

Learning Report Embedded Linux



GLOBAL
ENGINEERING
ACADEMY

Genesis



L&T Technology Services



Name	PS Number	Email ID
Niranjan Kumar M	99002516	niranjankumar.m@lts.com

Contentss

CONTENTS.....	3
LEARNING OBJECTIVES OF THE MODULE.....	4
ACTIVITY 1 – SETUP ACTIVITY.....	5
ACTIVITY 2 – DIFFERENCES BETWEEN RASPBERRY PIE , DRAGON, IMX7 SABRE, BBB.....	5
ACTIVITY 3 – EVOLUTION AND CHANGES OF BEAGLE BACK BONE BOARD.....	5
ACTIVITY 4 – PIN EXPANSION HEADER OF BBB AND LOCATE THE VARIOUS PERIPHERALS OF BONE.....	5
ACTIVITY 5 – PROCESS, THREADS AND IPC.....	5
ACTIVITY 6 – MINI PROJECT.....	5

ACTIVITY 1:SETUP ACTIVITY

Step by Step Configuration of the boards and set up in Windows OS:

Step 1: Plug in the Ethernet chord to Beagle Bone Black Ethernet port (to establish communication) (or) use any other bootable options such as SD card, MMC.
Plug in the USB chord to the host machine and the other end to the micro USB port in the Beagle Bone Black (To supply power to the Beagle Bone Black)

Step 2: Installing Drivers in the Host machine
Visit <https://beagleboard.org/getting-started>
Depending on the configuration of the host machine download and install the respective USB Driver Installer. Once installed Reboot the host machine.

Step 3: Connecting to Beagle Bone Black via Ethernet.
Open any Browser (preferably Chrome or Firefox) type the below IP address in the URL
IP address: 192.168.7.2 .
Beagleboard.org Web page gets loaded which is already present in the Beagle Bone Black
Now the connection to Beagle Bone Black is successful.

Step 4: Obtaining the unique IP address
Click on the Cloud9 IDE a web page loads and select once again Cloud9 IDE this takes to the
Cloud9 IDE which is running on the Beagle Bone Black.
(If any error occurs then make sure don't use the Internet Explorer Browser)
Open new terminal in the Cloud9 IDE and type the command "ifconfig".
Note down the IP address in the eth0 section.

Step 5: Connect to the Beagle Bone Black using simple SSH client i.e., PuTTY
Open the URL : <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>
Download the putty.exe depending on the host machine configuration.

Step 6: Launch the SSH client i.e., PuTTY. Enter the IP address (Noted in the Step 4) in the Host name *IP address text field*. Click on Open. This launches a Linux Terminal window in the host machine which asks for the login credential login as: root (which is default).

Step 7: Remotely connecting to the Beagle Bone Black using TightVNC viewer.
TightVNC server is already installed in the Beagle Bone Black. But TightVNC viewer is required for the remote PC in order to access remotely. Download the TightVNC viewer using
url
link : <https://www.tightvnc.com/download.php>
Download and install the TightVNC viewer based on the configuration on windows.

Step 8: Fire up VNC sever before running the TightVNC viewer in the remote machine.

Launch putty.exe in the Host machine and login. Then install tightvncserver.

Type the “sudo apt-get install tightvncserver” command.

(Note: The below steps are to be performed only once)

Then type command “typevncserver” (Press enter)

Set Password and Verify Password.

Now type command “vncserver :1 -geometry 1280x800 -depth 24 -dpi 96” (Press enter)

Step 9: Launch TightVNC viewer in the remote machine and enter the IP address (Noted in Step 4) along

with :1 Example: 10.1.15.25 : 1

Click on connect and enter VNC password which was set in the Step 8.

A graphical user interface window pops up.

Step by Step Configuration of the boards and set up in Linux OS:

Step 1: sudo minicom -s

Step 2: serial port setup (know the TTL cable name)
(In other terminal)

Step 3: dmesg (Search for Prolific Technology) port ttyUSB0

Step 4: Press a and enter /dev/ttyUSB0

Step 5: Press e check for Standard Bod rate : 115200 8N1
8-bit
N-no parity
1-Stop bit

Step 6: Press f, Check for Hardware flow control set it to NO

Step 7: Press g, Check for software flow control set it to NO

Step 8: Save the settings as dfl (default)

Step 9: Exit

Step 10: connect usb

ANGSTROM

Step 11: beaglebone login: root

root@beaglebone:~#

Step 12: Shutdownnow

Activity 2: Differences between Raspberry pie , Dragon, imx7 Sabre, BBB

Parameter	RASPBERRY PIE	BEAGLEBONE BLACK	Dragon	imx7 Sabre
Processor Type	It uses ARM11 processor.	It uses ARM Cortex-A8 processor.	Quad-core ARM® Cortex® A53 at up to 1.2 GHz per core with both 32-bit and 64-bit support	Two Arm Cortex-A7 core OS upto 1 GHz, Single Arm Cortex-M4 CORE os
RAM	For the functioning of raspberry pi, 512 MB SDRAM is used.	For the functioning of beaglebone black, 512 MB DDR3L is used.	1GB LPDDR3 533MHz / 8GB eMMC 4.5 / SD 3.0 (UHS-I)	1 GB DDR3, 533 MHz eMMC expansion footprint.
Processor Speed	It uses 700 MHz for processing.	It uses 1 GHz for its processing.	1.2 GHz per core with both 32-bit and 64-bit support	1 GHz :Arm Cortex-A7 200 MHz :Arm Cortex -M4
Min Power	It requires a power supply of 700mA (3.5W).	It requires min power of 210mA (1.05W) for its functioning.	It requires a power supply of 8-18V 2A.	It requires 5V/5A universal power supply.
GPIO Pins	It has 12 GPIO pins.	It has 69 GPIO pins.	It has 40 GPIO pins	It has 138 GPIO pins
Dev IDE	It uses IDLE, Scratch, Squeak/ Linux to perform tasks.	It uses Python, Scratch, Squeak, Cloud9/Linux to perform a particular task.	Android 5.1 (Lollipop) on Linux Kernel 3.10	Vivante Tool v6.2.4.p4.1.7.9 and linux based systems
USB Master	It has 2 USB 2.0 on board.	It has 1 USB 2.0 on its board.	one micro USB (device mode only), two USB 2.0 (host mode only)	<ul style="list-style-type: none"> • 1 USB host connector • 1 micro USB OTG connector
Audio Output	Supports HDMI, Analog audio output	It uses Analog output for audio.	It calls for a minimum of	i.MX7 has multiple audio interfaces and

			single channel audio through two interfaces, BT and HDMI/MHL/DisplayPort	one is fully available on the SODIMM connector of the Colibri iMX7.
Video Output	It supports HDMI, Composite output for video.	No such specific video output.	1080p@30fps HD video playback and capture with H.264 (AVC), and 720p playback with H.265 (HEVC)	Its supports HDMI, composite output for video.
UART	It uses 1 UART to transmit and receive serial data.	It uses 5 UART to transmit and receive serial data.	support for one SoC UART and an optional second UART both to be routed to the Low Speed Expansion Connector.	UART via USB port
No. of I/O pins	It has 8 Digital, 0 Analog pins.	It has 65 Digital, 7 Analog pins.	It has 11 Digital ,0 Analog pins.	It has total 138 pins

Activity 3: Evolution and Changes of Beagle back Bone Board

Revision	Additions(differences)
A4	Preliminary
A4A	Incorporated the capacitors to fix the noise issue on the display
A4B	Added a 100K pull down resistor between pins 1 and 4 of J1 to fix the serial port issue
A5.1	1.Added information on Power button and the battery access points 2.Final production released version.
A5.2	1) Updated the PCB to incorporate the modification that was being done on Rev A5A. There is NO difference at all in functionality between REV A5A and REV A5B. 2) Made the LEDs dimmer for those that could not sleep due to the brightness of the LEDs.
A5.3	1. Updated serial number locations. 2. Corrected the feature table for 4 UARTS 3. Corrected eMMC pin table to match other tables in the manual.
A5.4	1. Corrected revision listed in section 2. Rev A5A is the initial production release. 2. Added all the locations of the serial numbers 3. Made additions to the compatibility list. 4. Corrected Table 7 for LED GPIO pins. 5. Fixed several typos. 6. Added some additional information about LDOs and Step-Down converters. 7. Added short section on HDMI.
A5.5	1. Release of the A5B version. 2. The LEDS were dimmed by changing the resistors. 3. The serial termination mode was incorporated into the PCB.
A5.6	1. Added information on Rev A5C 2. Added PRU/ICSS options to tables for P8 and P9. 3. Added section on USB Host 4. Correct modes on Table 15. 5. Fixed a few typos
A5.7	1. Updated assembly revision to A6. 2. PCB change to add buffer to the reset line and ground the oscillator GND pin. 3. Added resistor on PCB for connection of OSC_GND to board GND.
A6	1. Added changes for rev A6 that covered fixing of the link LED, JTAG Reset, and DHCP issue. 2. Added PRU information and two additional signals for the PRU. 3. Added write protection to EEPROM.

	<ul style="list-style-type: none">4. Fixed numbering of subsections in Section 7.05. Fixed error in Table 9 pin 23 Mode 1 should be MMC1_DAT4.6. Updated Table 7 to show the revision number in the EEPROM matches the revision of the board.7. Corrected various typos.8. Updated Battery Interface section to accurately document the LDO dropout at 200mV.9. Added SW Support section.
A6A	<ul style="list-style-type: none">1) Added optional zero ohm resistor to tie GND_OSC1 to system ground.2) Changed C106 to a 1uF capacitor.3) Changed C24 to a 2.2uF capacitor.
B	<ul style="list-style-type: none">1. Changed the processor to the AM3358BZCZ2. No changes in features or operation of the board resulted from this change.
C	<ul style="list-style-type: none">1. This revision increased the eMMC from 2GB to 4GB

Activity 4: Pin expansion header of BBB and locate the various peripherals of Bone.

https://github.com/L99002516/embedded_linux.git

the above link has been updated with the Pin expansion header of BeagleBone Black. Which has been P8 and ref excel datasheet.

Types of Pins:

1. 23 – Reconfigurable Digital pins
2. 7 -- Analog Inputs Pins
3. 2 -- Shared I2C Pins
4. 7 -- Pulse width modulation
5. 25 – Digital Pins
6. 32 – Power management Pins

Activity-5: Testing MLO and U-boot image on BBB

Step 1: Download gparted using the command

```
sudo apt install gparted
```

Step 2: Insert card reader and select /dev/sdb(gb) and open gparted and make two partitions.

Step 3: Right click on unallocated and click on New and allocate new size as per SD card and give file system as fat16 and give label BOOT.

Step 4: Right click on unallocated and then remaining space will be allocated to this partition by default. Give file name as ext3/ext4 and give label as ROOTFS.

Step 5: Right click on BOOT and select manage flags then select boot option and then close.

Step 6: Connect the RX of TTL cable to TX of BBB board, TX of TTL cable to RX of BBB board and connect the common ground.

Step 7: Create a workspace copy MLO file in BOOT and Workspace. Then unmount the SD card.

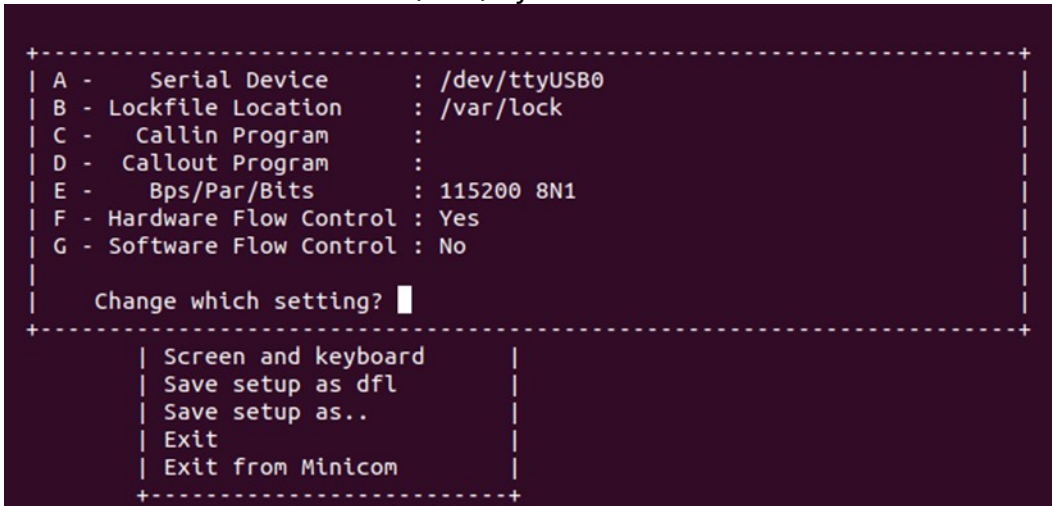
Step 8: Then connect SD Card in board and open the terminal.

Step 9: Install minicom by using the command

```
sudo apt install minicom
```

```
sudo minicom -s for setup changes
```

Then select on serial device as /dev/ttyUSB0



```
+-----+
| A -   Serial Device       : /dev/ttyUSB0   |
| B -   Lockfile Location   : /var/lock      |
| C -   Callin Program      :                |
| D -   Callout Program     :                |
| E -   Bps/Par/Bits        : 115200 8N1     |
| F -   Hardware Flow Control : Yes          |
| G -   Software Flow Control : No           |
|                                     |
|   Change which setting? █               |
+-----+
| Screen and keyboard |
| Save setup as dfl   |
| Save setup as..     |
| Exit                |
| Exit from Minicom   |
+-----+
```

Figure 1: Serial device select

- Step 10: Then enter to save and exit after saving it as the default configuration.
- Step 11: Copy the u-boot.img to the BOOT partition and repeat the step for boot-up.
- Step 12: Open the minicom again, and boot the BBB with the S2 button pressed.

Activity 6: Different booting stages in Beaglebone Black

1. ROM Bootloader (RBL)

The ROM boot loader for the BBB is located at the AM335x ROM. The AM335x have an internal RAM memory of 128KB. As soon as the power is applied to the device, the code at this location is executed. When we apply power to the SOC, it does some system level initializations and then goes for the initialization of the watchdog. The watchdog is configured for 3 minutes. If the watchdog is not fed for some time, watchdog senses some trouble and it will reset the system. So here, if even after three minutes the booting is not further connected to the SPL/MLO, the watchdog resets the system and the sequences restarts from the beginning.

After it does the watchdog initializations, it then moves on to initialize the PLL and clock configurations. PLL stands for phase lock loops. To be simple, PLLs are the one who will be providing different frequencies from a crystal input to various peripheral. In the case of BBB, we are giving a clock input of 24 MHz and we are able to generate clock frequencies of around up to 800 MHz. Then booting process is started.

In the booting, the SOC will look for the configuration of SYSBOOT pins and determine from where is the bootable image to be loaded. The AM335x have the capability of loading bootable image from various sources like the MMC interface, USB interface, UART interface etc. Based on the pins configuration and switch select, the device will then jump to the respective loader. In our case, the device will move on to the SPL/MLO.

If the device is not able to find any bootable image, the device will go into a dead loop and after three minutes, the watchdog will reset the system.

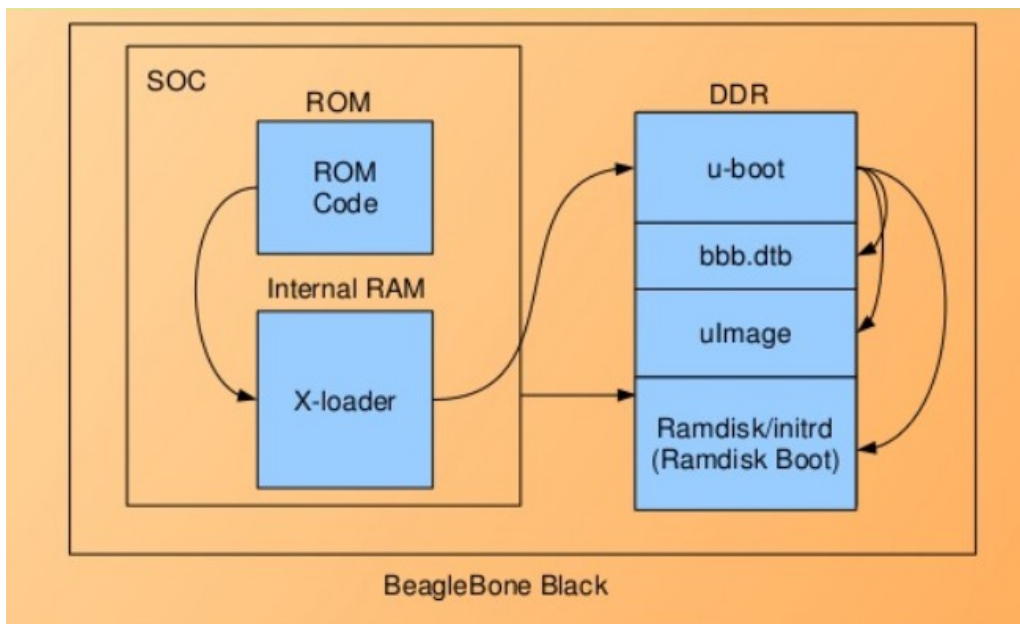


Figure 2: Booting Process

2. Secondary Program Loader/ MLO

Now, we have finished the RBL and the SPL was successfully loaded it runs out of the internal SRAM of the SOC. The SOC will do the initializations and preparations for the U-Boot that is the third stage bootloader to be executed. It is also possible to modify the PLL in order to derive a desired frequency of clock source from the second stage bootloader. It also initializes the DDR memory because the Linux Kernel is going to be executed from this memory.

The SPL additionally does an important operation known as pin muxing. Suppose the U-Boot is to be loaded from the MMC interface. Then the pins of the SOC have to be configured for supporting the MMC interface. The pins of the SOC can be used for various purposes based on the need of the user. This is known as multiplexing or muxing in the embedded systems.

```

Welcome to minicom 2.7.1

OPTIONS: I18n
Compiled on Aug 13 2017, 15:25:34.
Port /dev/ttyUSB0, 09:03:02

Press CTRL-A Z for help on special keys

U-Boot SPL 2013.04-dirty (Jul 10 2013 - 14:02:53)
musb-hdrc: ConfigData=0xde (UTMI-8, dyn FIFOs, HB-ISO Rx, HB-ISO Tx, SoftConn)
musb-hdrc: MHDRC RTL version 2.0
musb-hdrc: setup fifo_mode 4
musb-hdrc: 28/31 max ep, 16384/16384 memory
USB Peripheral mode controller at 47401000 using PIO, IRQ 0
musb-hdrc: ConfigData=0xde (UTMI-8, dyn FIFOs, HB-ISO Rx, HB-ISO Tx, SoftConn)
musb-hdrc: MHDRC RTL version 2.0
musb-hdrc: setup fifo_mode 4
musb-hdrc: 28/31 max ep, 16384/16384 memory
USB Host mode controller at 47401800 using PIO, IRQ 0
OMAP SD/MMC: 0
mmc_send_cmd : timeout: No status update
reading u-boot.img
reading u-boot.img

U-Boot 2013.04-dirty (Jul 10 2013 - 14:02:53)

I2C:   ready
DRAM:  512 MiB
WARNING: Caches not enabled
NAND:  No NAND device found!!!
0 MiB
MMC:   OMAP SD/MMC: 0, OMAP SD/MMC: 1
*** Warning - readenv() failed, using default environment

musb-hdrc: ConfigData=0xde (UTMI-8, dyn FIFOs, HB-ISO Rx, HB-ISO Tx, SoftConn)
musb-hdrc: MHDRC RTL version 2.0
musb-hdrc: setup fifo_mode 4
musb-hdrc: 28/31 max ep, 16384/16384 memory
USB Peripheral mode controller at 47401000 using PIO, IRQ 0
musb-hdrc: ConfigData=0xde (UTMI-8, dyn FIFOs, HB-ISO Rx, HB-ISO Tx, SoftConn)
musb-hdrc: MHDRC RTL version 2.0
musb-hdrc: setup fifo_mode 4
musb-hdrc: 28/31 max ep, 16384/16384 memory
USB Host mode controller at 47401800 using PIO, IRQ 0
Net:   <ethaddr> not set. Validating first E-fuse MAC
cpsw, usb_ether
Hit any key to stop autoboot:  0
U-Boot#

```

Figure 3: Booting till U-Boot

After this, the SPL checks for the U-Boot. It searches for the U-Boot image file called `u-boot.img`. After these processes, the U-Boot image is copied into the DDR memory and the control is passed to the U-Boot. This is our third stage bootloader.

The RBL could not copy the U-Boot to the internal SRAM because the size of the internal SRAM is only 128KB. So we are using a second stage bootloader to copy the U-Boot to the DDR. Also, the U-Boot cannot be loaded directly to the DDR by the RBL, because DDR is an external memory. The SOC doesn't know which DDR is being used.

3. U-Boot

Now the control have reached the third stage bootloader that is the U-Boot. This guy will load the Linux kernel to the DDR memory. In order for the U-Boot to load the Linux Kernel, we should tell the U-Boot where the Linux Kernel is located, through what interface it is accessible etc. With these information, the U-Boot will load the Linux Kernel into the DDR memory of the BBB.

We write these information in a file called uEnv.txt and the U-Boot will read this text file and find out from where it can load the Linux Kernel and to which address of the DDR memory is the Kernel to be loaded.

The U-Boot checks for a file called ulmage. UImage is actually a combination of U-Boot header, which have information about the image like the architecture, OS name etc. and something called zImage, which is actually a compressed version of Linux Kernel.

This is then decompressed and the Linux Kernel is loaded.

Activity-7: Linux boot sequence after the booting.

- By default, the ROM will boot from the MMC1 interface first (the onboard eMMC), followed by MMC0 (MicroSD), UART0 and USB0.
- If the boot switch (S2) is held down during power-up, the ROM will boot from the SPI0 Interface first, followed by MMC0, USB0 and UART0. This allows the Beagle-Bone Black to bypass the onboard eMMC and boot from the removable uSD (provided no valid boot device is found on SPI0.) This can be used to recover from a corrupted onboard eMMC.
- The AM3359 will try to load and execute the first stage bootloader called "MLO" from a Fat 12/16 or 32 bit MBR based filesystem. If using eMMC, this file is loaded using RAW mode. This means the ROM looks for a TOC at four specific offsets.
- MLO Booting (uBoot SPL Second Program Loader)

```
Welcome to minicom 2.7.1

OPTIONS: I18n
Compiled on Dec 23 2019, 02:06:26.
Port /dev/ttyUSB0, 15:16:51

Press CTRL-A Z for help on special keys

U-Boot SPL 2013.04-dirty (Jul 10 2013 - 14:02:53)
musb-hdrc: ConfigData=0xde (UTMI-8, dyn FIFOs, HB-ISO Rx, HB-ISO Tx, SoftConn)
musb-hdrc: MHDRC RTL version 2.0
musb-hdrc: setup fifo_mode 4
musb-hdrc: 28/31 max ep, 16384/16384 memory
USB Peripheral mode controller at 47401000 using PIO, IRQ 0
musb-hdrc: ConfigData=0xde (UTMI-8, dyn FIFOs, HB-ISO Rx, HB-ISO Tx, SoftConn)
musb-hdrc: MHDRC RTL version 2.0
musb-hdrc: setup fifo_mode 4
musb-hdrc: 28/31 max ep, 16384/16384 memory
USB Host mode controller at 47401800 using PIO, IRQ 0
OMAP SD/MMC: 0
mmc_send_cmd : timeout: No status update
reading u-boot.img
reading u-boot.img

U-Boot 2013.04-dirty (Jul 10 2013 - 14:02:53)

I2C: ready
DRAM: 512 MiB
WARNING: Caches not enabled
NAND: No NAND device found!!!
0 MiB
MMC: OMAP SD/MMC: 0, OMAP SD/MMC: 1
*** Warning - readenv() failed, using default environment

musb-hdrc: ConfigData=0xde (UTMI-8, dyn FIFOs, HB-ISO Rx, HB-ISO Tx, SoftConn)
musb-hdrc: MHDRC RTL version 2.0
musb-hdrc: setup fifo_mode 4
musb-hdrc: 28/31 max ep, 16384/16384 memory
USB Peripheral mode controller at 47401000 using PIO, IRQ 0
musb-hdrc: ConfigData=0xde (UTMI-8, dyn FIFOs, HB-ISO Rx, HB-ISO Tx, SoftConn)
musb-hdrc: MHDRC RTL version 2.0
musb-hdrc: setup fifo_mode 4
musb-hdrc: 28/31 max ep, 16384/16384 memory
```

Figure 4: Loading u-boot

- MLO is a first stage uBoot Bootloader designed to load a second stage uBoot bootloader with enhanced features. This second stage bootloader is also found on the FAT partition with the filename of "u-boot.img"

```

musb-hdrc: 28/31 max ep, 16384/16384 memory
USB Peripheral mode controller at 47401000 using PIO, IRQ 0
musb-hdrc: ConfigData=0xde (UTMI-8, dyn FIFOs, HB-ISO Rx, HB-ISO Tx, SoftConn)
musb-hdrc: MHDRC RTL version 2.0
musb-hdrc: setup fifo_mode 4
musb-hdrc: 28/31 max ep, 16384/16384 memory
USB Host mode controller at 47401800 using PIO, IRQ 0
Net: <ethaddr> not set. Validating first E-fuse MAC
cpsw, usb_ether
Hit any key to stop autoboot: 0
gpio: pin 53 (gpio 53) value is 1
mmc0 is current device
micro SD card found
mmc0 is current device
gpio: pin 54 (gpio 54) value is 1
SD/MMC found on device 0
reading uEnv.txt
370 bytes read in 3 ms (120.1 KiB/s)
Loaded environment from uEnv.txt
Importing environment from mmc ...
Running uenvcmd ...
8645240 bytes read in 1014 ms (8.1 MiB/s)
56827 bytes read in 25 ms (2.2 MiB/s)
## Booting kernel from Legacy Image at 82000000 ...
Image Name:   Linux-4.4.62
Image Type:   ARM Linux Kernel Image (uncompressed)
Data Size:    8645176 Bytes = 8.2 MiB
Load Address: 80008000
Entry Point:  80008000
Verifying Checksum ... OK
## Flattened Device Tree blob at 88000000
Booting using the fdt blob at 0x88000000
Loading Kernel Image ... OK
OK
Using Device Tree in place at 88000000, end 88010dfa

Starting kernel ...

[ 0.000000] Booting Linux on physical CPU 0x0
[ 0.000000] Initializing cgroup subsys cpuset
[ 0.000000] Initializing cgroup subsys cpu
[ 0.000000] Initializing cgroup subsys cpuacct
[ 0.000000] Linux version 4.4.62 (kiran@kiran-fastbitlab) (gcc version 6.3.1 20170109 (Linaro GCC 6.3-2017.02) ) #1 SMP Tue May 2 09:56:46 IST 2017
[ 0.000000] CPU: ARMv7 Processor [413fc082] revision 2 (ARMv7), cr=10c5387d
[ 0.000000] CPU: PIPT / VIPT nonaliasing data cache, VIPT aliasing instruction cache
[ 0.000000] Machine model: TI AM335x BeagleBone Black
[ 0.000000] cma: Reserved 48 MiB at 0x9c800000
[ 0.000000] Memory policy: Data cache writeback
[ 0.000000] CPU: All CPU(s) started in SVC mode.
[ 0.000000] AM335X ES2.0 (sgx neon )
[ 0.000000] PERCPU: Embedded 13 pages/cpu @df920000 s24268 r8192 d20788 u53248
[ 0.000000] Built 1 zonelists in Zone order, mobility grouping on. Total pages: 129408
[ 0.000000] Kernel command line: console=ttyO0,115200n8 root=/dev/mmcblk0p2 rw
[ 0.000000] PID hash table entries: 2048 (order: 1, 8192 bytes)

```

Figure 5: Loading from uEnv.txt

- uBoot will load using a default environment space. This default space includes a variable bootenv=uEnv.txt and associated script that allows additional variables to be added or overwritten by adding them to an uEnv.txt file placed on the FAT partition. uBoot will attempt to load this file and append the extra variables:
- uBoot will then load the Linux Kernel and compiled Device Tree Binary blob from eMMC:
- And boot with the ext4 root filesystem being loaded from /dev/mmcblk0p2

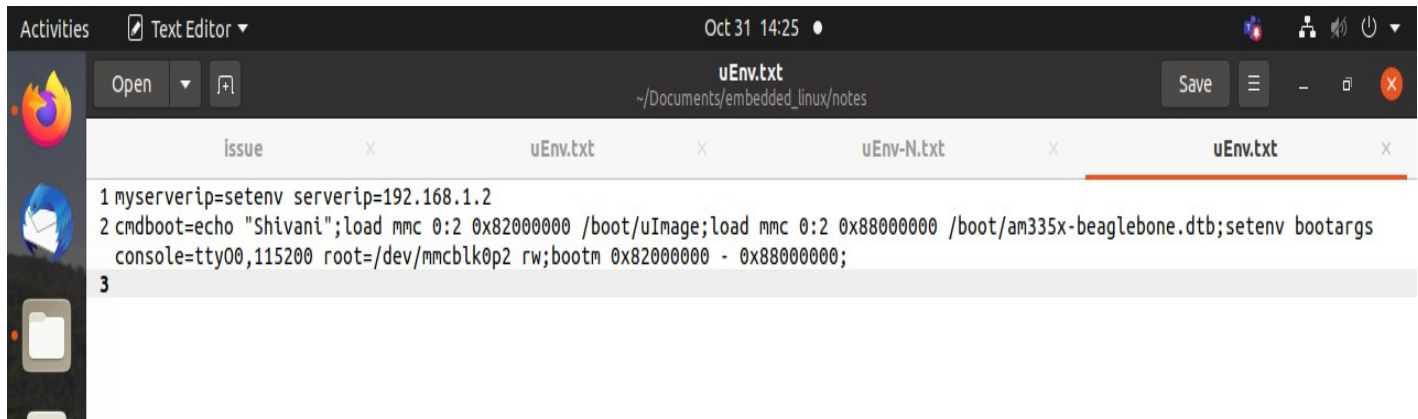
[illegible]

Figure 6: Booting to Kernel Level complete

In this step, the booting process is complete.

Activity 8-Challenge-Make uEnv.txt to Boot from MMC0 and MMC 1 .

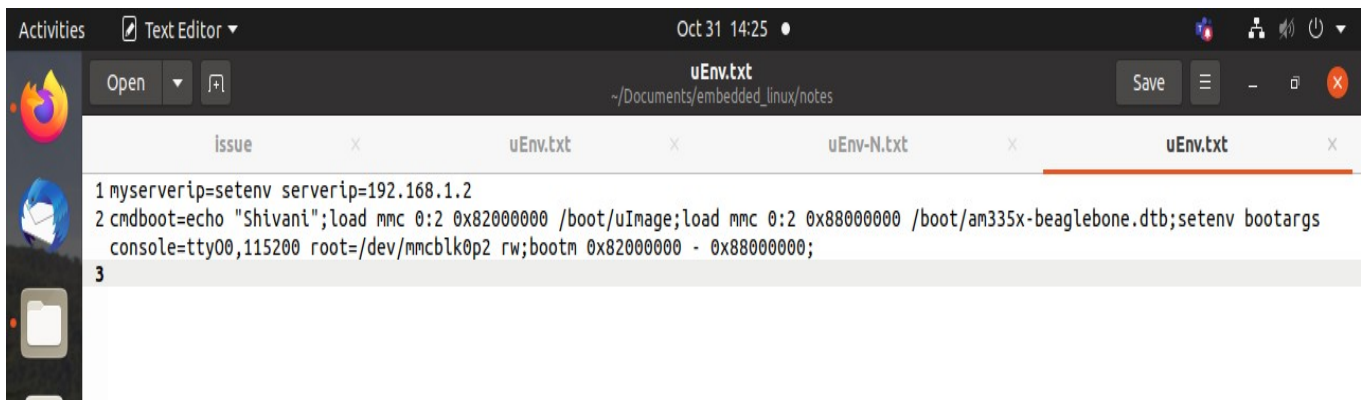
- Step 1:
Set IP address of the server with setenv and store inside a variable.
`myserverip=setenv serverip=192.168.1.2`
- Step 2:
Then set the console and baud rate and also set it the root to read and writable store inside a variable.
Set baud rate as 115200.
- Step 3:
The variable bootm should contain all the booting and loading contents to be taken from the host.
- Step 4:
Bootm should contain mmc value i.e whether eMMC or SD card, path and address of ulmage, path and address of .dtb file. It can also contain additional comments to be printed.
`cmdboot=echo "Shivani";load mmc 0:2 0x82000000 boot/ulmage; load mmc 0:2 0x88000000 boot/am335x-beaglebone.dtb;setenv bootargs console=ttyO0,115200 root=/dev/mmcblk0p2 rw;bootm 0x82000000 - 0x88000000;`



```
1 myserverip=setenv serverip=192.168.1.2
2 cmdboot=echo "Shivani";load mmc 0:2 0x82000000 /boot/uImage;load mmc 0:2 0x88000000 /boot/am335x-beaglebone.dtb;setenv bootargs
  console=ttyO0,115200 root=/dev/mmcblk0p2 rw;bootm 0x82000000 - 0x88000000;
3
```

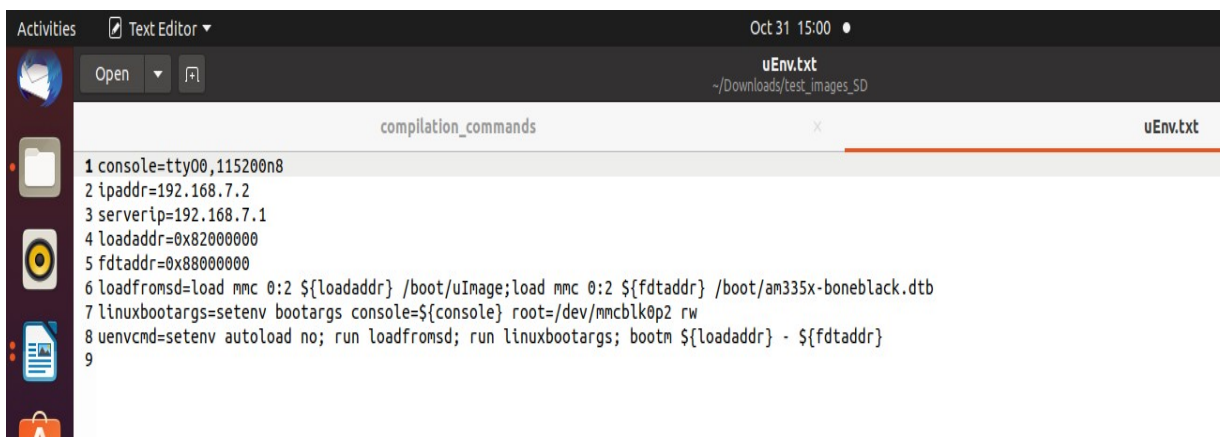
Activity 9-Challenge-Write a uEnv.txt file to automate TFTP boot.

Activity 10-Challenge-Write a generic uEnv.txt



A screenshot of a Linux desktop environment showing a text editor window titled "uEnv.txt" at the path ~/Documents/embedded_linux/notes. The window has tabs for "issue", "uEnv.txt", "uEnv-N.txt", and "uEnv.txt". The content of the file is as follows:

```
1 myserverip=setenv serverip=192.168.1.2
2 cmdboot=echo "Shivani";load mmc 0:2 0x82000000 /boot/uImage;load mmc 0:2 0x88000000 /boot/am335x-beaglebone.dtb;setenv bootargs
  console=tty00,115200 root=/dev/mmcblk0p2 rw;bootm 0x82000000 - 0x88000000;
3
```



A screenshot of a Linux desktop environment showing a text editor window titled "uEnv.txt" at the path ~/Downloads/test_images_SD. The window has tabs for "compilation_commands" and "uEnv.txt". The content of the file is as follows:

```
1 console=tty00,115200n8
2 ipaddr=192.168.7.2
3 serverip=192.168.7.1
4 loadaddr=0x82000000
5 fdtaddr=0x88000000
6 loadfromsd=load mmc 0:2 ${loadaddr} /boot/uImage;load mmc 0:2 ${fdtaddr} /boot/am335x-boneblack.dtb
7 linuxbootargs=setenv bootargs console=${console} root=/dev/mmcblk0p2 rw
8 uenvcmd=setenv autoload no; run loadfromsd; run linuxbootargs; bootm ${loadaddr} - ${fdtaddr}
9
```


Activity 11-Challenge-Increase the AUTOLOAD timings .

To configure autoloading timings, we have to create our own u-boot image.

- STEP 1: distclean : deletes all the previously compiled/generated object files.

make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- distclean

- STEP 2 : apply board default configuration for uboot

make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- am335x_boneblack_defconfig

- STEP 3 : run menuconfig, This is where you can change the autoloading timing.

```

Architecture select (ARM architecture) --->
ARM architecture --->
General setup --->
Boot images --->
API --->
Boot timing --->
Boot media --->
Environment ----
(10) delay in seconds before automatically booting
Console --->
() Default fdt file
[*] add U-Boot environment variable vers
[*] Display information about the CPU during start up
[*] Display information about the board during start up
Start-up hooks --->
SPL / TPL --->
Command line interface --->
Partition Types --->
Device Tree Control --->
*- Networking support --->
Device Drivers --->
File systems ----
Library routines --->
[ ] Unit tests ----

```

- Select the “delay in seconds before automatically booting” and press spacebar.
- Enter 10s as the delay and save it.

- STEP 4 : compile

make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- -j4 // -j4(4 core machine) will instruct the make tool to spawn 4 threads.

- After the compilation, start booting up BBB using serial booting method.
- Upload the newly created U-boot.img instead of the old one.


```
Completed on Dec 23 2019, 02:06:26.  
Port /dev/ttyUSB0, 11:50:31  
  
Press CTRL-A Z for help on special keys  
  
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC  
U-Boot SPL 2017.03-00270-g5cf618e (Mar 28 2017 - 17:14:21)  
Trying to boot from UART  
CCxyzModem - CRC mode, 2892(SOH)/0(STX)/0(CAN) packets, 8 retries  
Loaded 369972 bytes  
  
U-Boot 2017.05-rc2 (Oct 24 2020 - 11:33:09 +0530)  
  
CPU : AM335X-GP rev 2.0  
I2C: ready  
DRAM: 512 MiB  
MMC: OMAP SD/MMC: 0, OMAP SD/MMC: 1  
*** Warning - bad CRC, using default environment  
  
<ethaddr> not set. Validating first E-fuse MAC  
Net: cpsw, usb_ether  
Press SPACE to abort autoboot in 10 seconds  
=>
```

Activity 12-Challenge-Busybox "Dynamic" Compilation

Static Linking and Static Libraries is the result of the linker making copy of all used library functions to the executable file. Static Linking creates larger binary files, and need more space on disk and main memory. Examples of static libraries (libraries which are statically linked) are, *.a* files in Linux and *.lib* files in Windows. Static libraries occupy lot of space. They occupy space in kernel. Hence when memory is not a constraint we use static library.

Dynamic Linking doesn't require the code to be copied, it is done by just placing name of the library in the binary file. The actual linking happens when the program is run, when both the binary file and the library are in memory. Examples of Dynamic libraries (libraries which are linked at run-time) are, .so in Linux and .dll in Windows. Dynamic libraries occupy less space. They do not occupy space in kernel.

Steps for Busybox "Dynamic" Compilation

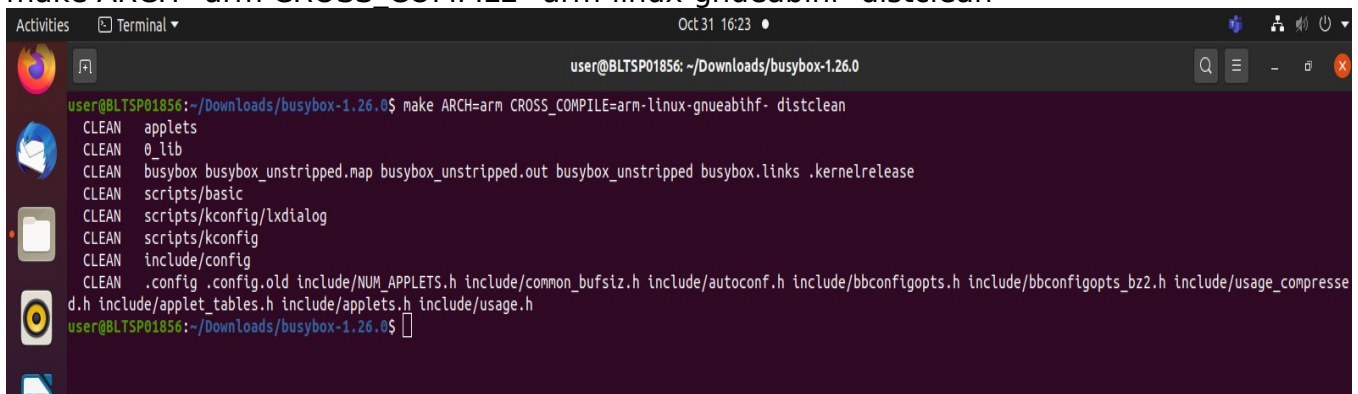
(open terminal)

STEP 1: Download busybox

<https://busybox.net/>

STEP 2 : Clean the earlier files

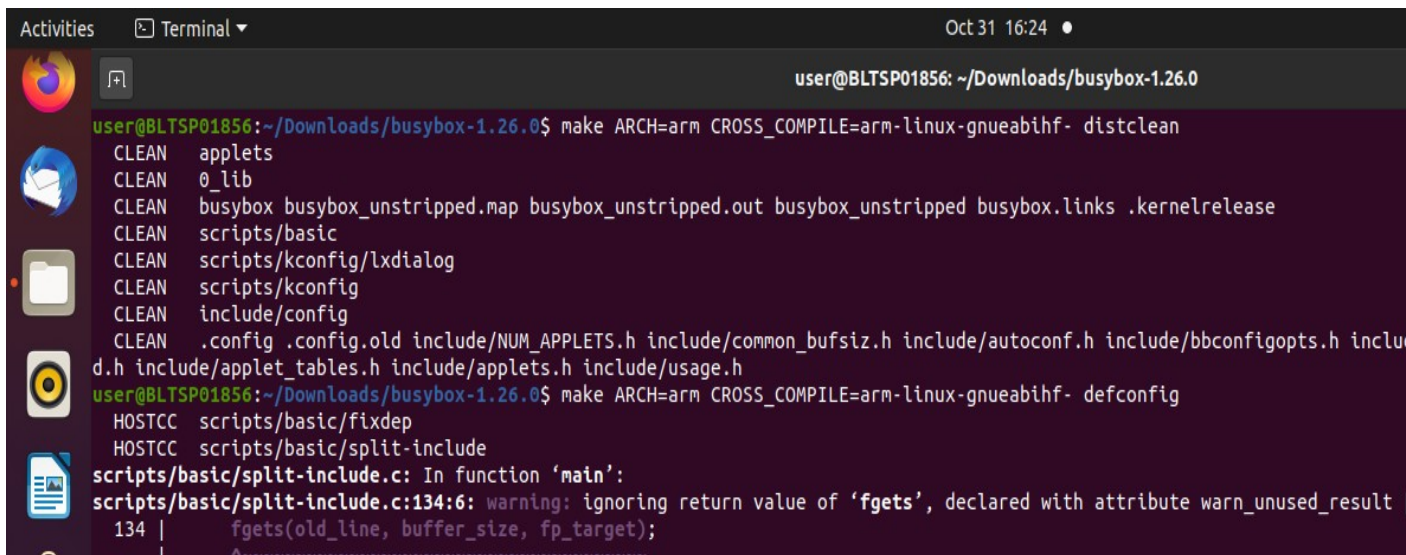
`make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi-hf- distclean`



```
user@BLTSP01856: ~/Downloads/busybox-1.26.0
user@BLTSP01856:~/Downloads/busybox-1.26.0$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi-hf- distclean
CLEAN    applets
CLEAN    _lib
CLEAN    busybox busybox_unstripped.map busybox_unstripped.out busybox_unstripped busybox.links .kernelrelease
CLEAN    scripts/basic
CLEAN    scripts/kconfig/lxdialog
CLEAN    scripts/kconfig
CLEAN    include/config
CLEAN    .config .config.old include/NUM_APPLETS.h include/common_bufsiz.h include/autoconf.h include/bbconfigopts.h include/bbconfigopts_bz2.h include/usage_compressed.h include/applet_tables.h include/applets.h include/usage.h
user@BLTSP01856:~/Downloads/busybox-1.26.0$
```

STEP 3 : Apply default configuration

`make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi-hf- defconfig`



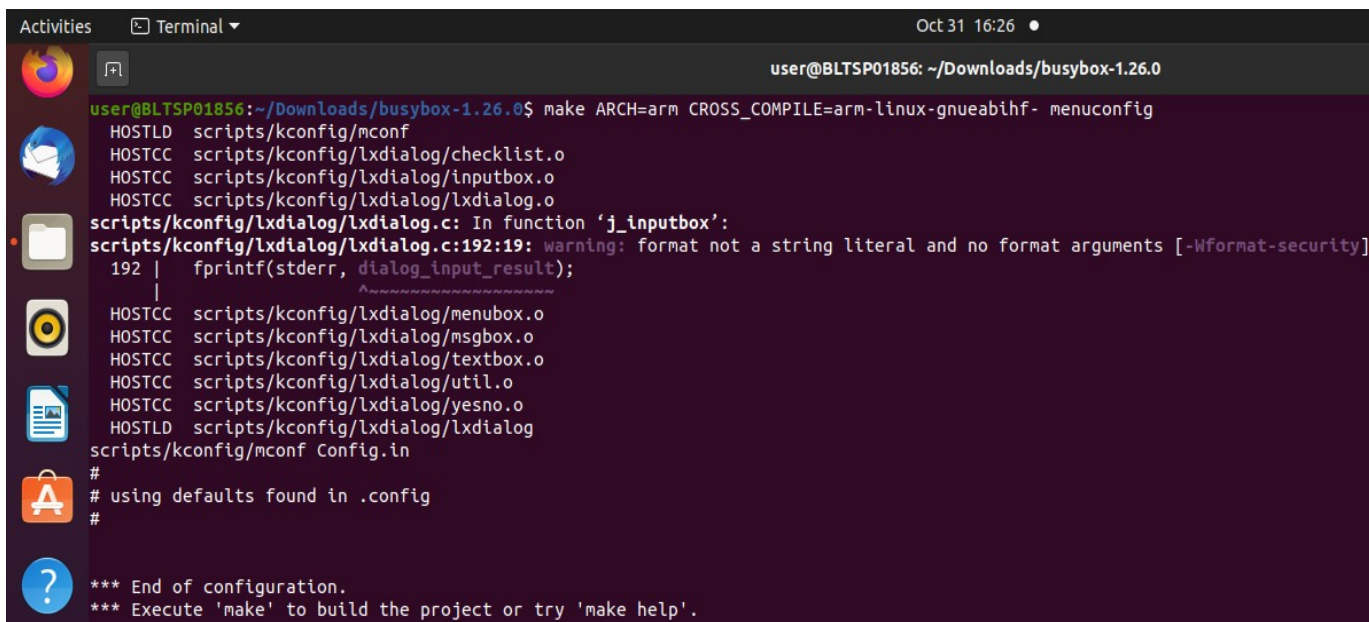
```

user@BLTSP01856: ~/Downloads/busybox-1.26.0
user@BLTSP01856:~/Downloads/busybox-1.26.0$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- distclean
CLEAN    applets
CLEAN    0_lib
CLEAN    busybox busybox_unstripped.map busybox_unstripped.out busybox_unstripped busybox.links .kernelrelease
CLEAN    scripts/basic
CLEAN    scripts/kconfig/lxdialog
CLEAN    scripts/kconfig
CLEAN    include/config
CLEAN    .config .config.old include/NUM_APPLETS.h include/common_bufsiz.h include/autoconf.h include/bbconfigopts.h inclu
d.h include/applet_tables.h include/applets.h include/usage.h
user@BLTSP01856:~/Downloads/busybox-1.26.0$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- defconfig
HOSTCC   scripts/basic/fixdep
HOSTCC   scripts/basic/split-include
scripts/basic/split-include.c: In function 'main':
scripts/basic/split-include.c:134:6: warning: ignoring return value of 'fgets', declared with attribute warn_unused_result
134 |     fgets(old_line, buffer_size, fp_target);
    |

```

STEP 4 : Change default settings.

make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- menuconfig



```

user@BLTSP01856: ~/Downloads/busybox-1.26.0
user@BLTSP01856:~/Downloads/busybox-1.26.0$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- menuconfig
HOSTLD   scripts/kconfig/mconf
HOSTCC   scripts/kconfig/lxdialog/checklist.o
HOSTCC   scripts/kconfig/lxdialog/inputbox.o
HOSTCC   scripts/kconfig/lxdialog/lxdialog.o
scripts/kconfig/lxdialog/lxdialog.c: In function 'j_inputbox':
scripts/kconfig/lxdialog/lxdialog.c:192:19: warning: format not a string literal and no format arguments [-Wformat-security]
192 |     fprintf(stderr, dialog_input_result);
    |
HOSTCC   scripts/kconfig/lxdialog/menubox.o
HOSTCC   scripts/kconfig/lxdialog/msgbox.o
HOSTCC   scripts/kconfig/lxdialog/textbox.o
HOSTCC   scripts/kconfig/lxdialog/util.o
HOSTCC   scripts/kconfig/lxdialog/yesno.o
HOSTLD   scripts/kconfig/lxdialog/lxdialog
scripts/kconfig/mconf Config.in
#
# using defaults found in .config
#
*** End of configuration.
*** Execute 'make' to build the project or try 'make help'.

```

STEP 5 :Menu Pops up

```

                                     Busybox Configuration
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modul
features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable

|| Busybox Settings --->
  Busybox Library Tuning --->
--- Applets
  Archival Utilities --->
  Coreutils --->
  Console Utilities --->
  Debian Utilities --->
  Editors --->
  Finding Utilities --->
  Init Utilities --->
  Login/Password Management Utilities --->
  Linux Ext2 FS Progs --->
  Linux Module Utilities --->
  Linux System Utilities --->
  Miscellaneous Utilities --->
  Networking Utilities --->
  Print Utilities --->
  Mail Utilities --->
  Process Utilities --->
  Runit Utilities --->
  Shells --->
  System Logging Utilities --->
---
  Load an Alternate Configuration File
  Save Configuration to an Alternate File
```

STEP 6 : Change default settings.
go in Busybox Setting->Build shared libbusybox
press space and save the setting.

```

                                Busybox Settings
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

[?] (-)
[*] Store applet usage messages in compressed form
[*] Include busybox applet
[*] Support --install [-s] to install applet links at runtime
[ ] Don't use /usr
[ ] Support for PAM (Pluggable Authentication Modules)
[*] Support for --long-options
[*] Use the devpts filesystem for Unix98 PTYs
[ ] Clean up all memory before exiting (usually not needed)
[*] Support utmp file
[*] Support wtmp file
[*] Support writing pidfiles
(/var/run) Path to directory for pidfile
[*] Support for SUID/SGID handling
[*] Runtime SUID/SGID configuration via /etc/busybox.conf
[*] Suppress warning message if /etc/busybox.conf is not readable
[ ] Support NSA Security Enhanced Linux
[ ] exec prefers applets
(/proc/self/exe) Path to BusyBox executable
--- Build Options
[ ] Build BusyBox as a static binary (no shared libs)
[ ] Build BusyBox as a position independent executable
[ ] Force NOMMU build
[?] Build shared libbusybox
[*] Produce a binary for each applet, linked against libbusybox
[*] Produce additional busybox binary linked against libbusybox
[*] Build with Large File Support (for accessing files > 2 GB)
() Cross Compiler prefix
() Path to sysroot
() Additional CFLAGS
() Additional LDFLAGS
() Additional LDLIBS
--- Installation Options ("make install" behavior)
[?] (+)

<Select>  < Exit >  < Help >

```

STEP 7 : Generate the busy box binary and minimal file system
 make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi-
 CONFIG_PREFIX=<install_path> install

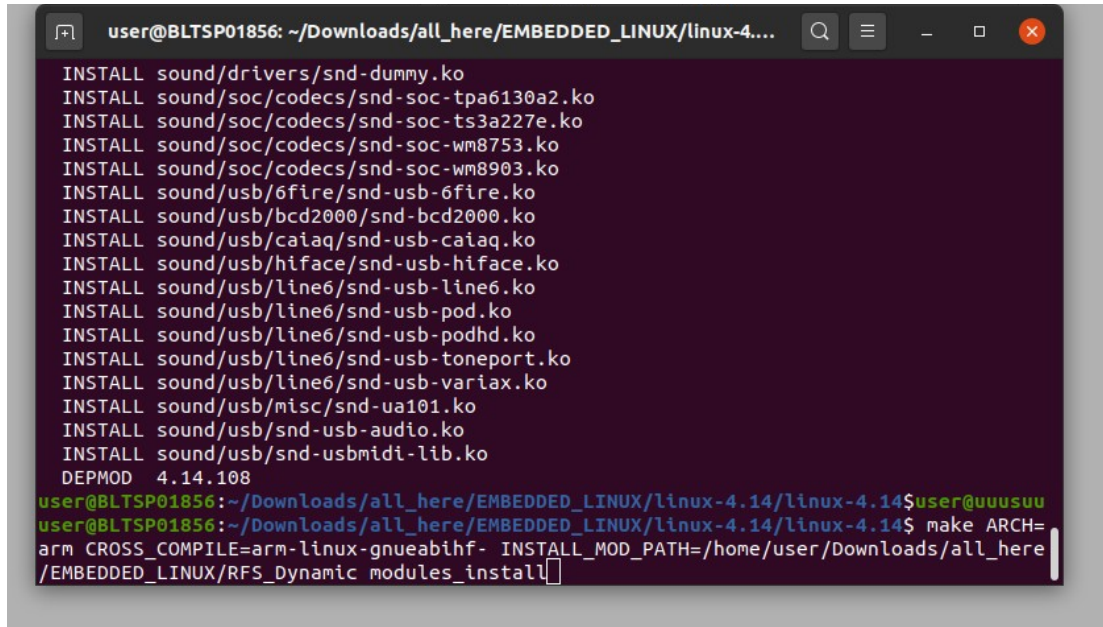
```

user@BLTSP01856: ~/Downloads/busybox-1.26.0
scripts/kconfig/mconf Config.in
#
# using defaults found in .config
#
*** End of configuration.
*** Execute 'make' to build the project or try 'make help'.
user@BLTSP01856:~/Downloads/busybox-1.26.0$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- INSTALL_MOD_PATH=/
home/user/Downloads/all here/EMBEDDED LINUX/RFS Dynamic modules_install
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- defconfig

```


STEP 7 : Open terminal in Linux-4.14

make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- INSTALL_MOD_PATH=<path of the RFS>



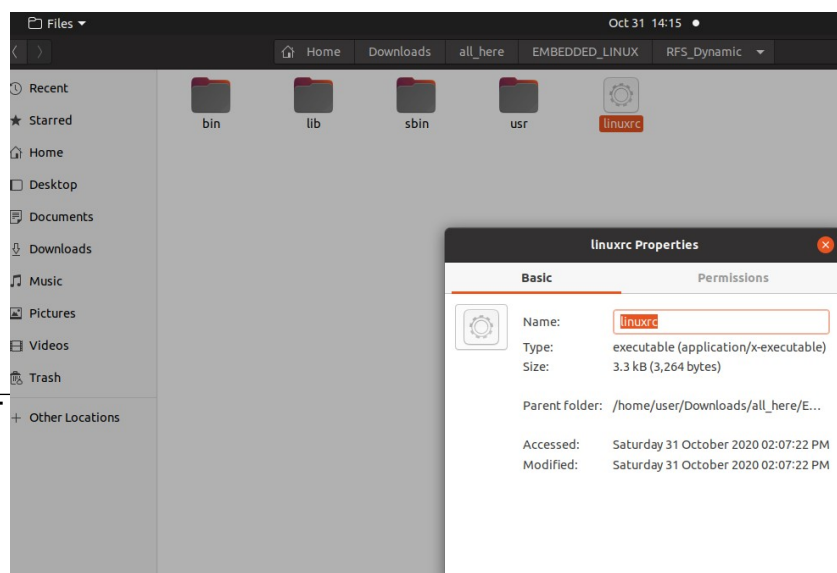
```

user@BLTSP01856: ~/Downloads/all_here/EMBEDDED_LINUX/linux-4....
INSTALL sound/drivers/snd-dummy.ko
INSTALL sound/soc/codecs/snd-soc-tpa6130a2.ko
INSTALL sound/soc/codecs/snd-soc-ts3a227e.ko
INSTALL sound/soc/codecs/snd-soc-wm8753.ko
INSTALL sound/soc/codecs/snd-soc-wm8903.ko
INSTALL sound/usb/6fire/snd-usb-6fire.ko
INSTALL sound/usb/bcd2000/snd-bcd2000.ko
INSTALL sound/usb/caiaq/snd-usb-caiaq.ko
INSTALL sound/usb/hiface/snd-usb-hiface.ko
INSTALL sound/usb/line6/snd-usb-line6.ko
INSTALL sound/usb/line6/snd-usb-pod.ko
INSTALL sound/usb/line6/snd-usb-podhd.ko
INSTALL sound/usb/line6/snd-usb-toneport.ko
INSTALL sound/usb/line6/snd-usb-variax.ko
INSTALL sound/usb/misc/snd-ua101.ko
INSTALL sound/usb/snd-usb-audio.ko
INSTALL sound/usb/snd-usbmidi-lib.ko
DEPMOD 4.14.108
user@BLTSP01856:~/Downloads/all_here/EMBEDDED_LINUX/linux-4.14/linux-4.14$ user@uuuuuu
user@BLTSP01856:~/Downloads/all_here/EMBEDDED_LINUX/linux-4.14/linux-4.14$ make ARCH=
arm CROSS_COMPILE=arm-linux-gnueabi- INSTALL_MOD_PATH=/home/user/Downloads/all_here
/EMBEDDED_LINUX/RFS_Dynamic modules_install

```

modules_install

STEP 8 :Dynamic files are generated.We can observe the drastic change in file size



Reference

- [1]Step by step configuration<https://www.youtube.com/watch?v=UMEUo6Wm6u4&list=PLGs0VKk2Di...>
- [2]Step by step configuration-<https://www.youtube.com/watch?v=c81tmb7WJxw&list=PLGs0VKk2Di...>
- [3]Beagleboard-<https://beagleboard.org/black>
- [4]Evolution of BBB-https://elinux.org/Beagleboard:BeagleBoneBlack#Revision_C_.28Production_Version.29
- [5]USB to TTL logic-<https://www.robotics.org.za/W7965>
- [6]TI PROCESSOR-<https://www.ti.com/lit/ug/swpu270t/swpu270t.pdf>
- [7]Different versions of BBB-<https://en.wikipedia.org/wiki/BeagleBoard>