

# **Embedded Linux on Zynq using Vivado Lab4**

**Innova Lee(이상훈)  
gcccompil3r@gmail.com**

## Introduction

완전한 TCP/IP 스택 및 다양한 네트워킹 애플리케이션의

준비된 유효성은 임베디드 리눅스를 사용하는 것을 선호하는 주요 기능이다.

이 Lab에서는 임베디드 리눅스 네트워킹에 대해 소개하고 응용 프로그램 개발 및 배포 중 유용할 수 있는 방법을 보여준다.

이전 Lab에서 이미 Linux 네트워킹 기능(TFTP 유틸리티)을 사용하여 네트워크를 통해 Linux 이미지를 가져왔다.

이 Lab에서는 시스템의 네트워킹 기능을 보다 명확하게 사용하고

특히 응용 프로그램 구축 / 다운로드 / 테스트 주기를 획기적으로 단축하는데 어떻게 사용되는지 확인한다.

또한 개발 보드에서 일부 물리적 I/O를 제어할 수 있는 웹 가능 애플리케이션을 빌드할 것이다.

이것은 매우 간단한 프로그램이지만 훨씬 더 강력한 것을 암시한다.

## Objectives

이 Lab을 완료하면 아래를 수행할 수 있다:

- 커널 구성 메뉴를 탐색하고 Linux TCP/IP 네트워킹을 가능하게 하는 구성 하위 메뉴를 확인한다.
- 텔넷을 사용하여 ARM Cortex-A9 프로세서 Linux System에 로그인한다.
- FTP를 사용하여 Linux와 파일 주고 받기
- NFS(Network File System)을 사용하여 Linux Target에 Host 파일 시스템을 탑재하고 이 기능이 임베디드 개발 주기에 영향을 주는 방식을 조사한다.
- Linux Target의 내장 웹 서버를 실험한다.
- Linux 기반의 웹 기반 응용 프로그램 구축 및 실험

## Preparation

이것이 첫 번째 Lab인 경우 환경 설정 방법에 대한 필수 준비 정보는 Lab 1의 "Before You Start" 섹션을 참조하라.

워크 스테이션이 다시 시작되거나 로그 아웃되면 아래 명령을 실행하여 Host에서 DHCP 서버를 시작한다:

```
[host]$ sudo service isc-dhcp-server restart
```

자세한 내용은 Lab 1의 "Initializing the Workshop Environment" 섹션을 참조하라.

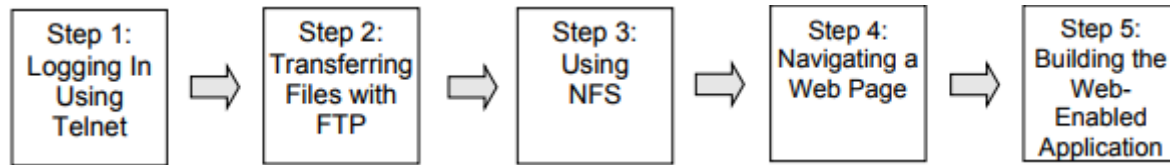
## Exploring Network Features

보드의 기본 내장 Linux 이미지는 네트워크 응용 프로그램을 지원한다.

이더넷 지원을 활성화하기 위한 Linux 설정 및

이 Lab 에서 사용되는 네트워크 응용 프로그램에 관심이 있는 경우 본 실습의 부록 절을 참조하라.

## General Flow for this Lab



## Logging In Using Telnet

이전 Lab 에서 Serial 회선을 통해 GtkTerm 을 사용하여 ARM Cortex-A9 MPcore 시스템에 로그인 했다.

디버깅 및 개발에 편리하지만 직접 Serial 연결이 필요하다.

시스템을 배포 할 때 사용할 수 없는 경우가 있다.

리눅스는 표준 텔넷 프로토콜을 직접 지원한다.

실제로 이것은 ARM Cortex-A9 MPcore 에서 이미 활성화되어 있다.

1-1. 경로를 프로젝트 디렉토리로 변경한다.

1-1-1. 아래 명령을 실행하여 프로젝트 디렉토리 경로를 만들고 변경한다.

```
[host]$ mkdir ~/emblnx/labs/lab4
```

```
[host]$ cd ~/emblnx/labs/lab4
```

1-2. petalinux-create 명령을 사용하여 새로운 임베디드 리눅스 플랫폼을 만들고 플랫폼을 선택한다.

1-2-1. lab4 디렉토리에서 아래 명령을 실행하여 새로운 Petalinux 프로젝트를 만든다.

```
[host]$ petalinux-create -t project -s /opt/pkg/ZYBO_petalinux_v2015_4.bsp
```

이 명령은 ~/emblnx/labs/lab4 아래에 ZYBO\_petalinux\_v2015\_4 라는 소프트웨어 프로젝트 디렉토리를 만든다.

1-2-2. 디렉토리를 PetaLinux 프로젝트로 변경한다:

~/emblnx/labs/lab4/ZYBO\_petalinux\_v2015\_4 에 해당한다.

1-3. BOOT.BIN 및 image.ub 파일을 미리 만들어진 디렉토리에서 MICRO-SD 카드로 복사한다.

1-3-1. BOOT.BIN 과 image.ub 파일을

~/emblnx/labs/lab4/ZYBO\_petalinux\_v2015\_4/pre-built/linux/images 디렉토리에서 MICRO-SD 카드로 복사한다.

1-3-2. 보드가 꺼져 있는지 확인한다.

1-3-3. MICRO-SD 카드를 Target Board 에 삽입한다.

1-3-4. Board 가 MICRO-SD 카드에서 부팅되도록 설정되어 있는지 확인한다.

1-4. Host 에서 DHCP 서버를 실행한다.

1-4-1. DHCP 서버를 실행한다.

```
[host]$ sudo service isc-dhcp-server restart
```

1-5. 보드의 전원을 켜고 Serial 포트 터미널을 설정한다.

1-5-1. 보드의 전원을 켜다.

1-5-2. 아래 명령을 실행하여 /dev/ttyUSB1 이 read/write 접근으로 설정되어 있는지 확인한다.

```
[host]$ sudo chmod 666 /dev/ttyUSB1
```

1-5-3. dashboard 의 검색 필드에 Serial 포트를 입력한다.

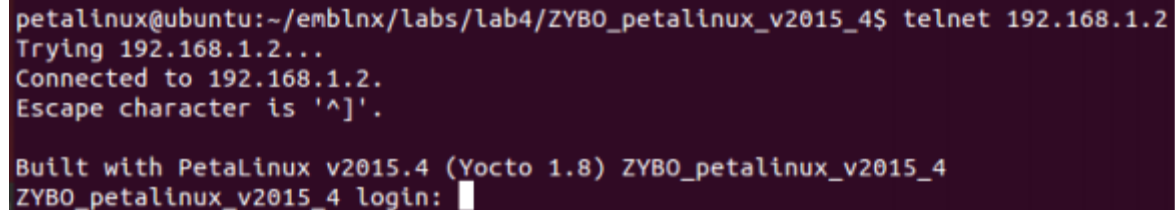
1-5-4. Serial 포트 터미널 응용 프로그램을 선택한다.

1-6. Target Board 의 ARM Cortex-A9 프로세서 시스템에 텔넷 연결

1-6-1. 새 터미널을 열고 192.168.1.2 IP 주소로 Host 에서 telnet 명령을 실행한다.

```
[host]$ telnet <IP 주소>
```

아래와 같이 Host 의 telnet 콘솔에 출력이 표시된다:



```
petalinux@ubuntu:~/emblnx/labs/lab4/ZYBO_petalinux_v2015_4$ telnet 192.168.1.2
Trying 192.168.1.2...
Connected to 192.168.1.2.
Escape character is '^]'.

Built with PetaLinux v2015.4 (Yocto 1.8) ZYBO_petalinux_v2015_4
ZYBO_petalinux_v2015_4 login: █
```

Figure 1. Telnet console on the host

1-6-2. 로그인 ID 와 암호로 root 를 사용하여 로그인한다.

1-6-3. telnet 콘솔에서 ls 또는 pwd 와 같은 일부 Linux 명령을 시도한다.

1-6-4. telnet 프로그램을 종료하려면 exit 를 입력한다.

## Transferring Files with FTP

FTP 는 자주 사용되는 또 다른 네트워크 기능이다.

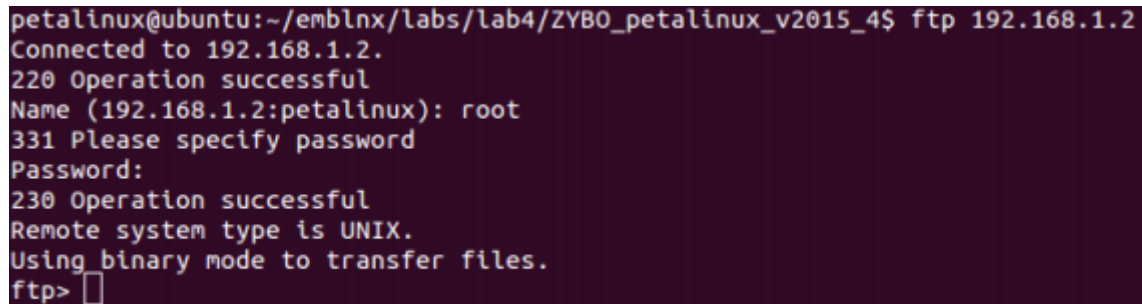
ARM Cortex-A9 MPcore Linux 시스템은 FTP 서버로 사전 구성되어 있다.

2-1. FTP 응용 프로그램을 실행하고 다양한 기능을 시험해보라.

2-1-1. 아래를 실행하여 Host 에서 FTP 응용 프로그램을 시작한다.

```
[host]$ ftp 192.168.1.2
```

2-1-2. 아래와 유사한 메시지가 나타나면 이름 프롬프트에서 엔터 키를 누른다:



```
petalinux@ubuntu:~/emblnx/labs/lab4/ZY80_petalinux_v2015_4$ ftp 192.168.1.2
Connected to 192.168.1.2.
220 Operation successful
Name (192.168.1.2:petalinux): root
331 Please specify password
Password:
230 Operation successful
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> 
```

Figure 2. FTP session

이제 FTP 프롬프트를 볼 수 있다:

```
ftp>
```

이제 ARM Cortex-A9 MPcore 시스템으로 파일을 전송하거나

ARM Cortex-A9 MPcore 시스템에서 파일을 전송할 수 있다.

ARM Cortex-A9 MPcore 시스템으로 파일을 보내는 경우

ARM Cortex-A9 MPcore 시스템의 FTP 홈 디렉토리는 /var/ftp 에 해당한다.

파일을 가져와서 해당 디렉토리에만 놓을 수 있다.

2-1-3. ~/emblnx/labs/lab4/ZYBO\_petalinux\_v2015\_4 에 있는 config.project 파일을 넣으려면 아래 명령을 입력한다.

```
[host] ftp> put config.project
```

2-1-4. 이름과 암호로 root 를 사용하여 보드에 로그인한다.

2-1-5. config.project 파일이 /var/ftp 디렉토리에 있는지 확인한다.

2-1-6. ftp 를 종료하려면 bye 를 입력한다.

2-1-7. 터미널을 닫는다.

## Using NFS

NFS(Network File System)는 오랫동안 지원 된 Linux(및 임베디드 Linux) 기능이다.

원격 파일 시스템을 네트워크를 통해 마운트하고 물리적으로 마치 로컬 호스트에 있는 것처럼 사용할 수 있다.

크로스 컴파일 된 임베디드 리눅스 시스템의 맥락에서 이것은 매우 중요하다.

NFS 는 응용 프로그램을 디버깅 할 때 매우 유용하다.

응용 프로그램을 변경할 때마다 전체 이미지를 다시 작성하고 다운로드 하는 대신

개발 디렉토리를 ARM Cortex-A9 MPcore 시스템에 간단히 마운트 할 수 있다.

응용 프로그램을 다시 컴파일하면 새 버전을 즉시 Target 에서 실행할 수 있다.

3-1. LiveUSB 파티션 이름을 결정하고 필요한 경우 두 번째 파티션을 마운트한다.

3-1-1. 시스템의 다양한 파티션을 보려면 아래 명령을 입력한다.

[host] ftp> df

다양한 마운트 지점에 다양한 파일 시스템 구성 요소가 마운트 되어 있다.  
/dev/sdb2 가 /media/petalinux/casper-rw 에 마운트되어 있는지 확인한다.  
Target 보드가 내용(/home/petalinux 디렉토리 및 하위 디렉토리 포함)에  
접근할 수 있도록 export 할 마운트 지점을 식별한다.

```
petalinux@ubuntu:~/emblnx/labs/lab4/ZYB0_petalinux_v2015_4$ df
Filesystem      1K-blocks    Used Available Use% Mounted on
/cow             55490620 43251308   9397508  83% /
udev            4045488      4   4045484    1% /dev
tmpfs           811396     1392   810004    1% /run
/dev/sdb1       5240272 5213744    26528 100% /cdrom
/dev/loop0     975872   975872      0 100% /rofs
none              4          0          4  0% /sys/fs/cgroup
tmpfs          4056980      8   4056972    1% /tmp
none            5120      4        5116    1% /run/lock
none          4056980     84   4056896    1% /run/shm
none           102400     36   102364    1% /run/user
/dev/sdb2       55490620 43251308   9397508  83% /media/petalinux/casper-rw
petalinux@ubuntu:~/emblnx/labs/lab4/ZYB0_petalinux_v2015_4$
```

Figure 3. Mounted partitions in the system

두 번째 파티션이 마운트되지 않으면 다음 섹션을 따르라.  
그렇지 않으면 건너뛰다.

3-2. 필요한 경우 두 번째 파티션을 마운트한다.  
그렇지 않으면 이 단계를 건너 뛰다.

3-2-1. dashboard 에서 Disk 를 입력한다.

3-2-2. Disk 유틸리티를 선택한다.



3-2-3. LiveUSB 장치를 선택한다.

3-2-4. 두 번째 파티션을 선택하고 이름을 적는다.  
아래 그림에서 casper-rw 파티션을 보여준다.

3-2-5. Mount Volume 을 클릭한다.

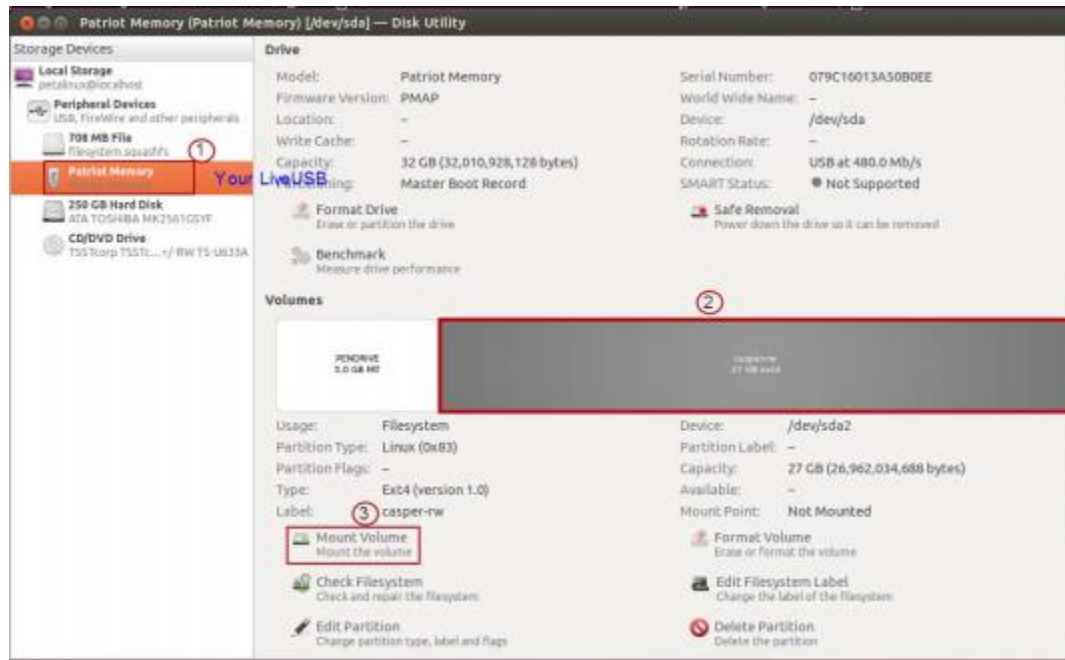


Figure 4. Determining the LiveUSB device's partition names and mounting the 2<sup>nd</sup> partition

마운트가 끝나면 마운트 지점이 `/media/casper-rw` 로 표시된다.

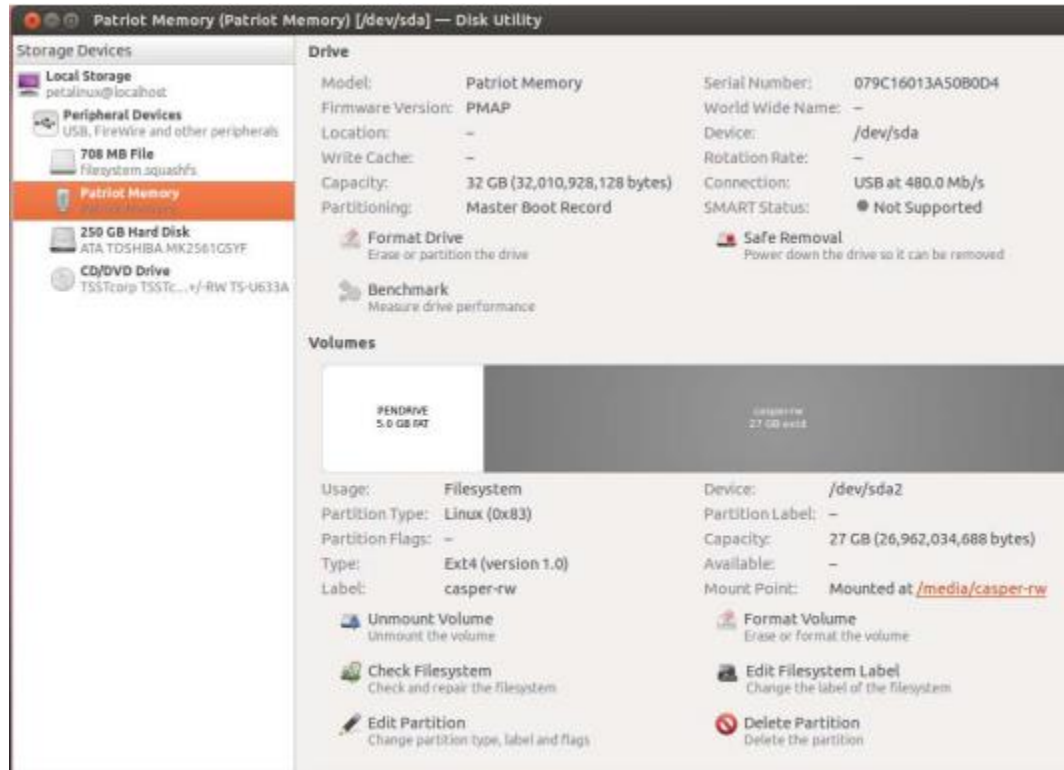


Figure 5. casper-rw mounted as `/media/casper-rw`

3-2-6. Disk 유틸리티 응용 프로그램을 닫는다.

3-3. ARM Cortex-A9 MPcore 시스템에서 Host 의 원격 파일 시스템을 마운트하려면 Host 가 이를 허용하도록 구성해야 한다.  
이것은 `/etc/exports` 파일에 지정된다.

Host 가 올바르게 구성되었는지 확인한다.

3-3-1. 새 터미널을 연다.

3-3-2. 아래 명령을 입력한다.

```
[host]$ df
```

참고: /dev/sdb2 (이 경우) 가 Host 시스템에서 /media/petalinux/casper-rw 로 마운트되어 있는지 확인한다.  
시스템에 따라 다를 수 있다.

3-3-3. 아래를 실행하여 /etc/exports 파일의 내용을 검사한다.

```
[host]$ cat /etc/exports
```

3-3-4. /etc/exports 파일에서 아래 줄을 찾는다

```
/home/petalinux 192.168.*.*  
(rw, sync, no_root_squash, no_subtree_check)
```

이것은 /home/petalinux 디렉토리가 IP 주소  
192.168.\*.\* (192.168.0.1 에서 192.168.255.255 까지의 IP 주소)를 갖는 컴퓨터로  
export 할 수 있고 read/write 권한으로 마운트 될 수 있음을 의미한다.

그러나 /home/petalinux 가 마운트되어 있지 않은 것에 주의하라.

/media/petalinux/casper-rw 가 마운트되었으므로  
/etc/exports 파일을 편집하고 (sudo 명령을 사용해야 함) 아래와 같이 행을 변경한다:

```
/media/petalinux/casper-rw/home/petalinux 192.168.*.*  
(rw, sync, no_root_squash, no_subtree_check)
```

이것은 /media/petalinux/casper-rw/home/petalinux 라는 디렉토리가  
IP 주소 192.168.\*.\* (192.168.0.1 에서 192.168.255.255 까지의 IP 주소)를 가진 머신으로  
export 될 수 있고 read-write 권한으로 마운트 될 수 있음을 의미한다.

### 3-3-5. Host 에서 NFS 서버를 다시 시작한다.

```
[host]$ sudo /etc/init.d/nfs-kernel-server restart
```

이 명령은 NFS 서비스가 실행 중일 때 NFS 서비스 실행을 중지 한 다음 다시 시작한다.

아래는 이 명령의 Host 출력이다:

```
Stopping NFS kernel daemon           [ OK ]
Unexporting directories for NFS kernal daemon... [ OK ]
Exporting directories for NFS kernel daemon... [ OK ]
Starting NFS kernel daemon:          [ OK ]
```

공유 폴더를 변경하려면 아래를 수행해야 한다.

- /etc/exports 파일을 편집하라.
- 아래를 실행하여 NFS 서버를 다시 시작한다.

```
[host]$ sudo /etc/init.d/nfs-kernel-server restart
```

이제 Host 는 ARM Cortex-A9 MPcore 시스템이 /home/petalinux 디렉토리에 NFS 마운트 되도록 한다.

### 3-4. NFS 마운트에는 portmap 애플리케이션이 필요하다.

사전 빌드 된 Images 가 포함되지 않았다.

따라서 rootfs 를 구성하여 portmap 응용 프로그램을 활성화하고 Image 를 다시 작성한다.

rootfs 구성 메뉴를 시작하고 요구 사항을 충족하도록 구성한다.

#### 3-4-1. 터미널에서 아래 명령을 실행한다.

```
[host]$ petalinux-config -c rootfs
```

### 3-4-2. Filesystem 패키지를 선택한다.

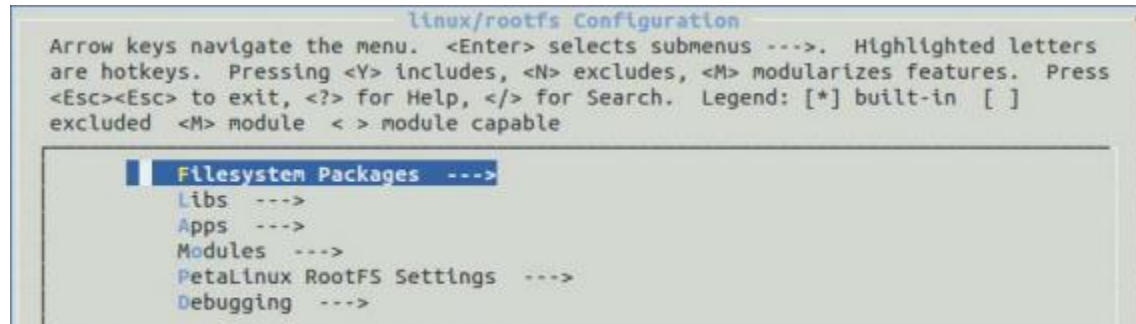


Figure 6. Selecting Filesystem Packages

### 3-4-3. 엔터 키를 눌러 하위 메뉴로 이동한다.

### 3-4-4. console/network 를 선택한다.

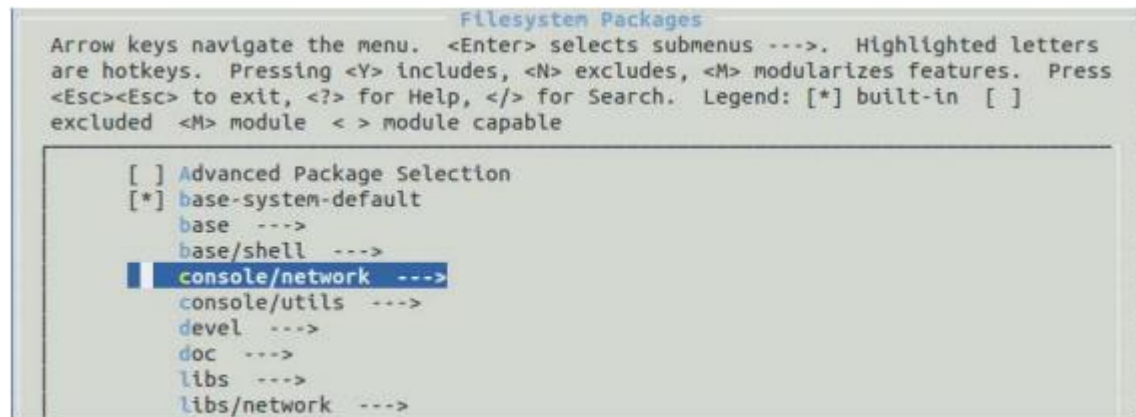


Figure 7. Selecting console/network option

### 3-4-5. 엔터 키를 눌러 하위 메뉴로 이동한다.

### 3-4-6. portmap 을 선택한다.

```
console/network
Arrow keys navigate the menu. <Enter> selects 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 capable

bridge-utils --->
dropbear --->
ethtool --->
netcat --->
openssh --->
portmap --->
tcp-wrappers --->
tcpdump --->
```

Figure 8. Selecting portmap option

### 3-4-7. 엔터 키를 눌러 하위 메뉴로 이동한다.

### 3-4-8. <Y> 를 클릭하여 portmap 을 활성화한다.

```
portmap
Arrow keys navigate the menu. <Enter> selects 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 capable

[*] portmap
[ ] portmap-utils
```

Figure 9. Selecting the portmap application

### 3-4-9. Exit 를 여러 번 선택하고 구성을 저장한다.

### 3-4-10. petalinux-build 명령을 실행하여 시스템 이미지를 빌드한다.

[host]\$ petalinux-build

3-5. 새로운 리눅스 이미지를 보드에서 부팅한다.

3-5-1. 보드가 부팅 절차를 진행할 때 GtkTerm 콘솔에서 부팅 정보를 보려면 보드(BTN7)을 재설정하라.

3-5-2. 자동 부팅을 중지하려면 아무 키나 누른다.

3-5-3. uboot 부팅 중에 "DHCP client bound to address" 메시지가 표시되지 않으면 dhcp 를 실행하여 IP 주소를 얻는다.

```
U-Boot-PetaLinux> dhcp
```

3-5-4. u-boot 콘솔에서 아래 명령을 실행하여 TFTP 서버 IP 를 Host IP 로 설정한다.

```
U-Boot-PetaLinux> set serverip 192.168.1.1
```

3-5-5. u-boot 콘솔에서 아래 명령을 실행하여 TFTP 를 사용하여 새 이미지를 다운로드하고 부팅한다:

```
U-Boot-PetaLinux> run netboot
```

이 명령은 Host 의 /tftpboot 에서 ARM Cortex-A9 MPcore 시스템의 주 메모리로 image.ub 파일을 다운로드하고 Image 로 시스템을 부팅한다.

netboot 명령은 이미지 다운로드가 끝나자마자 자동으로 시스템을 부팅한다.

보드가 바인딩 된 IP 주소를 기록한다.

ifconfig 명령을 사용하여 얻을 수도 있다.

3-6. 네트워크 장치 드라이버가 초기화 될 때, 리눅스 네트워킹 스택이 구성 될 때,  
그리고 결국 portmap 애플리케이션이 실행 될 때를 보아야 한다.

이 portmap 응용 프로그램은 NFS 마운트에 필요하다.

ARM Cortex-A9 MPcore 시스템의 데스크탑 PC 에 파일 시스템을 마운트한다.

3-6-1. Target Board 에 로그인한다.

3-6-2. 콘솔에서 아래 명령을 실행한다.

```
#mount -t nfs 192.168.1.1:/media/petalinux/casper-rw/home/petalinux/mnt
```

이 명령은 mount 에게 아래와 같이 알려준다:

- NFS 유형(-t NFS) 의 파일 시스템을 마운트하려고 한다.
- 이 파일 시스템의 Host 는 IP 주소 192.168.1.1 을 가진다.
- 마운트 할 Host 의 디렉토리는 /home/petalinux(홈 디렉토리)에 해당한다.

3-7. ARM Cortex-A9 시스템에서 /mnt 디렉토리로 변경한다.

이전 Lab 에서 사용한 myapp 응용 프로그램을 변경하여 실험해 보라.

예로 "printf("Hello, Petalinux World!\n")" 를 "printf("Hello, Welcome to the Xilinx workshop!\n")" 으로 변경한다.  
Host 에서 다시 빌드하고 NFS 마운트를 통해 ARM Cortex-A9 MPcore 시스템에서 다시 실행한다.

3-7-1. 아래를 실행한다:

```
# cd /mnt
```

```
# ls
```

이것은 모두 이상하게 익숙한가 ?

데스크탑 컴퓨터의 홈 디렉토리여야 한다.

읽기/쓰기 권한이 있으므로 주의해야 한다.

이 마운트 된 NFS 드라이브에서 파일을 삭제하면 실제 컴퓨터에서 파일이 삭제됨을 의미한다.

3-7-2. NFS 마운트가 Host 시스템에서 어떻게 유용 할 수 있는지 보려면

GtkTerm 창에서 아래 명령을 실행하여 이전 Lab 에서 사용한 myapp 응용 프로그램을 살펴보면 된다.

```
# cd /mnt/emblnx/labs/lab3/ZYBO_petalinux_v2015_4/build/linux/rootfs/apps/myapp
```



3-7-3. 아래를 실행하여 네트워크를 통해 myapp 응용 프로그램을 직접 실행한다.

```
# ./myapp
```

3-7-4. 새 터미널을 연다.

3-7-5. myapp 디렉토리로 변경한다.

```
[host]$ cd ~/emblnx/labs/lab3/ZYBO_petalinux_v2015_4/components/apps/myapp
```

3-7-6. myapp.c 파일(예: print 문)을 약간 변경해본다.

```
[host]$ gedit myapp.c
```

3-7-7. 첫 번째 printf 문을 printf("Hello, Welcome to the XUP workshop!\n") 으로 변경한다.

3-7-8. PetaLinux 프로젝트 디렉토리로 변경한다.

```
[host]$ cd ~/emblnx/labs/lab3/ZYBO_petalinux_v2015_4
```

3-7-9. 응용 프로그램만 빌드한다.

```
[host]$ petalinux-build -c rootfs / myapp -x clean
```

```
[host]$ petalinux-build -c rootfs / myapp
```

3-7-10. 아래 명령을 실행하여 ARM Cortex-A9 MPcore 시스템에서 myapp 을 다시 실행한다.

```
# ./myapp
```

응용 프로그램의 출력에는 변경 사항이 반영되어야 한다.

응용 프로그램에 대한 Host 변경 사항은 NFS 마운트를 통해  
즉시 ARM Cortex-A9 MPcore 시스템에서 테스트 할 수 있다.

# Building the Web-Enabled Application

웹 인터페이스를 사용하여 장치를 제어하거나 센서 입력을 모니터링 할 때 웹 서비스 내장 응용 프로그램이 훨씬 유용하다. 이 단계에서는 ARM Cortex-A9 MPcore 시스템에서 간단한 웹 지원 응용 프로그램을 빌드하고 실험해 보겠다.

4-1. 이 단계에서는 웹 사용 응용 프로그램을 빌드한다.

보드의 LED 켜기/끄기를 제어하는 샘플 CGI 응용 프로그램이 제공된다.

이 프로그램을 빌드하고 단계별로 실행한다.

4-1-1. PetaLinux 프로젝트 위치에 있는지 확인한다.

즉 ~/emblnx/labs/lab4/ZYBO\_petalinux\_v2015\_4 에 해당한다.

4-1-2. 아래 명령을 입력하여 PetaLinux 프로젝트 내에 새로운 사용자 응용 프로그램을 생성한다:

```
[host]$ petalinux-create -t apps --name cgi-leds
```

새로 만든 응용 프로그램은 <projectroot>/components/apps/cgi-leds 디렉토리에서 찾을 수 있다. 여기서 <project-root> 는 ~/emblnx/labs/lab4/ZYBO\_petalinux\_v2015\_4 에 해당한다.

4-2. sources/lab4/cgi-leds 디렉토리에서 cgi-leds 소스를 복사한다.

4-2-1. 새로 생성 된 응용 프로그램 디렉토리로 변경한다.

```
[host]$ cd <project-root>/components/apps/cgi-leds
```

4-2-2. sources/lab4/cgi-leds 디렉토리에서 cgi-leds 응용 프로그램 관련 파일을 복사한다.

```
[host]$ cp ~/emblnx/sources/lab4/cgi-leds/* ./
```

주요 응용 프로그램은 cgi\_leds.c, led.cgi.c, 그리고 led-gpio.c 로 구성된다.

다른 파일은 작은 CGI 라이브러리 전용이다.

cgi-leds 프로젝트에서 찾을 수 있다.

Makefile 을 열면 Target 응용 프로그램 이름이 led.cgi 로 설정되어 있음을 알 수 있다.

4-3. 빌드 프로세스에 포함될 새 애플리케이션을 선택한다.  
응용 프로그램은 기본적으로 활성화되어 있지 않다.

4-3-1. 프로젝트 디렉토리에 있는지 확인한다.  
즉 ~/emblnx/labs/lab4/ZYBO\_petalinux\_v2015\_4 에 해당한다.

4-3-2. 아래 명령을 입력하여 rootfs 구성 메뉴를 실행한다.

```
[host]$ petalinux-config -c rootfs
```

4-3-3. 아래쪽 화살표 키를 눌러 메뉴를 Apps 로 스크롤 한다.

4-3-4. 엔터 키를 눌러 Apps 하위 메뉴로 이동한다.  
새 응용 프로그램 cgi-led 가 메뉴에 나열 된다.

4-3-5. cgi-leds 로 스크롤 하고 <Y> 키를 눌러 응용 프로그램을 선택한다.

4-3-6. 메뉴를 종료하고 <Yes> 를 선택하여 새 구성을 저장한다.  
구성 변경 사항이 적용되는데 몇 초가 걸린다.  
명령 콘솔에서 쉘 프롬프트로 돌아갈 때까지 기다린다.

4-4. 이미지를 빌드한다.

4-4-1. 아래 명령을 입력하여 이미지를 빌드한다.

```
[host]$ petalinux-build
```

빌드 프로세스가 완료되고 이미지가 작성되도록 한다.

4-5. 네트워크를 통해 새로운 임베디드 리눅스 이미지를 부팅한다.

4-5-1. 리셋 버튼(BTN7)을 눌러 재부팅을 시작한다.

4-5-2. GtkTerm 창에 자동 부팅 메시지가 나타나면 아무키나 눌러 자동 부팅을 중지한다.

uboot 부팅 중에 "DHCP client bound to address" 메시지가 표시되지 않으면 dhcp 를 실행하여 IP 주소를 얻어야 한다.

```
U-Boot-PetaLinux> dhcp
```

4-5-3. u-boot 콘솔에서 아래 명령을 실행하여 TFTP 서버 IP 를 Host IP 로 설정한다.

```
U-Boot-PetaLinux> set serverip 192.168.1.1
```

4-5-4. u-boot 콘솔에서 아래 명령을 실행하여 TFTP 를 사용하여 새 이미지를 다운로드하고 부팅한다.

```
U-Boot-PetaLinux> run netboot
```

이 명령은 Host 의 /tftpboot 에서 ARM Cortex-A9 MPcore 시스템의 주 메모리로 image.ub 파일을 다운로드하고 이미지로 시스템을 부팅한다.

4-6. led.cgi 프로그램을 실행한다.

4-6-1. 보드가 재부팅되면 로그인하여 httpd 서비스를 시작한다.

```
# httpd -p 8080 -h /home/httpd
```

4-6-2. Host 의 웹 브라우저에서 보드를 다시 가리킨다.

http://<게시판의 IP>

부팅 메시지의 끝 부분에 보드의 IP 주소가 표시되거나 ifconfig 명령을 사용한다.

4-6-3. 새로운 led.cgi 응용 프로그램의 경로를 포함하도록 URL 을 수정한다.

http://<게시판의 IP>:8080/cgi-bin/led.cgi

4-6-4. 사용 가능한 여러 GPIO 의 ID 번호를 표시하려면 아래 명령을 입력하라.

```
# ls /sys/class/gpio
```

ID 번호 898 은 LED 에 해당한다.

ID 번호는 사용한 SD 카드 이미지에 따라 다를 수 있다.

4-6-5. 브라우저에서 LED GPIO ID 필드에 898 을 입력한다.

4-6-6. 웹 페이지에서 ON/OFF 를 클릭하고 Board 와 웹 페이지에서 어떤 일이 일어나는지 살펴본다.



Figure 10. Providing the LED GPIO ID and turning ON/OFF the LEDs

4-6-7. 모든 LED 를 끄려면 clear 버튼을 클릭한다.

4-6-8. 작업이 완료되면 보드의 전원을 끈다.

## Conclusion

이 Lab에서는 아래를 수행하는 방법을 배웠다:

- 텔넷을 사용하여 Linux 시스템에 로그인한다.
- ftp를 사용하여 파일 전송
- NFS를 사용하여 개발 시스템을 Linux Target에 탑재한다.
- 완전히 새로운 이미지 파일을 업데이트하고 다운로드하지 않고 NFS 마운트를 통해 직접 Linux 응용 프로그램 실행하기
- 내장된 웹 서버에서 서비스할 수 있도록 간단한 정적 HTML 페이지를 만들고 수정한다.
- 웹에서 실행되는 간단한 응용 프로그램을 Linux Target에서 실행하는 방법을 설명한다.

## Completed Solution

솔루션을 실행하려면 labsolution/lab4/SDCard 디렉토리의 BOOT.bin을 SD 카드에 복사한다.  
SD 카드를 Zybo에 넣는다.  
SD 카드 부팅 모드에서 Zybo를 설정한다.  
이더넷 케이블을 사용하여 Zybo를 Host 컴퓨터에 연결한다.

아래 명령을 실행하여 Host에서 DHCP 서버를 시작한다.

```
[host]$ sudo service isc-dhcp-server restart
```

labsolution/lab4/tftpboot 디렉토리의 image.ub 파일을 /tftpboot 디렉토리로 복사한다.

보드의 전원을 켜다.  
터미널 세션을 설정한다.  
자동 부팅 메시지가 표시되면 부팅 프로세스를 중단한다.  
Target Board 터미널 창에서 아래 명령을 사용하여 server ip 주소를 설정한다.

```
#set serverip 192.169.1.1
```

netboot 명령을 실행한다.

```
#run netboot
```

시스템에 로그인하고 Lab을 테스트한다.

# References

<https://www.xilinx.com/support/university/vivado/vivado-workshops/Vivado-embedded-linux-zynq.html>