

임베디드시스템 설계 및 실습 - 보고서

제출일: 2021-03-24; Lab 1

이름: 박관영(2016124099), 오한별(2018124120), 김서영(2017106007); 4조

✓ Assignment 1

개요

Linux 기반으로 동작하는 HPS를 사용하기 위하여 ubuntu를 modifying 하여 만든 linux image를 board에 booting 하는 실험을 수행한다. 이 때 micro SD 카드에 image를 write하고 PuTTY를 통해 board를 booting 할 수 있게 된다. 다음 과정을 통해 board의 HPS에서 linux 환경을 구축하고 기본 C 프로그램을 동작하는 실험을 진행하게 된다.

진행 내용

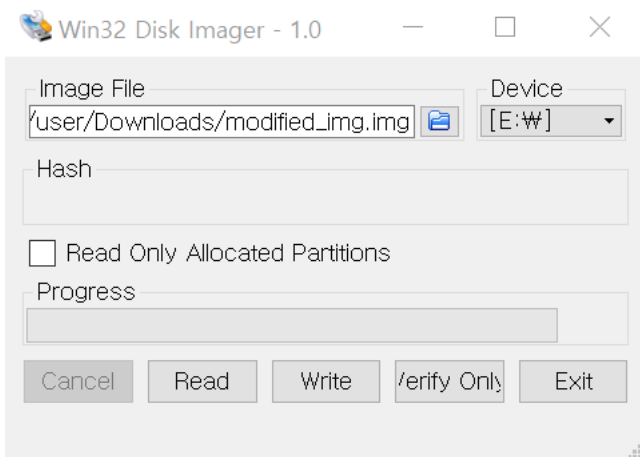
```
#include<stdio.h>
Int main(void){
    Printf("Hello World"); // hello world 를 출력하는 기본 program
    Return 0;
}
```

진행 결과

1. Linux image download

 modified_img.img	2021-03-15 오후 8:31	디스크 이미지 파일	7,761,920KB
---	--------------------	------------	-------------

2. Linux image write to micro SD card



→ Win32DiskImager program을 통해 수행할 수 있었다.

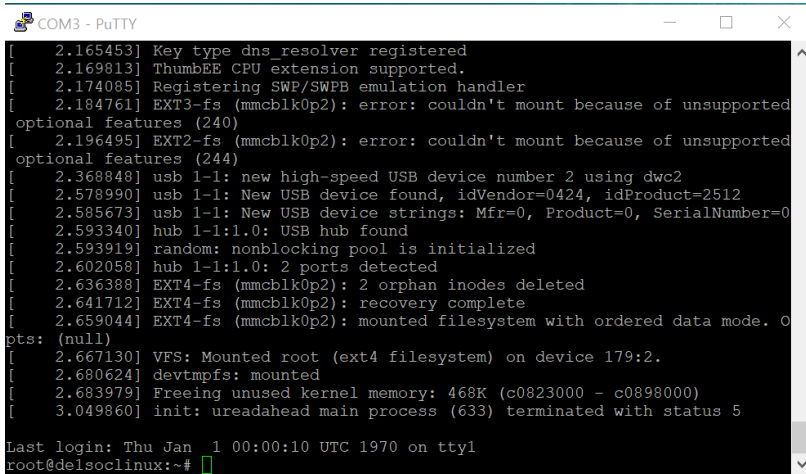
3. FPGA Bridge 설정



→ FPGA와 HPS를 동시에 사용하기 위해 Bridge를 다음과 같이 설정한다.

임베디드시스템 설계 및 실습 - 보고서

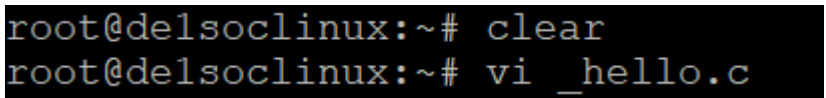
4. PuTTY를 통해 Board booting



```
COM3 - PuTTY
[ 2.165453] Key type dns_resolver registered
[ 2.169813] ThumbEE CPU extension supported.
[ 2.174085] Registering SWP/SWPB emulation handler
[ 2.184761] EXT3-fs (mmcblk0p2): error: couldn't mount because of unsupported
optional features (240)
[ 2.196495] EXT2-fs (mmcblk0p2): error: couldn't mount because of unsupported
optional features (244)
[ 2.368848] usb 1-1: new high-speed USB device number 2 using dwc2
[ 2.578990] usb 1-1: New USB device found, idVendor=0424, idProduct=2512
[ 2.585673] usb 1-1: New USB device strings: Mfr=0, Product=0, SerialNumber=0
[ 2.593340] hub 1-1:1.0: USB hub found
[ 2.593919] random: nonblocking pool is initialized
[ 2.602058] hub 1-1:1.0: 2 ports detected
[ 2.636388] EXT4-fs (mmcblk0p2): 2 orphan inodes deleted
[ 2.641712] EXT4-fs (mmcblk0p2): recovery complete
[ 2.659044] EXT4-fs (mmcblk0p2): mounted filesystem with ordered data mode. O
pts: (null)
[ 2.667130] VFS: Mounted root (ext4 filesystem) on device 179:2.
[ 2.680624] devtmpfs: mounted
[ 2.683979] Freeing unused kernel memory: 468K (c0823000 - c0898000)
[ 3.049860] init: ureadahead main process (633) terminated with status 5
Last login: Thu Jan  1 00:00:10 UTC 1970 on tty1
root@delsoclinux:~#
```

→ PuTTY에서 serial line과 speed 설정 후 board의 power를 on 하여 수행할 수 있었다.

5. vim editor를 사용하여 _hello.c 파일 생성



```
root@delsoclinux:~# clear
root@delsoclinux:~# vi _hello.c
```

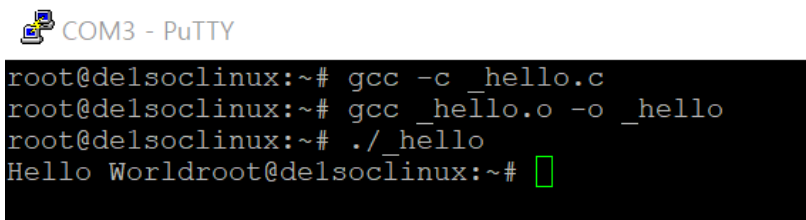
6. Hello World 출력 program 생성



```
COM3 - PuTTY
#include<stdio.h>
int main(void){
printf("Hello World");
return 0;
}
```

→ code를 기술한 후 :wq명령을 통해서 파일을 저장할 수 있었다.

7. Compile and Run



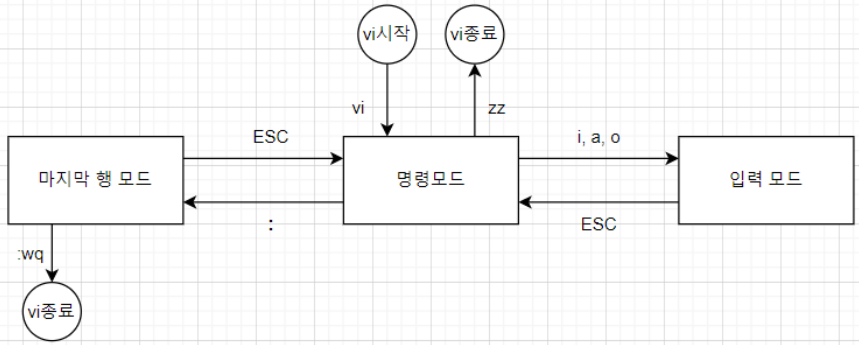
```
COM3 - PuTTY
root@delsoclinux:~# gcc -c _hello.c
root@delsoclinux:~# gcc _hello.o -o _hello
root@delsoclinux:~# ./_hello
Hello Worldroot@delsoclinux:~#
```

→ gcc의 -c 옵션을 사용하여 _hello.c compile 한 후 실행 파일 명을 _hello로 설정하여 실행할 수 있었다.

결과 분석 및 팀원 간 토의 사항

나머지 실습에 필요한 setup을 진행하였다. micro SD card에 linux image 파일을 다운로드 받아 FPGA 보드에 올린 후, PuTTY 프로그램을 통해 보드와의 Serial communication을 시작하여 로그인이나 별다른 조치 없이 리눅스를 부팅할 수 있다. 리눅스에서 파일을 만들기 위해서는 리눅스에 기본 탑재된 vim 편집기를 사용한다. 다음은 vi의 기본 동작법을 모드 별로 정리한 다이어그램과 각각을 설명한 표이다.

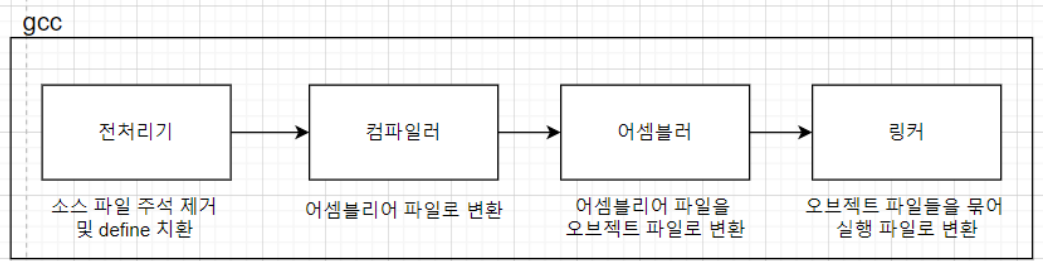
임베디드시스템 설계 및 실습 - 보고서



모드	역할 및 특징
명령모드	커서의 이동, 수정, 삭제, 복사 붙이기, 탐색 등을 수행할 수 있음
입력모드	글자를 입력하는 문서를 만드는 모드
마지막 행 모드	명령모드에서 ':'를 입력했을 때 화면 맨 아랫줄에서 명령을 수행하는 모드; 저장, 종료, 탐색, 치환 등의 동작을 수행할 수 있음

vi 명령어를 사용하여 hello.c 파일을 생성하고, 파일 안에서 i를 눌러 입력 모드로 들어간 뒤 파일의 내용을 작성한다. 파일을 작성한 뒤에 ESC를 눌러 명령모드로, :를 눌러 마지막 행 모드로, wq를 입력하여 저장하고 vi 편집기를 종료하는 순서로 진행되었다.

gcc 컴파일러는 아래의 일련의 과정을 통해 실행 파일을 만들어준다.



-c 명령어로 c 파일을 링킹 전 단계까지 compile 할 수 있고, -o 명령어로 출력 파일명을 지정할 수 있다. 출력 파일을 실행하기 위해서는 ./[파일명] 을 실행하면 된다.

✓ Assignment 2

개요

Assignment 1에서 수행한 내용을 기반으로 Board의 HPS에서 linux를 기반으로 multi-object C program을 만드는 실험을 수행한다. 두 개 이상의 object를 생성/컴파일 하고, link과정을 통해 하나의 실행파일을 만들어 프로그램을 실행할 수 있다.

진행 내용

```
//////// first_file.c
#include<stdio.h>
Void second_file(void);          // 다른 object의 함수를 선언
Int main(void){
Printf("first_file success\n");
Second_file();                  //다른 object의 함수를 실행
Return 0;
}
```

임베디드시스템 설계 및 실습 - 보고서

```
////////second_file.c
```

```
#include<stdio.h>
```

```
void second_file(void){
```

```
Printf("second_file success");
```

```
}
```

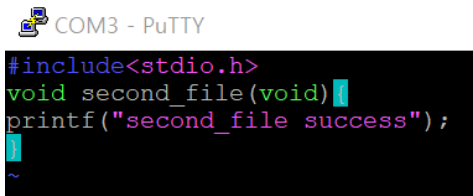
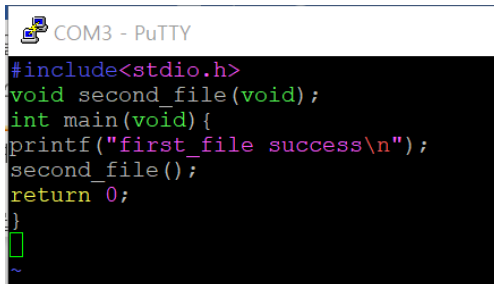
```
root@de1soclinux:~# gcc -c first_file.c // first_file compile
```

```
root@de1soclinux:~# gcc -c second_file.c // second_file compile
```

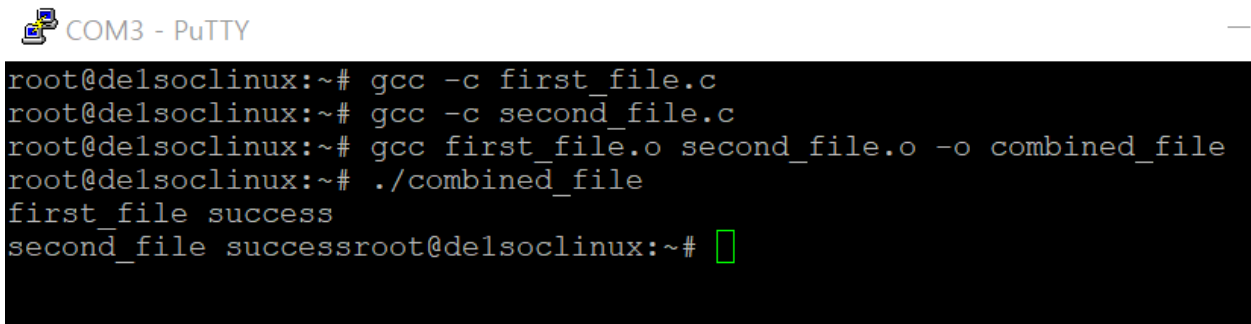
```
root@de1soclinux:~# gcc first_file.o second_file.o -o combined_file // 두 object file link하여 하나의 실행파일 생성
```

```
root@de1soclinux:~# ./combined_file // 실행파일 실행
```

진행 결과



→ assignment 1 과 마찬가지로 vim editor를 사용하여 code를 기술한 후 :wq명령을 통해 두 소스파일을 저장한다.



→ 두 object를 gcc를 통해서 컴파일한 후 생성된 각 오브젝트 파일을 하나의 실행파일(combined_file)로 링크한다. 그 실행파일을 실행하게 되면 정상적으로 두 코드가 동작하는 것을 확인할 수 있다.

결과 분석 및 팀원 간 토의 사항

Assignment1을 바탕으로 gcc 컴파일러의 사용 방법을 익히고 이를 통해 두 개의 소스파일 링크하여 기계어로 된 하나의 오브젝

임베디드시스템 설계 및 실습 - 보고서

트 파일로 변환하였다. gcc컴파일러는 파일의 확장자에 따라 다르게 처리할 수 있으며, 확장자에 따라 파일의 종류와 처리 방법은 다음과 같다.

확장자	종류	처리방법
.c	C소스 파일	gcc로 전처리, 컴파일, 어셈블, 링크
.i	전처리된 C파일	gcc로 컴파일, 어셈블, 링크
.s	어셈블리어로 된 파일	어셈블, 링크
.o	오브젝트 파일	링크
.a, .so	컴파일된 라이브러리 파일	링크

본 실습에서는 c언어를 사용하였으므로 소스 파일을 이름.c로 지정하며, 향후 어셈블리어로 프로그램을 작성할 때 파일 확장자를 .s로 지정해야 함을 알 수 있다.

gcc 컴파일러의 옵션 중에 -o라고 하면 바이너리 형식의 출력 파일 이름을 지정할 수 있다.

```
root@delsoclinux:~# gcc first_file.o second_file.o -o combined_file
```

따라서 본 실습에서는 두 개의 c언어 소스파일을 하나의 출력 파일로 지정하였다. 만약 -o 옵션을 생략하고 컴파일을 하면 실행 파일 이름은 a.out이 된다.

File1	File2	실행 결과
<pre>void file2(); void main(){ printf("file1 \n"); file2(); }</pre>	<pre>void file2(){ printf("file2"); }</pre>	<pre>root@delsoclinux:~# gcc file1.o file2.o root@delsoclinux:~# ./a.out file1 file2root@delsoclinux:~#</pre>

이 때 두 가지 다른 소스를 차례로 컴파일 할 때 먼저 생성된 실행 파일 a.out을 나중에 생성된 a.out이 경고 없이 덮어쓰므로 주의해야한다.

만약 아래와 같이 파일별로 컴파일을 하면 컴파일 오류가 발생한다.

```
root@delsoclinux:~# gcc file1.c -o main
/tmp/ccDLHcYh.o: In function `main':
file1.c:(.text+0x10): undefined reference to `file2'
```

```
root@delsoclinux:~# gcc file2.c -o file2
/usr/lib/gcc/arm-linux-gnueabi/4.6/../../../../arm-linux-gnueabi/libcrt1.o: In function `_start':
(.text+0x30): undefined reference to `main'
```

file1에 대한 오류는 호출하는 함수 file2가 정의되지 않았기 때문이고, file2에 대한 오류는 main함수가 없기 때문에 발생하는 것임을 알 수 있다.

임베디드시스템 설계 및 실습 - 보고서

✓ Assignment 3 (Optional)

개요

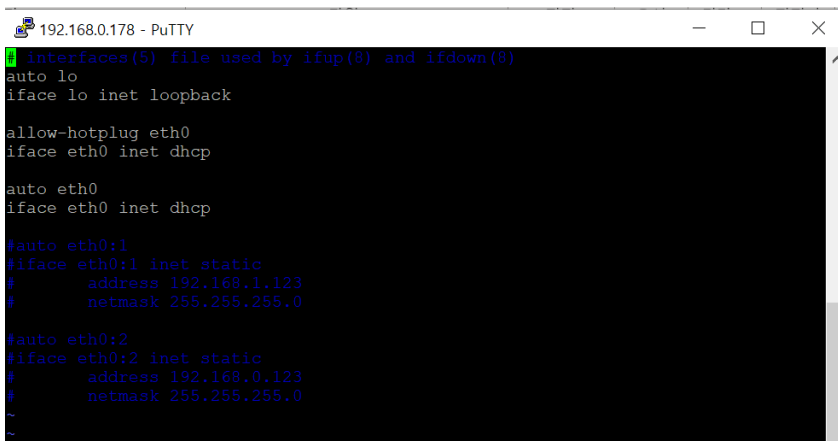
Assignment 1, 2에서 serial connection을 통해 수행한 내용을 Ethernet connection을 통해 다시 수행하는 파트이다. Linux상에서 board가 ethernet 케이블을 통해 ip를 할당 받을 수 있도록 setup하는 과정과 router에서 수동으로 board에 ip를 할당할 수 있도록 setup하는 과정이 필요했다.

진행 내용

Assignment 1, Assignment 2의 코드 내용과 동일합니다.

같은 linux 명령어를 통해서 Assignment 1, 2에서 생성한 프로젝트와 실행파일에 접근하였고 동일한 수행결과가 도출되는 것을 이후 진행 결과 부분에 사진으로 게시하였습니다

진행 결과



```
192.168.0.178 - PuTTY
# interfaces(5) file used by ifup(8) and ifdown(8)
auto lo
iface lo inet loopback

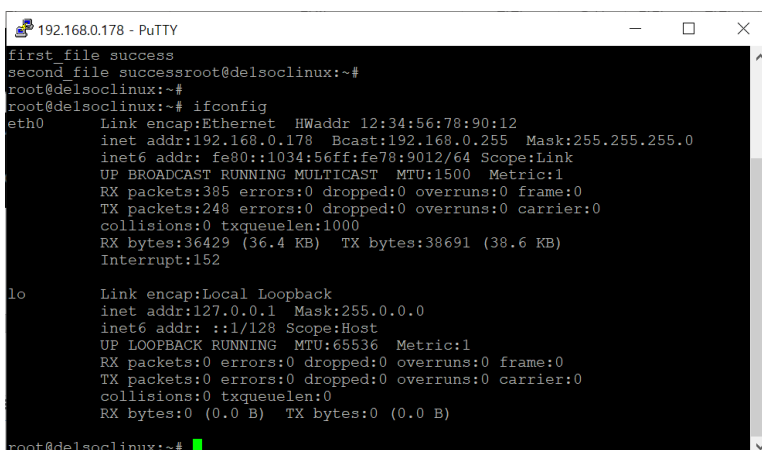
allow-hotplug eth0
iface eth0 inet dhcp

auto eth0
iface eth0 inet dhcp

#auto eth0:1
#iface eth0:1 inet static
#    address 192.168.1.123
#    netmask 255.255.255.0

#auto eth0:2
#iface eth0:2 inet static
#    address 192.168.0.123
#    netmask 255.255.255.0
~
~
```

→ 다음과 같이 linux 상에서 ip를 할당 받을 수 있도록 설정해주었다.



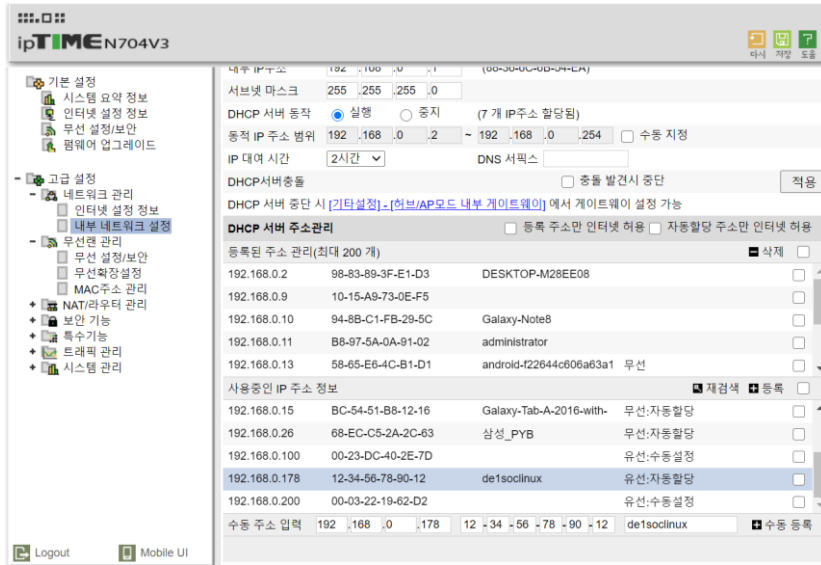
```
192.168.0.178 - PuTTY
first file success
second file successroot@delsoclinux:~#
root@delsoclinux:~#
root@delsoclinux:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 12:34:56:78:90:12
          inet addr:192.168.0.178  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::1034:56ff:fe78:9012/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:385 errors:0 dropped:0 overruns:0 frame:0
          TX packets:248 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:36429 (36.4 KB)  TX bytes:38691 (38.6 KB)
          Interrupt:152

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

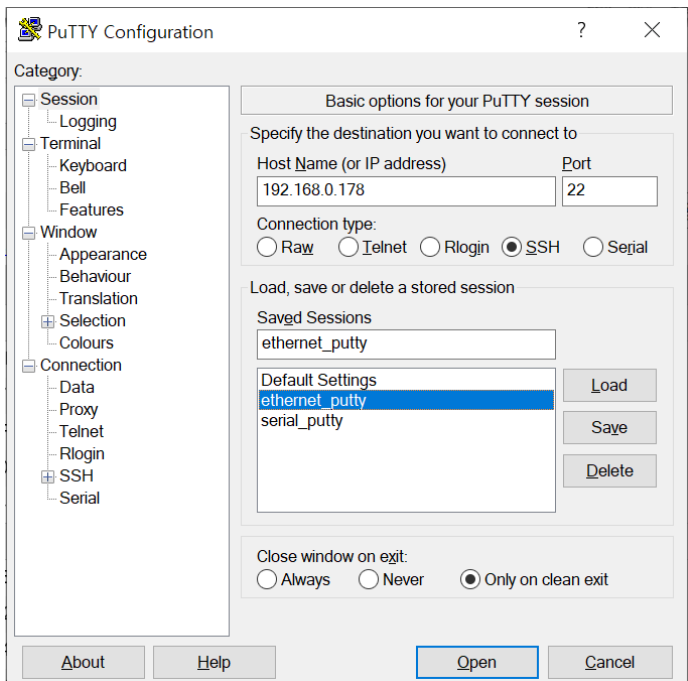
root@delsoclinux:~#
```

→ ifconfig 명령을 통해 HWaddr을 확인하고 router를 수동으로 setup할 수 있었다.

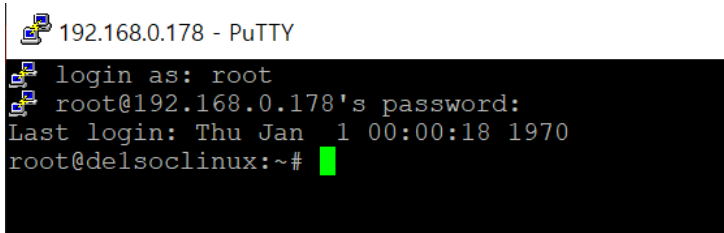
임베디드시스템 설계 및 실습 - 보고서



→ 수동으로 HWaddr를 입력하고, 할당 받을 ip 주소를 입력한 뒤 해당 ip이름을 de1soclinux로 설정하였다.



→ 할당 받은 ip로 PuTTY를 통해 ssh communication을 할 수 있었다.



→ 초기 id와 비밀번호를 입력하면 serial 통신과 마찬가지로 HPS 상에서 linux를 구동할 수 있게 된다.

임베디드시스템 설계 및 실습 - 보고서

```
192.168.0.178 - PuTTY
#include<stdio.h>
int main(void){
printf("Hello World");
return 0;
}
```

```
192.168.0.178 - PuTTY
login as: root
root@192.168.0.178's password:
Last login: Thu Jan 1 00:00:18 1970
root@delsoclinux:~# vi _hello.c
root@delsoclinux:~# ./_hello
Hello Worldroot@delsoclinux:~#
```

→ Assignment 1에서 board에 생성한 _hello.c 프로젝트와 hello 실행파일을 접근할 수 있고, 동일한 실행 결과를 얻게 되는 것을 확인할 수 있었다.

```
192.168.0.178 - PuTTY
#include<stdio.h>
void second_file(void);
int main(void){
printf("first_file success\n");
second_file();
return 0;
}
```

```
192.168.0.178 - PuTTY
#include<stdio.h>
void second_file(void){
printf("second_file success");
}
```

```
root@delsoclinux:~# ./combined_file
first_file success
second_file successroot@delsoclinux:~#
```

→ Assignment 2에서 board에 생성한 first_file.c, second_file.c 프로젝트와 combined_file 실행파일에 접근할 수 있었고, 동일한 실행 결과를 확인할 수 있었다.

결과 분석 및 팀원 간 토의 사항

<결과분석>

Assignment 1, 2 에서 수행한 과정을 board와의 connection 방식을 달리하여 수행하는 파트였기 때문에 이미 생성된 파일을 다시 접근하고 실행하는 과정을 수행했다. Ethernet connection은 serial communication과는 다르게 linux 상에서 DHCP를 받을 수 있도록 설정하는 과정과 router에서 board에 할당할 ip 주소와 HWaddr를 수동으로 설정하는 과정이 필요했고, 이 과정에서 많은 시행착오가 있었다. Ethernet connection이 성공한 후에는 serial communication과 동일하게 board의 HPS 상에서 linux 환경이 구축되어 동작하는 것을 확인할 수 있었다.

<토의 내용>

주로 카카오톡 오픈채팅방을 통해 소통하며 LAB을 진행하였고, Assignment 3에 대해서는 router를 설정하는 방법을 알아내는 것이 핵심 요소였다. Ethernet connection을 사용할 때의 장점은 password를 통한 보안, 빠른 통신 속도, SSH 등 ip-based application 들을 사용할 수 있다는 것이다. 두 가지 방식으로 board에 접근할 수 있다는 것을 알았고, ip를 직접 할당하여 board에 연결하는 과정이 생소했지만 Ethernet connection을 통해 가능하다는 것을 확인할 수 있었다.

임베디드시스템 설계 및 실습 - 보고서

✓ 조원 별 기여 사항 및 느낀점

이름	기여도 (0 - 100%)	기여 사항	느낀점
박관영	33.3%	- assignment 1,2,3 진행 결과 작성 - assignment 1,2,3 수행	처음 board 상의 linux 환경을 접하는 것이 생소했지만, 두 가지 방식으로 board에 접근하여 assignment를 수행하면서 linux 환경에 익숙해질 수 있었다.
오한별	33.3%	- assignment 1,2,3 수행 및 결과 분석	평소에 쓰던 VS code나 Pycharm과 달리 shell command는 user interface가 잘 안 되어있어 조금은 불편했다. vi와 gcc의 기본적인 command line에 대해 배울 수 있어 유익했고, 앞으로 assignment를 진행하는 데에 많은 도움이 될 것 같다.
김서영	33.3%	- assignment 1,2,3 수행 및 결과 분석	Linux 환경에서 vim을 사용하는 법을 익혔고, . gcc컴파일러의 기본적인 동작 원리와 command 입력 방식을 익히고 확인할 수 있었다. 특히 uart와 ssh 등의 통신 방법을 직접 실습해봄으로써 통신 방법에 관한 지식을 쌓을 수 있었다