TEXAS INSTRUMENTS

AM3517 Half-Day Workshop

Lab Exercises

Revision 1.0.7
September 2010

Technical Training Organization (TTO)

Notice

Technical Training Organization
Texas Instruments, Incorporated
7834 Churchill Way, MS 3984
Dallas, TX 75251-1903
(972) 917-3889

Training Materials Wiki

The training materials used for this workshop can be found at:

`http://processors.wiki.ti.com/index.php/AM3517_On-line_Workshop`

Note: Throughout this lab handouts document we use the phrase *AM3517 EVM* to denote the target board we are working with. In fact, most of you will be working with the *AM3517 Experimenter Board* with the optional LCD module attached. The *Experimenter Board* is actually a subset of the *Evaluation Module* (*EVM*); the *EVM* kit contains an additional board which provides additional ports, connectors, and options. Either kit may be used to complete these exercises.

Hands-on Labs Overview

These hands-on exercises are broken into four distinct labs:

Lab 1: Connect and boot the AM3517 EVM

Lab 2: Start Gnome (Linux GUI) desktop running on the AM3517 EVM
There are four parts to this lab which correspond to the four different ways that you can invoke Gnome

a. *Automatically* start GNOME on the AM3517 EVM's LCD
b. *Manually* start GNOME on the AM3517 EVM's LCD
c. Display the GNOME GUI *remotely* on the host PC over a secure SSH connection
d. *Export* the GNOME GUI onto the *host* PC

Lab 3: Running video (and audio) on the AM3517

a. Play a .wmv video movie trailer
b. (Take Home Lab) View video from a USB Webcam
c. (Take Home Lab) Listen to audio via USB speakers

While everyone should perform the first video exercise (Part A), the webcam & audio lab exercises are provided as a take-home-work. This is due to the fact that we were not able to provide USB webcams and speakers in this venue.

Lab 4: Investigating Linux Networking with the AM3517

a. Configure Linux Ethernet connection, both temporarily and permanently
b. Configure a Samba Server
c. Configure an HTML Server

Detailed Table of Contents

Lab 1 – MMC Linux Boot

The AM3517 can be booted up in a variety of ways: via the network (TFTP for kernel, NFS for root filesystem); using the on-chip NAND flash; or, placing the kernel and root filesystem on a SD/MMC card and placing this in the EVM card slot.

This lab exercise uses the latter method, which has fast become the most popular method for booting the board.

Hint: Originally we planned to use a `.dd` image file to create an exact replica of our working SD card. Unfortunately, we found problems when applying `.dd` files to different brands of SD/MMC cards.

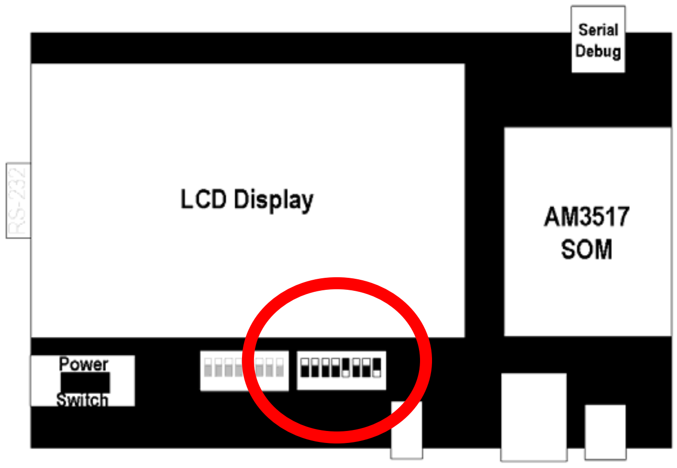
We found that using the following script (`build_sd.sh`) to be more reliable. This script formats the SD/MMC card with two partitions, then extracts the appropriate files (from two `tar.gz` files) to each of the partitions.

Verify Hardware Setup

1. **Verify that EVM's *Switch Bank 7* is configured for MMC card boot.**

Setting the first and fourth switches on, while the others are off, tells the board to boot using the MMC card.

```
sw 7-1: on
sw 7-2: off
sw 7-3: off
sw 7-4: on
sw 7-5: off
sw 7-6: off
sw 7-7: off
sw 7-8: off
```



These switches modify the boot mode pins on the AM3517, which are used by the ROM bootloader (1st stage) to use the XLOADER (2nd stage bootloader) found on the first partition of the MMC card. If all switches are off, the device will boot using the XLOADER found in the EVM's onboard NAND flash.

To learn more about the switches (and configuration) of the AM3517 board, visit:

`http://processors.wiki.ti.com/index.php/GSG:_AM35x_EVM_Hardware_Setup`

2. **Verify that a multimedia card (MMC) is inserted into the SD/MMC card reader.**

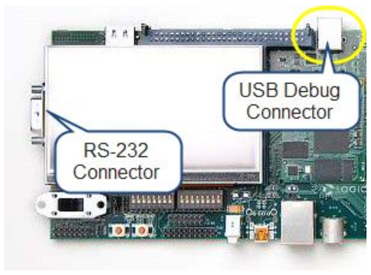
Note that because the SD/MMC card reader is mounted into the underside of the AM3517 EVM, the label on the MMC card makes the card appear to be inserted upside down.

The MMC card slot is spring loaded. It should lock into place when inserted. (Note, you can remove it by gently pressing in on it and letting it pop out slightly – but leave it in for now.)

Establish a Serial Connection

3. **Make a serial connection between your host computer and the AM3517 EVM.**

   The lack of RS-232 ports on many modern days PC's forces us to provide two options in this step. If your PC has this port, using a standard RS-232 cable will be the easiest method to control the AM3517 EVM. If not, the AM3517 EVM is equipped with an on-board USB to serial adaptor.



- If you happen to have a PC with a standard RS-232 serial port, connect a cable between your host and the RS-232 connector on the AM3517. (Skip **Step 4**)

  - OR –

- If your PC does not have an RS-232 port, connect a standard USB A/B cable between the PC and the USB ***Debug*** port on the AM3517. (Complete **Step 4**)

4. **Verify Serial over USB COM port mapping**

   If using a USB to RS-232 connection, follow these steps to verify the USB COM port mapping.

   - Start → Control Panel → System
   - Under Hardware tab, left click "*Device Manager*" button
   - Expand "*Ports (COM & LPT)*"
   - Locate "*USB Serial Port (COMx)*"

   (Optional) Set USB COM port mapping

   - Right click "*USB Serial Port (COMx)*" and select "*Properties*"
   - Under "*Port Settings*" tab, left-click "*Advanced…*" button
   - Select a COM port from "*COM port number*" drop-down

5. **Start Tera Term serial console using the desktop shortcut**

   Since the PC's provided by TI for this Hands-On workshop USB serial emulation, the COM port usually maps to COM Port 4 (but use whatever was mapped from steps 3 and 4). To that end, we've pre-configured the Tera Term desktop icon to automatically configure the following settings.

   Tera Term serial port settings are managed via "*Setup→Serial Port…*":

Port:	COM4 (select the port identified in steps 3 and 4)
Baud:	115,200
Data:	8 bit
Parity:	None
Stop Bits:	1
Flow Control:	None

   Ensure these settings above match what you see…

Powering and Booting the EVM

6. **Power on AM3517 EVM … and hit any key on your PC keyboard when prompted.**

   You should see the following text displayed on the Terra Term serial console:

```
Texas Instruments X-Loader 1.44 (Dec  8 2009 - 22:58:46)
Starting X-loader on MMC
Reading boot sector

213144 Bytes Read from MMC
Starting OS Bootloader from MMC...
Starting OS Bootloader...

U-Boot 2009.08 (Mar 05 2010 - 09:29:45)

AM35xx-GP ES1.0, L3-165MHz
am3517evm board + LPDDR/NAND
I2C:   ready
DRAM:  256 MB
NAND:  512 MiB
*** Warning - bad CRC or NAND, using default environment

In:    serial
Out:   serial
Err:   serial
I2C read: I/O error
HECC U20: port before = 000000FF
I2C read: I/O error
I2C read: I/O error
HECC U20: programmed CAN_STB low
I2C read: I/O error
HECC U20: port after = 000000FF
Die ID #1f520000000000001543b210200601d
Net:   davinci_emac_initialize
Ethernet PHY: GENERIC @ 0x00
DaVinci EMAC
Hit any key to stop autoboot:  (Countdown from 5)
```

   Make sure that you have focus on the Terra Term window by left-clicking the mouse within the display window, then:

   `Press any key to halt u-boot before the countdown reaches 0`

Note: If you let the board boot beyond the 5 secconds, simply power-cycle the board and hit any key when prompted.

7. **Verify u-boot bootcmd environment variable setting using the printenv command**

```
(u-boot) #  printenv bootcmd
```

```
bootcmd=if mmc init; then if run loadbootscript; then run bootscript;
else if run loaduimage; then run mmcboot; else run nandboot; fi; fi; else
run nandboot; fi
```

The U-BOOT bootloader (3rd stage) uses a variable called ***bootcmd*** to indicate how Linux should be configured to start.

The default setting (shown above) for *bootcmd* automatically runs a U-BOOT macro after counting down from 5.

This macro tests if there is an SD/MMC card present ('if mmc init'). If there is, it runs the "mmcboot" macro. Else it runs the "nandboot" macro. *(There is a little more to this command, but those are the two important points.)*

8. **Boot Linux on the AM3517 EVM.**

Now that we've verified our boot settings, we can boot the board by:

```
(u-boot) #  run bootcmd
```

or

```
Power cycle the board and don't stop U-BOOT's countdown
```

You should see progress reports on the serial console as Linux boots. A few warnings and/or failures are normal … as long as Linux continues and completes the boot process.

9. **Log into Linux via Tera Term serial console.**

```
Username: root
Password: (press return)
```

Lab 2 – Start GNOME Desktop on AM3517 EVM

The Angstrom Linux distribution that we are running was built with a GNOME-based GUI desktop enabled, therefore, the easiest way to start GNOME on the AM3517 EVM is to not disable it in U-BOOT. (This is Lab 2a.)

Labs 2b, 2c, and 2d explore various other ways to enable the GNOME desktop … even if you don't have a keyboard, mouse, and display hooked up to your EVM.

Lab 2a – Start GNOME on the AM3517 EVM LCD

While Part A is a good place to begin for understanding how to start the Linux GNOME graphical desktop, it's not really that practical when using a small LCD. Furthermore, you won't be able to do anything with the desktop when it boots without a powered USB hub, keyboard, and mouse (which we haven't provided in this workshop).

Note: If your AM3517 board does not contain and LDC display, skip to Lab 2b.

10. (Optional) **Connect USB powered hub/keyboard/mouse to your AM3517 EVM.**

This step is optional as most of you probably don't have these items at your workstation. If that's the case, just skip this step.

AM3517 EVM may not provide sufficient power over USB to properly operate a keyboard, mouse or keyboard/mouse paired device. Please be sure to always connect USB devices to the AM3517 EVM via a powered USB hub.

11. Power cycle the board and modify the U-BOOT bootargs to enable GNOME desktop.

Power cycle and press any key to halt U-BOOT.

```
(u-boot) #  printenv mmcargs

mmcargs=setenv bootargs console=\${console} root=/dev/mmcblk0p2 rw
rootfstype=ext3 rootwait text
```

These are the arguments that are sent to the Linux kernel when it is booted. The final argument (`text`) tells the kernel to boot in text mode; that is, not to start the GNOME desktop. To boot the GNOME desktop, you only need to remove this final argument by redefining the *mmcargs* macro:

Delete this

```
(u-boot) #  setenv mmcargs setenv bootargs console=\${console}
            root=/dev/mmcblk0p2 rw rootfstype=ext3 rootwait ~~text~~
```

Note: This should be one contiguous string. That is, there should only be a space between:
`{console}` and `root=`
You should not use a return character when setting U-BOOT variables.

Hint: Since strings like this can be hard to enter – without typo's, that is – we have created a TeraTerm macro to help with this step.

`Control -> Macro` *then select* **`am3517_lab2a.ttl`**

12. Boot Linux.

```
(u-boot) #  run bootcmd
```

You should be presented with a GNOME desktop login screen on the AM3517 EVM LCD.

Note: Without an external keyboard and mouse connected via a powered USB hub, you actually won't be able to log in. Powered USB hub is required since the EVM may not supply sufficient power over USB to run a mouse and/or keyboard.

Hint: **SideNote:** For those familiar with u-boot, this version has been compiled with CONFIG_ENV_IS_NOWHERE option which disables saving of environment to NAND.

This forces the default environment stored in the u-boot binary to guarantee MMC boot works properly (even on boards that have an environment saved in their NAND).

As such, the *saveenv* command is disabled and you can only make environment changes in RAM; that is, on a one-per basis.

13. (Optional) **If you've connected a USB hub/keyboard/mouse (per Lab 2a : Step 10), you may login.**

```
User:     root
Password: (press return)
```

Note: USB hub, keyboard and mouse may work improperly if hot-plugged. Please connect these devices before power cycling the board as per Lab 2a, step 11

Lab 2b – Manually start GNOME on AM3517 EVM

In this exercise, we'll leave the ***text*** argument in the boot settings and start the GNOME desktop manually.

14. Power cycle and log into your EVM board -- without the GNOME desktop running.

Do not process a key to stop the boot process this time

As per the Side Note above (Lab2a : Step 12), changes made to U-BOOT arguments are never saved back to Flash. Therefore, the change we made deleting text is wiped out by power-cycling the EVM.

As a reminder, **do not** edit the mmcargs as you did in Lab 2a..

15. Start the x11 server on the AM3517 EVM.

```
(am3517-evm) #  startx &
```

Notice the trailing ampersand (&) in our command above. This will start the *x server* in the background, thus allowing you to continue to enter commands in the Terra Term terminal.

If you forget the ampersand, you can press *ctrl-c* in the Terra Term terminal to exit from the startx command, then retype the command with the ampersand.

Note: If you don't get the prompt back, just hit the <enter> key again.

16. Start the GNOME session on the AM3517.

Think of your display in a client/server fashion. The startx command starts an X11 display server running. Then, we start GNOME and tell it to display the data to our local display.

```
(am3517-evm) # gnome-session --display=localhost:0.0
```

17. Once you've had enough, stop the GNOME display session

```
Press Ctrl-C in Terra Term window to end gnome session
```

Bottom Line

Lab2a and Lab2b accomplish the same thing. The first starts GNOME automatically (by removing the ~~*text*~~ boot argument). The second starts the X11 server and GNOME session manually.

While the automated version may be a bit easier, the manual method (Lab 2b) makes it easier to compare to the remote GNOME desktop lab (lab 2c) coming up next.

Lab 2c (optional) – Start GNOME on AM3517, x11 via ssh

What if you didn't have a display, keyboard, or mouse? Wouldn't it be handy to show the AM3517's display on your host computer, thus allowing those of you without a keyboard/mouse the ability to interact with the EVM.

In this part of the lab exercise, we'll boot up a command-line version of Ubuntu on our PC. We've provided a Linux VMware image, so you won't have to take the time to setup Ubuntu yourself.

Note, Labs 2c and 2d ask you to temporarily configure some network settings. We need to do this so that the displays can be exported properly. Lab 4 further explores these configurations setting, including how to make these settings permanent.

Get Ubuntu Running on your PC

18. Launch *ubuntu-x11-client* virtual machine.

```
Start VMware

Open the C:\vm_images\Ubuntu_x11_client.vmx

Click:     Power on this virtual machine (green play arrow)
or choose: VM → Power → Power on (ctrl-b)
```

Hint: To type into Ubuntu (i.e. VMware image), you may have to click inside the VMware window first.

To get control of the mouse again, as noted in the lower left-hand corner of the VMware window, just press <Ctrl> + <Alt> simultaneously.

19. Login to virtual Ubuntu Linux machine.

```
User:     user
Password: (press return)
```

20. Set a static IP address of `192.168.1.10` for the *Ubuntu x11 client* virtual machine.

```
(x11-client) #  sudo ifconfig eth0 192.168.1.10
When prompted for a sudo password, press return
```

The *sudo* command is used to execute with root permission. As a security measure, Ubuntu does not allow logging in as **`root`**, so *sudo* must be used for system configuration commands.

Note: If you are running this lab exercise with the Ubuntu x11 client connected to the network via a router or some other DHCP-enabled device, you can dynamically acquire an IP address using the command:

```
(x11-client) #  sudo dhclient eth0
```

On the AM3517 EVM

21. Power cycle and log into your EVM board -- without the GNOME desktop running.

Move to the AM3517 evaluation board's Terra Term terminal to execute the following steps.

22. Set a static IP address of `192.168.1.20` for the AM3517 EVM.

```
(am3517-evm) #  ifconfig eth0 192.168.1.20
```

Notes:

- *sudo* isn't needed on the AM3517 since you usually log into the target board as `root`.
- Similar to Ubuntu on our x86 PC, you can dynamically acquire an IP address for the AM3517 EVM using the command: `dhclient eth0`

On the Ubuntu x11 Client Virtual PC

Now that both systems are running, let's run GNOME remotely over an xterm connection (we'll be using ssh) … from the AM3517 to the PC running Ubuntu.

First we'll start X windows and the terminal in Ubuntu, then kick off the GNOME session on the AM3517 EVM.

23. Launch the X windows server with a client terminal (xterm).

```
(x11-client) #  sudo MYVAR=yes startx
```

The environment variable *MYVAR* is not a standard Linux variable. Rather, the Xsession script has been modified to allow the option of starting with or without an xterminal. The above command represents the behavior of the unmodified script.

Hint: If interested, you can examine it by using the Linux *cat* command:

```
cat /etc/X11/Xsession/99x11-common_start
```

24. Form a secure shell (ssh) tunnel from Ubuntu Linux on virtual machine to Linux on AM3517.

```
(x11-client) #  sudo ssh –X root@192.168.1.20
When prompted for password, press return
```

Make sure you use a capital "X" in the ssh command.

You may get the following warning. If so, ignore it (i.e. enter yes to continue).

```
The authenticity of host '192.168.1.100 (192.168.1.100)' can't be
established.
RSA key fingerprint is 76:72:6d:bd:57:31:00:f4:cc:a3:4d:d6:f4:9e:f1:c0.
Are you sure you want to continue connecting (yes/no)? yes
```

Launching GNOME from the AM3517 EVM

25. Examine the `DISPLAY` environment variable.

In Ubuntu →

```
(root@omap3517-evm) #  echo $DISPLAY
```

You should be returned:

```
localhost:10.0
```

Note, if you reconnect multiple times, you may get "*localhost:11.0*" or similar. This is a virtual DISPLAY which is used to forward X11 over the ssh connection.

26. Launch the GNOME session.

Again, in Ubuntu →

```
(root@omap3517-evm) #  gnome-session --display=$DISPLAY
```

It might take a minute or so for the display to show up on the Ubuntu PC. Once it's up try poking around GNOME to see it in action. What you are seeing is being generated by the AM3517 EVM, and remotely displayed on Ubuntu.

27. (Optional) **If desired, change the size of the virtual machine's window.**

This can be done by left-click dragging the bottom right hand corner of the virtual machine window to resize. The resolution will be displayed as the window size is changed.

If you wish to match the size of the AM3517 LCD display, drag until the window resolution is 480x272. (Or chose any size you want.)

Cleanup:

Don't execute these steps if you want to skip Lab 2d and move onto Lab 3.

Otherwise, if you want to try Lab 2d, then close out this lab to cleanup the environment.

28. Press Ctrl-C inside the xterm window (labeled root@x11-client) to exit GNOME.

Make sure that you have focus within this window before pressing Ctrl-C. You may have to press return to get a new xterm prompt.

```
(root@x11-client) #  Ctrl-C
```

The background of the GNOME session will be retained, but the icons and pull-down menus should be removed. Only the single xterm window (black background) is now active.

29. Close secure shell (ssh) session.

```
(root@omap3517-evm) #  exit
```

You should see the message:

```
Connection to 192.168.1.20 closed
```

30. Exit from X window session.

```
(root@x11-client) #  exit
```

Lab 2d – Start GNOME on AM3517 evm, exported display

This lab accomplishes something similar to the previous one: The AM3517 will create a display that will show up on the Ubuntu PC. Rather than transporting the data remotely over a terminal (ssh), it will export the display to Ubuntu.

You may be able to skip these steps

If you just completed Lab 2c and have not reset the AM3517 EVM or Ubuntu x11 client virtual machine, you may skip to step 35. If you did reset one (or both) of them, we'll just need to set them back up again.

31. If Ubuntu virtual machine is not still running, then launch it.

```
Start VMware Workstation (or VMware Player)
Find and open the Ubuntu_x11_client.vmx
Start (i.e. Play) the Ubuntu VM image
```

32. Login to virtual machine Ubuntu Linux.

```
User:     user
Password: (press return)
```

33. Set a static IP address of `192.168.1.10` for the Ubuntu x11 client virtual machine.

```
(x11-client) # sudo ifconfig eth0 192.168.1.10
Press return when prompted for a sudo password
```

See the previous lab if you're using a dhcp server.

34. If the AM3517 EVM was reset …

Move to Terra Term (i.e. EVM) and set the `192.168.1.20` for the AM3517 EVM.

```
(am3517-evm) # ifconfig eth0 192.168.1.20
```

Export the GNOME desktop to the Ubuntu PC

Now that our machines are setup and ready to go again…

35. Move to the Ubuntu x11-client virtual machine terminal and …

Launch the X windows server:

```
(x11-client) #  sudo startx
```

**36. Now run GNOME on the AM3517 EVM via the Terra Term terminal …
and start a GNOME manager.**

```
(am3517-evm) #  gnome-session –-display=192.168.1.10:0.0
```

Just as before, you can resize the screen and play around with GNOME.

Cleanup:

Don’t execute these steps if you’re moving onto Lab 3.

- **Are you sure you want to go through this cleanup?**
  If you are doing lab 3 next, you should not be doing the remaining steps in this lab, but instead progressing to lab 3.
- The following cleanup steps are only necessary if you want to go back and retry: Lab 2a, Lab 2b or Lab 2c before progressing to lab 3.
- Press Ctrl-C within the AM3517’s Terra Term window to exit from the GNOME session
- Press enter within the AM3517 evaluation module Terra Term window to get a Linux prompt

Choosing a GNOME display method

Before proceeding to lab 3, choose the method of lab 2a, 2b, 2c or 2d to view the GNOME display of the AM3517 EVM.

In order to utilize the method Lab 2a or 2b methods, you will need an LCD, USB hub, USB keyboard and USB mouse.

We recommend the Lab 2d method for those without external LCD, keyboard, or mouse. Direct X11 over TCP is preferred to X11 over ssh because the latter method requires an extra ssh terminal to be maintained, whereas Lab 2d provides the exact look and feel of running the GNOME display on the AM3517.

Lab 3 – Video and Audio on the EVM

There are three parts to Lab 3. We expect that you will be able to get the first part (A) running in this session, while parts (B) and (C) are provided as optional, *take-home* lab exercises. This is because the latter two parts require a USB webcam and USB audio I/O devices which are not available in most venues. Hopefully you will have such items back at your office or home, so you have something to look forward to.

Lab 3a – View .wmv video clip using mplayer

This exercise doesn't require any additional hardware. It makes use of whatever display used by the GNOME desktop you (left) setup from Lab 2.

Play Video within exported GNOME

37. Ensure that you have the GNOME display manager running.

If you don't have GNOME running, please go back and re-run Lab 2c or Lab2d.

38. Open an xterm (X11 terminal) window within the GNOME desktop.

With mouse, select *terminal* from GNOME desktop menus:

```
Applications → Accessories → terminal
```

39. Launch *mplayer* and view the movie trailer for "*The Crow*".

We copied *The Crow's* movie trailer (in .wmv format) into the filesystem for you. There's nothing special about this movie clip; in fact, you could even add your own clip using any format Linux *mplayer* understands.

```
(root@omap3517-evm) #  mplayer -nosound thecrow_trailer.wmv
```

To view full-screen, include the appropriate mplayer command line option (-fs):

```
(root@omap3517-evm) #  mplayer -nosound -fs thecrow_trailer.wmv
```

Note: Since there isn't audio on the board (and hence no driver) use –nosound or the playback will fail.

Play Video Directly on LCD

40. Power cycle and log into your EVM board -- without the GNOME desktop running.

Move to the AM3517 evaluation board's Terra Term terminal to execute the following steps.

41. In the Tera Term serial console, start the X11 display server on the AM3517 EVM.

```
(am3517-evm) #  startx &
```

42. Again, in the Tera Term serial console, start mplayer on the local display

```
# DISPLAY=localhost:0.0 mplayer -nosound thecrow_trailer.wmv
```

Lab 3b – (Take Home Lab) View a USB webcam image

Note: This lab exercise requires an externally connected powered USB hub and USB camera.

This exercise uses an external USB webcam to generate video to be displayed by the AM3517 board. The webcam implements the standard Linux V4L2 video driver which can be displayed by the Linux *mplayer* application.

Hint: **Hint:** The following lab steps have been tested with the Logitech C200 (and Logitech C250) USB webcam.

In General, the make and model of the powered USB hub should not be important al long as it's capable of supplying the power required by the USB webcam, since all USB hubs should use the same Linux driver. However, the Linux webcam drivers can vary widely for different USB webcams. (There are more than 20 independent webcam drivers built into the Linux kernel you are using.)

43. Once again, ensure that you have the GNOME display manager running.

If you don't have GNOME running, please go back and re-run Lab 2c or Lab2d.

44. Open an xterm (X11 terminal) window within the GNOME desktop.

With mouse, select *terminal* from GNOME desktop menus:

```
Applications → Accessories → terminal
```

45. Make sure the Logitech C200 USB webcam is plugged in.

If you have not already plugged the Logitech webcam into the powered USB hub, please do so at this time.

46. Verify that Linux has recognized and mounted as a USB device for your webcam.

```
# lsusb
```
(that's the small letters L S USB)

You should see an entry for your camera such as:

```
Bus 001 Device 003: ID 046d:0802 Logitech, Inc.
The listed ID is a four digit (hex) ID for vendor and product (colon
separated). This ID is used to determine the proper driver to attach to
the USB device.
```

47. Verify that the webcam has been associated with the video-over-usb driver.

Linux maintains a great deal of system information in the `/proc` directory of its filesystem. In our case, we'd like to view the list of various USB devices that have been registered by Linux, which conveniently is one of the items maintained within `/proc`.

For those of you not familiar with Linux, cat is a standard command which outputs (text, in our case) to the standard display (our terminal command line).

```
# cat /proc/bus/usb/devices
```

Hint: The Linux `cat` command is the Unix command to list a file's contents onto your screen. The name *cat* command comes from the word concatenate, as in you're concatenating files to standard output (i.e. display).

48. Verify that a V4L2 device node has been created for the video-over-usb driver.

Next we'll list (Linux's `ls` command) the various Linux drivers that begin with the name "video".

```
# ls /dev/video*
```

On the Linux distribution you are using, there are two device nodes created by default:

```
/dev/video1  :  non-usb video input
/dev/video2  :  non-usb video output
```

If the webcam has been recognized and associated to a driver, you will also see:

```
/dev/video0  :  usb webcam
```

Hint: The term *node* (or *device node*) refers to the virtual filename created for a driver within Linux. So, the webcam driver can often be referred to by name (in this case `/dev/video0`) just like any file within the filesystem.

While most device nodes are placed into the `/dev` directory of the filesystem, this is not a requirement within Linux. It's just the most common place to put them.

49. Use mplayer to view the webcam video stream

```
# mplayer tv:// -tv driver=v4l2:width=320:height=240 -fps 30 -nosound
```

Then, to exit:

```
Press Ctrl-C to exit
```

Lab 3c – (Take Home Lab) Attach USB speakers

You got video passing through your Linux system in Lab 3b. This exercise adds to that by passing audio as well. As there are no audio ports on this particular EVM board, we utilize external USB speakers to output the sound from the .wmv video file.

Note: This lab exercise requires an externally connected powered USB hub and USB speakers.

50. We assume here that you have already completed Lab 3b.

Therefore, your GNOME desktop, terminal window within GNOME are already setup and working.

51. Make sure your USB speakers are plugged into the powered USB hub.

Next, if the USB speakers are not already plugged into the powered USB hub, please do so at this time.

52. Verify that the speakers have been recognized and mounted as a USB device.

```
# lsusb
```

You should see an entry for your device, such as:

```
Bus 001 Device 004: ID 08bb:2704 Texas Instruments Japan
The listed ID is a four digit (hex) ID for vendor and product (colon
separated). This ID is used to determine the proper driver to attach to
the USB device.
```

53. Verify that the device has been associated with the USB audio driver.

```
# cat /proc/asound/cards
```

You should see an entry such as:

```
2 [default  ]: USB-Audio - USB Audio DAC
               Burr-Brown from TI USB Audio DAC at usb-ehci-omap.0-1.2, full
```

The first number indicates the card number that has been associated with the speakers.

54. Execute mplayer using –ao to route audio to the USB speakers

```
# mplayer thecrow_trailer.wmv –ao alsa:device=hw=2.0
```

Here the setting `hw=2.0` indicates that the Linux audio driver (i.e. ALSA driver) will be associated to: `Card 2, Device 0`. With this setting, the driver will use the device node:

```
/dev/snd/pcmC2D0p
```

Lab 4 – Explore Linux Networking

This lab exercise assumes a GNOME desktop is running on the AM3517 EVM (per Lab 2).

Linux provides a rich networking environment. It's one of the things that attracts many of us to this operating system. Our Angstrom Linux distribution contains the various networking components we'll be playing with here.

This lab exercise explores just a few of these networking capabilities.

- We'll begin by exploring command-line IP network configuration within Linux. *(We have actually completed a couple of these steps in making our earlier labs work. In this exercise, we'll explain what we did and learn how to make our changes permanent.)*
- Once configuration is complete, we'll move on to setting up a Samba server so that we can share files between Linux and Windows.
- Finally, we'll setup an HTTP webserver. This is a common way your embedded device can easily share information to other components/users of the system.

Lab 4a – Configure Ethernet Connection

1. **Ensure that you have the GNOME display manager running.**

   If you don't have GNOME running, please go back and re-run Lab 2c or Lab2d.

2. **Open an xterm (X11 terminal) window within the GNOME desktop.**

   With mouse, select *terminal* from GNOME desktop menus:

   ```
   Applications → Accessories → terminal
   ```

Temporary static IP configuration

Don't do this step.

You've actually done it already. If you do it right now, it'll freeze the display since it will interrupt your remote Gnome session.

We recommend reading this step and moving onto the next step.

3. **In the xterm window, configure the Ethernet connection with a static IP address.**

   This is the step that we already used in an earlier lab. Remember, though, that configuring the Ethernet address in this way is only temporary.

   ```
   (root@omap3517-evm:~) #  ifconfig eth0 192.168.1.10
   ```

   Remember, this configuration will remain only until the AM3517 EVM is power cycled (or until it's reconfigured).

Note: Your board may freeze at this point. If you choose a different IP address, or if this step somehow disrupts the X11/GNOME connection, the desktop may freeze. If this happens, just reboot the AM3517 board, recreate the GNOME session, and start with the next lab step.

Hint: As we showed earlier in Labs 2c, if you are running this lab exercise connected to the network via a router or some other DHCP-enabled device, you can dynamically acquire an IP address using the command:

(root@ am3517-evm:~) # dhclient eth0

Permanent static IP configruation

4. **Verify that Network Manager will honor: `/etc/network/interfaces`**

```
# gedit /etc/NetworkManager/nm-system-settings.conf
```

Verify that the *ifup/ifdown* setting reads as follows:

```
[ifupdown]
 managed=false
```

This will ensure that Network Manager does not manage the *ifup/ifdown* process and instead honors the settings in the `/etc/network/interfaces` file. If this is instead set to *true*, Linux Network Manager will attempt to acquire a *dhcp* address and will only honor the settings in `/etc/network/interfaces` as a fallback if the *dhcp* request fails.

5. **Reconfigure the connection permanently by editing the `eth0` configuration file.**

Since we now are sure the ifupdown managed is false, we can control the IP address by modifying the config file for eth0.

```
# vi /etc/network/interfaces
```

Scroll down to you find *"auto eth0"*, and then change that the configuration script to reads as follows:

```
auto eth0
iface eth0 inet static
                 address 192.168.1.39
                 network 192.168.1.0
                 netmask 255.255.255.0
                 broadcast 192.168.1.255
```

In VI:
"x" key to delete a character
"r" to replace
"i" to insert a character

Close `/etc/network/interfaces` file when the changes are complete.

Esc

```
: wq
```
(where w is to write changes, q is to quite vi)

Hint: If you were in a more complex system (instead of connected via crossover), you could configure a gateway with:

```
gateway <your_gateway_address_here>
```

Hint: If instead of a static configuration you always wished to receive an IP address dynamically from a DHCP device such as a router, replace the above with:

```
auto eth0
iface eth0 inet dhcp
```

6. **Reboot the AM3517 board, start GNOME, verify new Ethernet address.**

You can check the Ethernet IP address using the *ifconfig* command:

```
# ifconfig eth0
```

Lab 4b – Configure a Samba Server

In the exercise, we will setup a *Samba* Server which should allow us to share files between our Windows and Linux environments. While VMware provides a convenient *Shared Folder* feature, it's more fitting with our networking topic to implement this with Samba.

> From Wikipedia
>
> ***Samba*** is a free software re-implementation of SMB/CIFS networking protocol ... which provides file and print services for various Microsoft Windows clients and can integrate with a Windows Server domain... (SMB = Server Message Block; CIFS = Common Internet File System)

As a side note, if we wanted to share files between two Linux systems, we would probably use NFS (network file system) as this is fairly ubiquitous feature amongst Linux and Unix operating systems. Since Windows doesn't typically support NFS, Samba is more appropriate for that task.

7. **Find the default *runlevel* for the system.**

```
# gedit /etc/inittab
```

Within this file, you should find the line:

```
id:5:initdefault:
```

This line indicates that the default *runlevel* for the system; in our case it indicates we are at `runlevel 5`.

Exit without making, or saving, changes.

8. **Verify that Samba server will start automatically under default startup profile.**

```
# ls /etc/rc5.d
```

Among the files listed you should see **`S20samba`**, which is actually a link to `/etc/init.d/samba`, the *samba daemon startup script*.

9. **Edit Samba configuration file to allow *null passwords*.**

```
# gedit /etc/samba/smb.conf
```

Directly after the line "[global]" enter the following:

```
   null passwords = yes
```

10. **Add *root* as a new Samba user, no password required.**

```
# smbpasswd –a root –n
```

Where `–a` indicates to add a *new user*; and, `-n` indicates *no password* required.

11. **Restart samba to make sure changes take effect.**

```
# /etc/init.d/samba restart
```

12. **View the *root* user's home directory from your windows PC.**

Let's setup access to a shared folder in Windows. An easy way to do this is to setup a mapped drive; that is, set a drive letter (say "Z") to the shared URL path.

```
Start → My Computer
Tools → Map network drive…
```

Select any free drive (default is usually `Z:`) and enter the following as the folder path:

```
\\192.168.1.39\root
```

Note: Use the address you configured for your AM3517 earlier in step 5 on page 22.

13. **Test the shared Samba path by playing video.**

We can test our ability to access the path by playing the movie trailer across the Samba share.

```
Left-click on "thecrow_trailer.wmv" from the share
and choose to play the video
```

In this case, the Windows media player should start playing the file that is located on the MMC card plugged into the AM3517 EVM, (*Note, the file is small enough that it should play OK even without copying it to the local machine.*)

Lab 4c – Configure an HTML Server

Finally, let's turn our sights to HTML web servers. We expect you're familiar with the general concepts of the web, so we won't define that here. What may be new to many of us, though, is how to setup a Web Server – even further, most of us have probably not setup a server for an embedded processor like the AM3517.

We'll begin by setting up the web server on our EVM, then test it by accessing the server from our Windows PC.

Setting up the HTTP Web Server

14. **Verify that *thttpd server* is configured to start at *runlevel 5* (default runlevel).**

    ```
    # ls /etc/rc5.d
    ```

    You should see *thttpd* listed (`S60thttpd`). This is a link to `/etc/init.d/thttpd` startup script.

15. **View the startup file to determine the server's startup options.**

    ```
    # gedit /etc/init.d/thttpd
    ```

    You should see the following line:

    ```
    start-stop-daemon --start --quiet --exec $thttpd -- -d /webfiles
    ```

    Everything after the double dash are options that will be passed to the *thttpd* application as it's started up; in the above example, *thttpd* will be passed: `-d /webfiles`

16. **What do these startup options mean?**

    Let's examine the *thttpd* help topic to see what `–d` means.

    ```
    # thttpd --help
    ```

    Within the help listing, you will see that the `–d` option specifies the ***start*** directory for the *thttpd server*; therefore, any webpage we want displayed should be placed into the `/webfiles` directory.

17. **What files are we serving up?**

    Let's examine our /webfiles directory to see what it contains.

    ```
    ls  /webfiles
    ```

    You should find a simple `test.html` webpage we created so that they would be something for our server to serve-up. Let's quickly examine our test file.

    ```
    # gedit /webfiles/test.html
    ```

    This is a very simple *hello world* html page. Feel free to spice it up a little if you are familiar with editing HTML syntax.

    ```
    Save and close the file when you are done.
    ```

Let's Test the Web Server

We'll quickly try out the AM3517 web server locally, before moving back over to and testing it from Windows.

18. View the webpage locally via Epiphany browser on AM3517 desktop.

The Angstrom Linux distribution we're running on the AM3517 contains a web browser named *Epiphany*. Let's try to display our page locally, before trying it from another computer.

```
Applications → Internet → Epiphany
```

In the "Go" box of the browser, try all three of the following URLs. (*Obviously, make sure you press enter or the go button for each, then wait for the page to show before moving onto the next.*)

```
file:///webfiles/test.html
http://127.0.0.1/test.html
http://192.168.1.39/test.html
```

(Please note the three hashes after `file:`*)*

19. View the webpage from the connected Windows computer.

Perform this step from your Windows computer – which should connected to the AM3517 target board. In our hand-on lab environment, the two boards are simply connected via an Ethernet crossover cable. (*Though, back at your desk you may have them connected via an Ethernet switch.*)

```
Start Windows Internet Explorer
```

In the address field, enter our webpage's address:

```
http://192.168.1.39/test.html
```

Note: Use the address you configured for your AM3517 earlier in step 5 on page 22.

While this webpage was overly simplistic, it belies the power that is now in your hands. You can easily create more power webpages and serve them up via the AM3517. **In fact, it's become quite common for embedded systems to use this simple approach for providing outside configuration (or status) information**.

And, if you need more features than the simple *thttp* web server application can provide, there are many others available within the Linux community. Have fun!