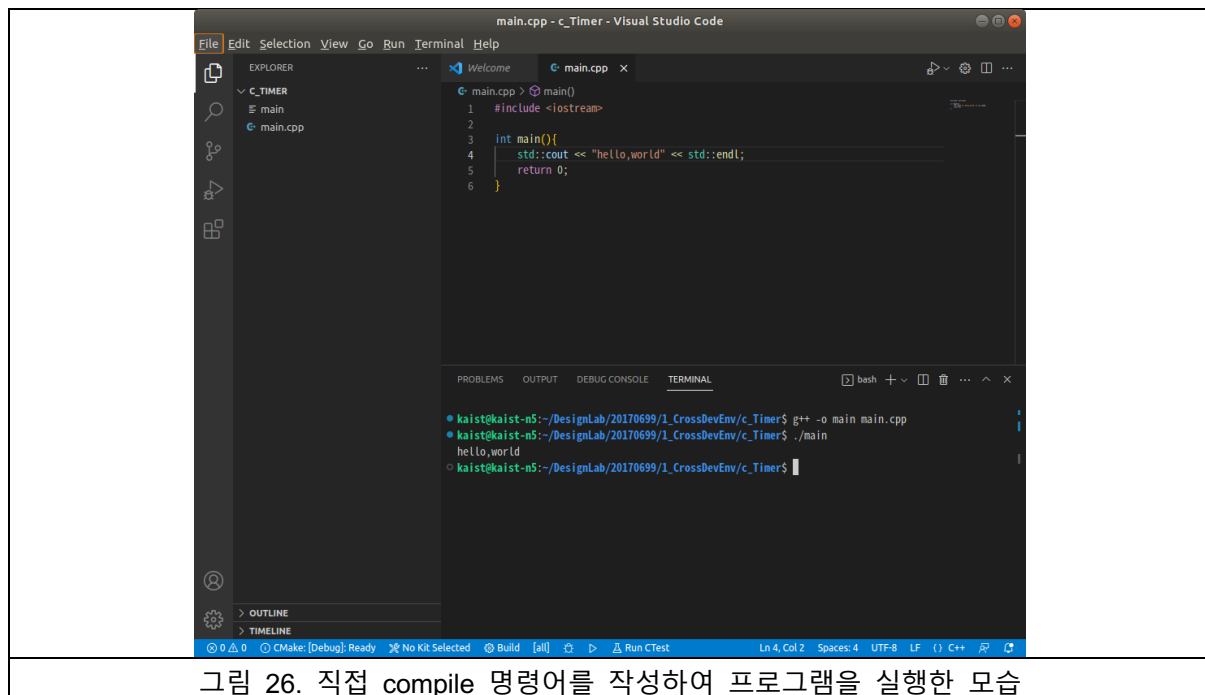
그림 24. VSCode 설치 후 C/C++, C/C++ extension pack, CMake, CMake Tools를 설치한 모습



CMake: Quick Start 선택 및 project 이름을 입력하고 executable을 선택하면 세개의 아이템이 자동으로 생성된다.

- Build(folder): Compile을 위한 파일들의 temporary storage place로 executable file 역시 이 위치에 생성된다.
- CMakeLists.txt: build process를 자동화 시켜주는 macro 파일이다.
- main.cpp: 실행 코드 파일이다.

간단한 hello world 프로그램을 일반적인 compile과정과 CMake를 이용한 과정을 통해 각각 실행시켜 보았다. 이를 통해 CMake 작동 방식을 이해하였다.



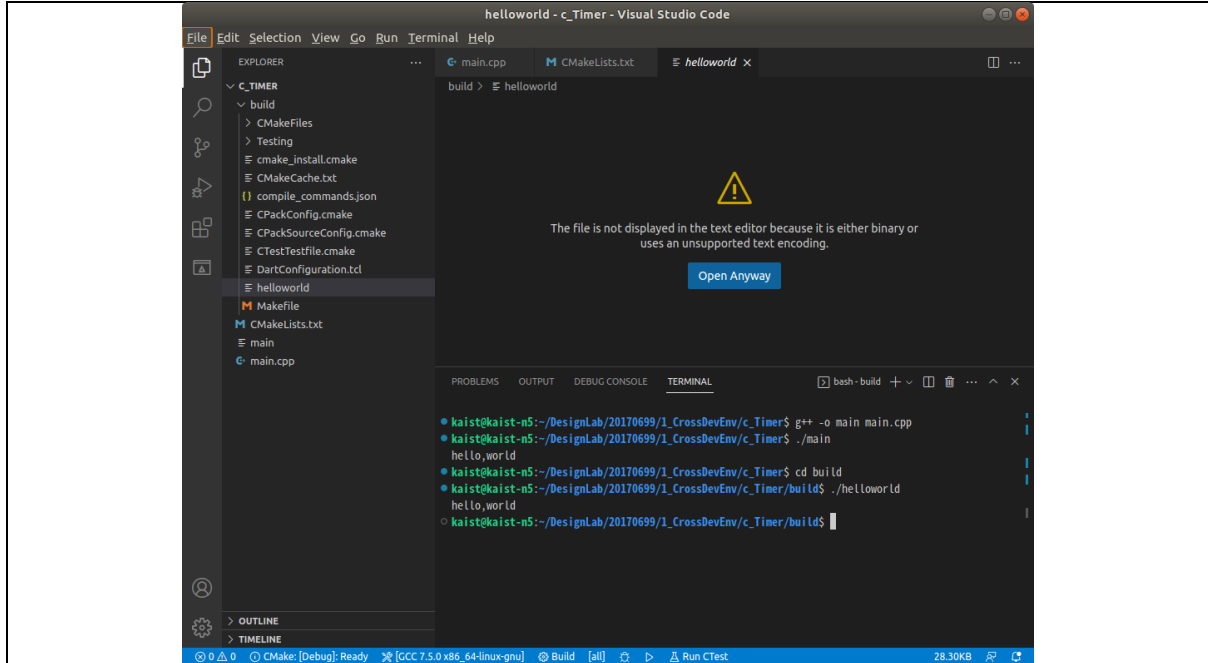


그림 27. CMake를 이용하여 프로그램을 실행한 모습

본 문제에서 요구한 프로그램을 작성하기 위해 getche.cpp와 getche.h 파일을 프로젝트에 포함시켜 간단한 예제 코드를 실행해보았다. getch() 함수와 getche() 함수는 키보드 입력값을 변수에 저장하는 역할을 한다. 정상적으로 프로그램이 작동하기 위해 CMakeLists.txt 파일에 변경된 file structure를 반영하였다.

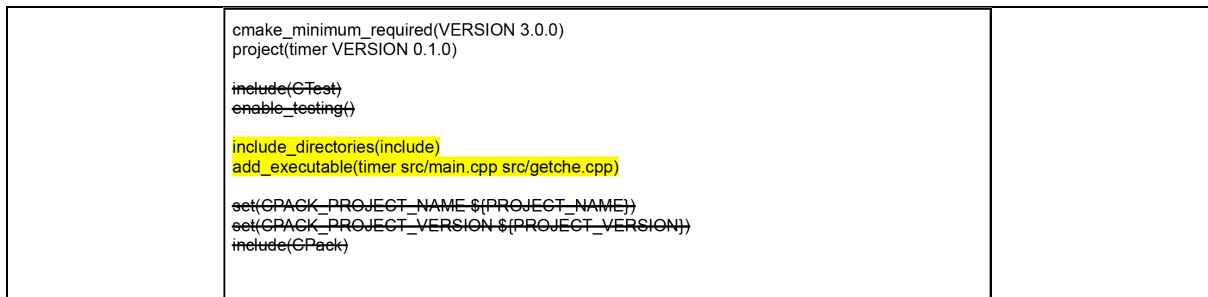


그림 28. CMakeLists.txt 변경 모습

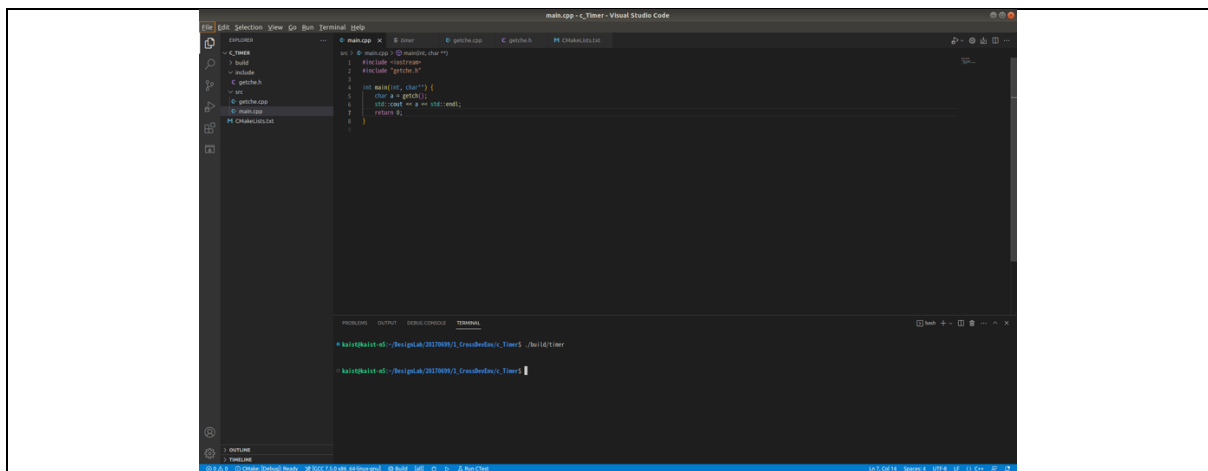


그림 29. 예제 코드 실행 모습

본 문제에서는 getch() 함수를 활용한 프로그램을 요구한다. 's' 입력을 받았을 시 입력까지 걸린 시간을 출력하고, 다른 키보드 입력을 받았을 시 'wrong input'이라는 메시지를 출력 후 새로운 입력을 기다리는 프로그램을 짠다. 구체적으로 작동해야 하는 모습은 다음과 같다.

Start! Press 's' to stop	
f: Wrong input	//if wrong input(f), print this
q: Wrong input	//if wrong input(q), print this
Elapsed time(sec): 4.8924	//if s, print this and elapsed time in second (float)

그림 30. User Input Program

이를 C++를 활용하여 작성하였으며 PC와 Beaglebone Black에서 각각 실행하였다. 실행 결과는 아래와 같다.

```

main.cpp - c_Timer - Visual Studio Code
src > main.cpp > main(int, char**)
1 #include <iostream>
2 #include <chrono>
3 #include "getche.h"
4
5 int main(int, char**) {
6     std::chrono::system_clock::time_point start, end;
7
8     std::cout << "Start! Press 's' to stop" << std::endl;
9
10    start = std::chrono::system_clock::now();
11
12    while(1){
13
14        char keyboard_input = getch();
15
16        if (keyboard_input == 's'){
17            break;
18            std::cout << "f: Wrong input" << std::endl;
19        }
20
21        else {
22            std::cout << keyboard_input << ": Wrong input" << std::endl;
23        }
24    }
25
26    end = std::chrono::system_clock::now();
27    auto elapsed_us = std::chrono::duration_cast<std::chrono::microseconds>(end - start);
28
29    std::cout << std::fixed;
30    std::cout << std::precision(6);
31
32    std::cout << "Elapsed time(sec):" << (float)elapsed_us.count()/1000000 << std::endl;
33    return 0;
34 }
35
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
* kaist@kaist-n5:~/DesignLab/20170699/1_CrossDevEnv/c_Timer$ ./build/timer
Start! Press 's' to stop
f: Wrong input
q: Wrong input
Elapsed time(sec): 7.783301

```

그림 31. User Input program 코드 및 PC에서 실행 모습

```

alvinjinsung@beaglebone: ~/nfs_client/DesignLab/20170699/1_CrossDevEnv
File Edit View Search Terminal Help
alvinjinsung@beaglebone:~/nfs_client/DesignLab/20170699/1_CrossDevEnv$ ls
b_counter b_counter.cpp c_Timer
alvinjinsung@beaglebone:~/nfs_client/DesignLab/20170699/1_CrossDevEnv$ ./c_Timer
/build/timer
Start! Press 's' to stop
f: Wrong input
q: Wrong input
Elapsed time(sec): 6.702511
alvinjinsung@beaglebone:~/nfs_client/DesignLab/20170699/1_CrossDevEnv$

```

그림 32. Cross-compile을 통해 Beaglebone Black에서 실행한 user input program 모습

문제에서 요구하는 조건을 모두 만족하는 코드를 작성하였으며 VSCode와 CMake를 이용한 개발 및 cross-compiling을 통한 target system에서의 실행까지 모두 성공하였다.

## 4. Discussion

### (1) What is the difference between the development system and the target system?

Development system은 개발자들이 software를 개발, 변형, 그리고 test하기 위해 사용하는 환경을 의미한다. 넓은 범위의 development tools, libraries, software dependencies가 설치되어 있으며 local한 머신인 경우가 많다. 이러한 특성으로 인해 개발자들이 주로 코드를 작성 및 compile하고 테스트와 디버깅을 하는 환경이다. 개발자의 효율성 극대화를 가장 큰 목적으로 한다. 실제 software가 deploy되는 환경을 고려하지 않는다.

Target system은 반대로 software가 결과적으로 deploy되고 실행되는 환경을 의미한다. single machine일수도, cloud-based 환경일수도 있으며 end-user가 software와 실제로 interact하는 환경이며. 해당 시스템은 performance 극대화, scalability, security가 주 목적으로 development system과 다른 하드웨어, OS, network configuration등을 가질 수 있다.

### (2) Is it possible to native-compile on BeagleBone? Describe the advantages and disadvantages of native-compiling and cross-compiling.

Beaglebone에서 native-compile하는 것은 가능하다. Native-compiling은 실제 코드가 실행되는 architecture와 platform이 동일한 환경에서 compile하는 것을 의미하는데, Beaglebone은 general-purpose embedded board로서 native-compile이 가능하다.

Native-compiling의 장점은 아래와 같다.

- 쉬운 사용: 새로운 setup이나 tool 설치 등이 필요하지 않다.
- Optimization: 최종 실행 환경과 개발 환경이 동일하기에 특정 hardware에 optimize를 더욱 효율적으로 할 수 있다.
- 즉각적인 feedback: native-compiling은 개발자들이 target device에서 바로 test를 해볼 수 있기에 즉각적인 feedback이 가능하다.

반대로 cross-compiling의 단점은 additional setup이 필요하며 target system과의 dependency와 compatibility를 체크해야 하고 optimization에 한계가 있다,

Native-compiling의 단점은 아래와 같다.

- limited portability: 개발환경에만 compatible하기에 새로운 platform이나 device에서 실행하기가 어렵다.
- 긴 compile 시간: cross-compiling에 비해 target device에서의 compiling이 오래 걸린다.
- damage 위험성: 지속적으로 동일한 device에서 compile 및 실행을 할 시 hardware에 damage가 올 수 있다.

반대로 cross-compiling의 장점은 portability가 높고, compile time이 짧으며, damage 위험성이 낮다.

### (3) Can the timed loop in the countdown program ensure a pre-defined time interval (hard real-time)? If not, what is the reason and what are the ways to ensure it?

Countdown program 내의 timed loop는 hard-real time을 보장하지 못 할 수도 있다. Process scheduling, system load, hardware interrupt 등 다양한 이유가 있을 수 있으며 이를 해결하기 위해서는 특별한 방법들을 사용해야 한다. Low-level language를 사용해 원인이 최소화되도록 할 수 있으며 RTOS(Real-Time Operating System)나 deterministic timing을 support하는 microcontroller에서 실행하는 방법도 있다. 또는, hardware timer나 interrupt를 활용해 timed loop를 트리거하는 방법이



있다. 이와 같이 hard-real time을 보장하기 위해서는 system에 대한 이해를 바탕으로 세밀한 조정이 필요하다.

**(4) Verifying your code is very important for developing programs rapidly. What methods did you use to get your code to work on the BeagleBone, and are there any alternatives?**

우선 코드가 정상적으로 작동하는지 확인하기 위해 PC에서 native-compile을 통해 코드가 제대로 실행되는지 확인 후 cross-compile하여 Beaglebone에서 실행시켰다. 코드 작성 과정에서는 unit testing과 integration testing, functional testing을 이용해 코드를 작성하였으며 debug과정에서 VSCode의 debugger를 활용, 특정 시점에 변수가 원하는 값을 가지고 있는지 확인하였다. 최종적으로는 PC에서 테스트 해 본 후 target system인 Beaglebone Black에서 정상적으로 작동하는지 확인하였다. 이 외에도 continuous integration, test-driven development 등 코드를 verify하는 방법들이 존재한다.

**(5) Choose your own discussion topic related to Lab 1 and provide an answer to it.**

Q. Why is optimizing code for embedded systems is important and provide some ways to do it.

A. Beaglebone Black과 같은 embedded system의 경우 processing power와 memory 등이 제한적이기에 code optimization은 굉장히 중요하다. 이를 위해서는 다양한 방법들이 있는데 아래에 몇가지를 소개한다.

- memory allocation 최소화: dynamic memory allocation은 embedded system의 경우 느리고 비효율적일 수 있으므로 그 사용을 줄이고 stack-based variables를 사용, recursive function 사용 자제, pro-allocate memory 사용을 해야 한다.
- inline function 사용: Inline function은 function call overhead를 줄이고 performance를 높일 수 있다. 다만, code size 증가와 cache miss 증가로 이어질 수 있으므로 적절한 사용이 필요하다.
- 효율적인 data structure 선택

## 5. References

- [1] Lab1 Experiment Guide.pdf
- [2] Lab1 Procedure.pdf
- [3] Lab1 TA Sessions.pdf
- [4] Lab1\_supp\_video\_vscode\_cmake\_debugging.mp4
- [5] [https://en.wikipedia.org/wiki/Operating\\_system](https://en.wikipedia.org/wiki/Operating_system)
- [6] [https://en.wikipedia.org/wiki/Secure\\_Shell](https://en.wikipedia.org/wiki/Secure_Shell)
- [7] [http://en.wikipedia.org/wiki/Network\\_File\\_System](http://en.wikipedia.org/wiki/Network_File_System)
- [8] Difference between Native compiler and Cross compiler,  
<https://www.geeksforgeeks.org/difference-between-native-compiler-and-cross-compiler/>
- [9] [https://en.wikipedia.org/wiki/Real-time\\_computing](https://en.wikipedia.org/wiki/Real-time_computing)
- [10] What is Real-Time Operating System(RTOS)?,  
<https://www.ni.com/ko-kr/shop/data-acquisition-and-control/add-ons-for-data-acquisition-and-control/what-is-labview-real-time-module/what-is-a-real-time-operating-system--rtos--.html>
- [11] Code Verification Techniques in Software Engineering,  
<https://ecomputernotes.com/software-engineering/code-verification-techniques>
- [12] Verification Methods in Software Verification,  
<https://www.geeksforgeeks.org/verification-methods-in-software-verification/>