Embedded Linux Learning Report

GLOBAL ENGINEERING ACADEMY

Genesis

L&T Technology Services

Details:

Name	PS Number	Email ID
Asha N	99002646	ashan.n@ltts.com

Table of Contents

Illustration Index

Day-1 Activity -1

1. Configuration of Boards

'Beagle Bone Black' BBB is a low-cost, community-supported development platform for developers and hobbyists. Boot Linux in under 10 seconds and get started on development in less than 5 minutes with just a single USB cable.

1.1 Linux

Linux is a family of open-source Unix-like operating systems based on the Linux kernel, an operating system

1.2 Steps for BBB configuration on Linux

1. Connect the hardware: In this step, we need to connect the Serial debug with the TTL(Transistor transistor logic) to USB converter and attached to the CPU
2. TTL to USB pinout: In this we are having four pins which are namely

Wire	Colour	Position
Gnd	Black	1st from power supply
TxD	Green	4th from power supply
RxD	white	5th from power supply
+5v	Red	Not req

3. Install: First, we need to install minicom by using the following command

   sudo apt-get install minicom

4. Set up setting: BBB name and the TTL connector is being setup. ***sudo minicom -s***
5. Check for Menu Options: Check for name and Version and hardware and software configuration from the the menu.

   Hardware workflow - NO

   Software workflow – NO

6. Check for message log: ***dmesg.***
7. Login to minicom: This can be done using ‘root’ as a password.
8. Folder creation: Beagle Bone Black folder will be created on the desktop.

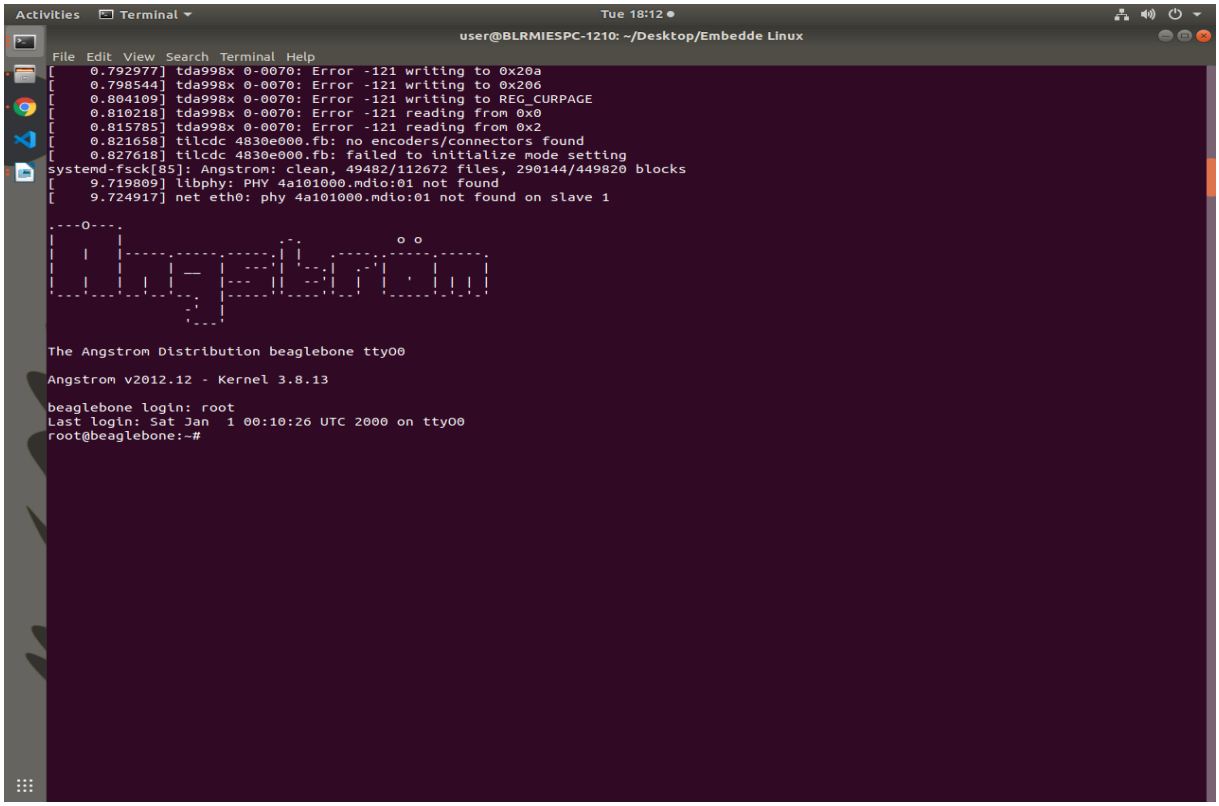


Illustration 1: BBB login page.

1.3 Windows

- Ethernet cable have to the board to join a local network.
- Power ON the the board.
- Install drivers for the OS and reboot the system.
- Search the IP address 192.168.7.2. this redirects to the local webpage Beagle Bone Black Board.
- Click on the Cloud9 IDE. This opens an Integrated Development Environment where the user codes can be written and executed.
- Open a new terminal and type ifconifg. It shows a ip address of the board.
- Install putty or teraterm software. Now, open putty and type the ip address.
- Install the Tight VNC Viewer in the Windows OS and Tight VNC Server in the BBB board. Initialize the VNC server and establish a connection between the both.
- Latest software image should be downloaded.
- Install compression utility to the windows.
- Install SD card programming utility.
- Copy the correct image to your SD card.
- unmount the SD card.
- Boot your board through the SD card.

Day-1 Activity -2

1. Differences between Raspberry pie , Dragon, imx7 Sabre, BBB

PARAMETER	RASPBERRY PI	IMX7 SABRE	BEAGLEBONE BLACK
Model	It uses Model B version	Freescale i.MX6 Quad	It uses Rev A5 version.
Processor TypeI.	t uses ARM11 processor.	Two Arm® Cortex®-A7 Single Arm Cortex -M4	It uses ARM Cortex-A8 processor.
RAM	For the functioning of raspberry pi, 512 MB SDRAM is used	1 GB DDR3, 533 MHz 2)eMMC expansion footprint 3)NAND flash expansion footprint 4)QSPI flash expansion footprint	For the functioning of beaglebone black, 512 MB DDR3L is used.
Processor Speed	It uses 700 MHz for processing.	Two Arm® Cortex®-A7 core operating up to 1 GHz Single Arm Cortex -M4 core operating up to 200 MHz	It uses 1 GHz for its processing.
GPIO Pins	It has 12 GPIO pins.		It has 69 GPIO pins.
USB Master	It has 2 USB 2.0 on board.	USB 3.0 Type-A Male, USB micro-B Male	It has 1 USB 2.0 on its board.
UART	It uses 1 UART to transmit and receive serial data.	USB port	It uses 5 UART to transmit and receive serial data.
No. of I/O pins	It has 8 Digital, 0 Analog pins.		It has 65 Digital, 7 Analog pins.

DAY-1 Activity-3

1. Versions of BBB and the evolution of the Beagle bone.

1.1 Versions of BBB

Parameters	PocketBeagle	BeagleBoard-X15	BeagleBone Black	BeagleBone	
Release Date:	September 21, 2017	23 September 2016	April 23, 2013	October 31, 2011	September 14
SoC	OSD3358-SM	Sitara AM5728	AM3358/9		DM3730
CPU	Sitara AM3358 ARM Cortex-A8	Dual ARM Cortex-A15 + Dual ARM M4 (212 MHz) + Quad PRU (200 MHz)	Cortex-A8 + Dual PRU (200 MHz)		
GPU	PowerVR SGX530	Dual PowerVR SGX544	PowerVR SGX530(200 MHz)		
USB ports:	1 x Micro USB Type B	3 x USB 3.0 Type A Host 4 x USB 2.0 Host 1 x Micro USB Type B	1 x Standard A host port (direct). 1x mini B device port (direct)	1 x Standard A host port . 1x mini B device port	4 x Standard hub with Ethernet 1x mini AB
Memory (SDRAM)	512 MiB DDR3	2048 MiB DDR3L	512 MiB DDR3	256 MiB DDR2	512 MiB DD
Power	150 mA @ 5 V	210–460 mA @5 V			

1.2 Evolution of the Beagle bone.

Revision C

➢ This revision increases the eMMC from 2GB to 4GB.

1) Complaints from the community about lack of space left in the eMMC.

2) For those worried about their eMMC wearing out, the added space will help in the area of

moving the data around to prevent wear out. Assuming of course you don't try and use it all.

3) Concerns over the long-term availability of the 2GB device. 4GB is currently the low end of the offering. This also gives us two sources.

Revision B

➢ This version moves to the AM3358BZCZ100 processor as we are no longer able to get the limited production version of the AM3359AZCZ100.

➢ No changes in features or operation of the board resulted from this change.

Revision A6A

➢ Added optional ohm resistor zero to tie GND_OSC1 to system ground.

➢ Changed C106 to a 1uF capacitor.

➢ Changed C24 to a 2.2uF capacitor.

Revision A6

➢ Based on notification from TI, in random instances there could be a glitch in the SYS_RESETn signal from the processor where the SYS_RESETn signal was taken high for

a momentary amount of time before it was supposed to. Noise issues were observed in other

designs where the clock oscillator was getting hit due to a suspected issue in ground bounce.

A zero ohm resistor was added to connect the OSC_GND to the system ground.

Revision A5C

Production had some fallout of boards when running the HDMI tests in the previous production run. Resistor values were tweaked to improve the test results.

1) Changed R46,R47,R48 to a 0 ohm.

2) Changed R45 to a 22 Ohm.

Revision A5B

➢ 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.

➢ Made the LEDs dimmer for those that could not sleep due to the brightness of the LEDs.

Revision A5A

➢ Boards are built using the XAM3359AZCZ100 processor.

➢ PCB Change...LCD noise issue was resolved by adding 47pf bypass caps on some of the LCD signals.

Revision A4B

Added a 100K pull down resistor between pins 1 and 4 of J1 to fix the serial port issue.

Revision A4A

Incorporated the capacitors to fix the noise issue on the display

Revision A4

First prototype release version of the board. Limited distribution. One notable issue here is that the board has an AM3352 processor instead of an AM3359, despite how the part is marked. Part was mismarked as an AM3359. The SGX and PRU are not operational.

DAY-1 Activity-4

1. Pin expansion header of BBB

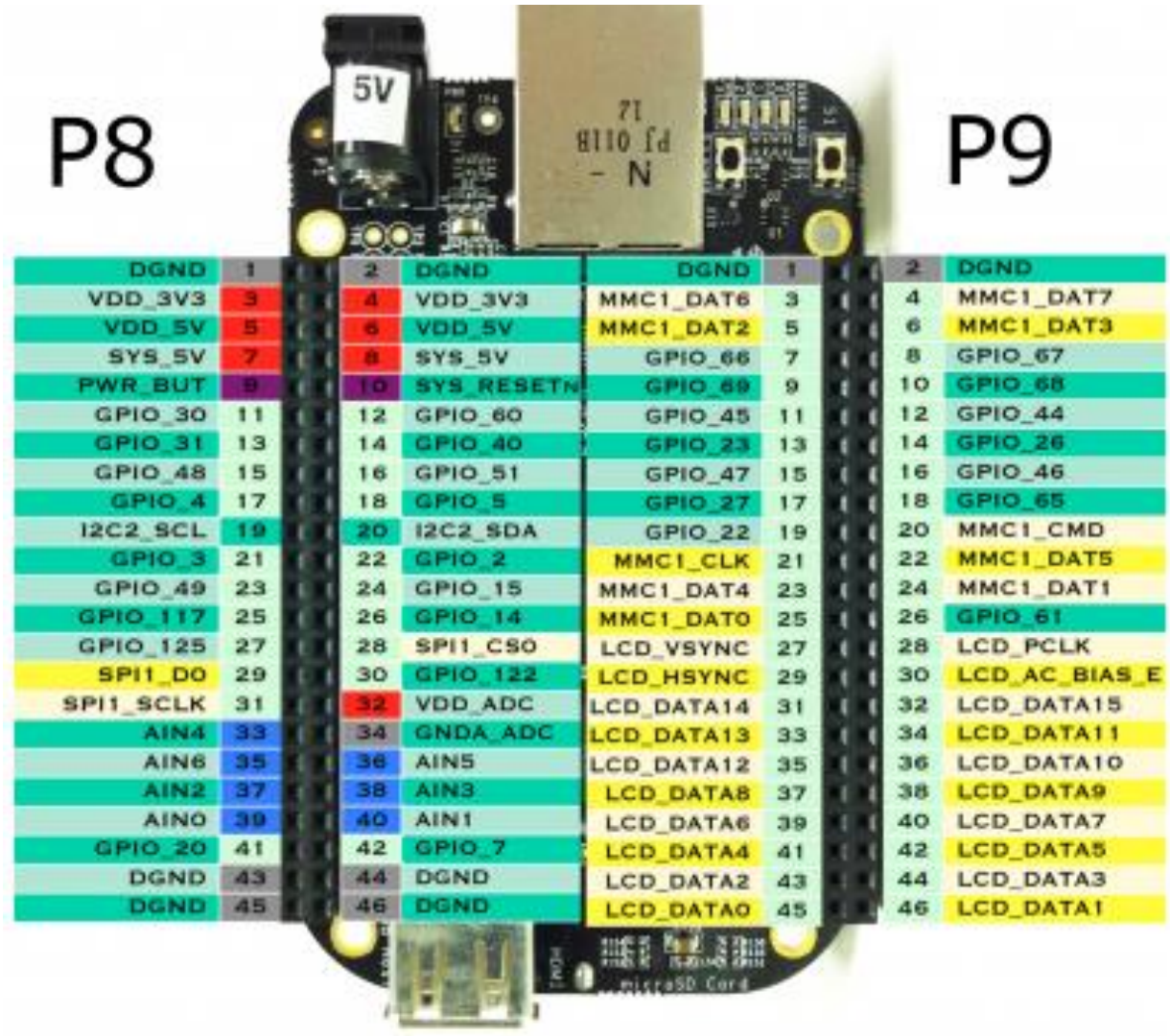


Illustration 2: BBB expansion pin and configuration

DAY-2 Activity-1

1. Testing MLO image on BBB and Testing U-boot image on BBB

1.1 MLO image on BBB

Before starting the process we need to update the Linux system and upgrade the different packages already available these things can be done using the command.

- ***sudo apt-get update***
- ***sudo apt-get upgrade***

After the updating the system we can create a disk part-ion in our SD card using Gparted in the linux system. GParted is a graphical tool for managing disks on Linux. It is very powerful. we can do almost any type of partitioning and disk management with GParted. GParted has a simple user interface and it is very easy to use. For installing Gparted on Linux system we can use following command.

- ***sudo apt-get install gparted***

After installation we need to check for the SD card name to do the further part-ion process on it. We can get it by using the command.

- ***dmesg***

At the end of the terminal it will show the number of removal disk available from which we need to check for our SD card and mark the exact name on the Gparted graphical interface.

Process for creating part-ion on the Gparted graphical interface and pasting MLO.

- Open gparted select the SD card name from the extreme right corner
- After that delete all the files previously there.
- Select the option of UN-allocated space under FILE SYSTEM
- Search for file system and select fat16, for getting a space of of 4gb.
- The 1st half will be label as ‘BOOT’.
- Search the file system again and this time select ex3 for getting remaining space.
- The 2nd half will be label as ‘ROOTFS’.
- Now two part-ion is being created named as BOOT and ROOTFS
- Once check all the allocation and confirm the changes occurred.
- After that flag is being set for visibility in the file system.
- For BOOT we select and mange the flag value with boot.
- Once the boot flag is being set we will close the gpart GUI.
- After that from the file access we will paste the MLO file in the BOOT section.

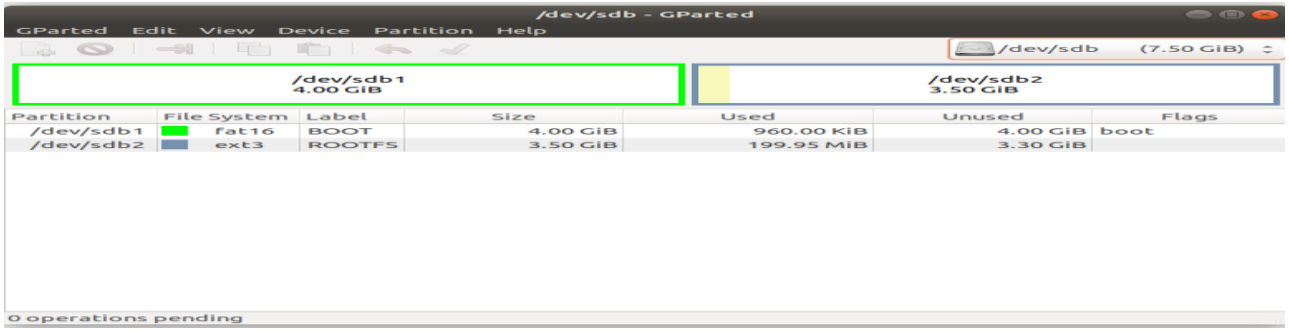


Illustration 3: Gparted partion on linux sytem for bootable pendrive

Now we need to configure the BBB for initiating the booting process

- Start minicom using the command ***sudo minicom***
- Now connecct the TTL to USB converter to serial debug.

- Once the above step is done the terminal will start for the minicom.
- Insert SD card to the SD card slot of the BBB.
- Now insert the mini power cable to BBB.
- After that press S2 button that is the boot button on BBB.
- After which the booting process will start and we will get the following output.

Message and output we will get on the command line:

U-Boot SPL 2017.05-rc2 (May 02 2017 - 08:53:40)

Trying to boot from MMC1

reading u-boot.img

spl_load_image_fat: error reading image u-boot.img, err - -1

Failed to mount ext2 filesystem...

spl_load_image_ext: ext4fs mount err - 0

Loading error we will get because while booting it will search for the U-boot image but in the BOOT file system it is not present so it will fail to initiate the booting.

```
ltts@ltts-OptiPlex-7060:~$ sudo minicom
[sudo] password for ltts:


Welcome to minicom 2.7.1

OPTIONS: I18n
Compiled on Aug 13 2017, 15:25:34.
Port /dev/ttyUSB0, 14:56:52

Press CTRL-A Z for help on special keys


U-Boot SPL 2013.04-dirty (Jun 19 2013 - 09:57:14)
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
reading u-boot.img
spl: error reading image u-boot.img, err - -1
### ERROR ### Please RESET the board ###
```

1.2 U-boot image on BBB

Here we need to keep the uboot image file in the BOOT section along with MLO.

Process to follow:

- Start the minicom
- Check for the BBB connection and configuration
- Enable the power source using the mini power cable.
- We will get the UEnv.txt missing as it is necessary.
- The further booting process it will ask for uEnv file

```
Port /dev/ttyUSB0, 14:56:52

Press CTRL-A Z for help on special keys


U-Boot SPL 2013.04-dirty (Jun 19 2013 - 09:57:14)
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
reading u-boot.img
reading u-boot.img


U-Boot 2013.04-dirty (Jun 19 2013 - 09:57:14)

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
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
** Unable to read file uEnv.txt **
gpio: pin 55 (gpio 55) value is 1
** File not found /boot/uImage **
U-Boot#
```

DAY-2 Activity-2

1. Linux boot sequence.

First Stage:

- The Soc is powered and the logic execution at rest vector.
- The control given to the Rom boot loader
- Rom boot loader decides boot device order based on hardware boot setting.
- Rom boot loader loads the first stage boot loader from boot devices into the internal SOC memory and passes control to it.
- The first stage boot loader copies the second stage boot loader into RAM and passes control to it.

Second Stage

- The second stage Boot Loader loads its configuration setting either statistically embedded or an external file.
- After that it find and load the Linux Kernel and device tree binary into RAM. {DDR3 RAM for BBB}
- After which setting up the kernel boot arguments takes place.
- It passes control to the kernel which uses the boot arguments and device tree address to initiate itself and hardware device.

Third stage

- Using boot arguments the kernel locates all the mounts and the root file system.
- Kernel runs the init process to start the user space.
- At this stage the user threads and the main thread start their execution.

- After this we reached to the user application interface where the user can directly access the application in the system.

Challenge 1

Make uEnv.txt to Boot from MMC0 and MMC 1. In this case, we loaded the boot images from MMC1 interface(eMMC) or MMC0 and used the file system present on the MMC0(MiroSD card) or MMC1, Challenge for you is to change this uEnv.txt such that boot images are loaded from MMC0(Micro SD) and MMC1 (eMMC).

----Solution----

In this problem statement we need to make the change in uEnv.txt such that the boot images are loaded from MMC0 and MMC1. It can be done using the help of any of the 3 protocols they are.

-1 loadx

-2 loady

-3 loadz

With the help these protocol we will select the boot loaders files that are {MLO, uboot image} and the kernel files that is {Uimage} from the system and will directly place thoes into the BBB.

For the above process we need tp follow these commands

myserverip=setenv serverip=192.168.1.2

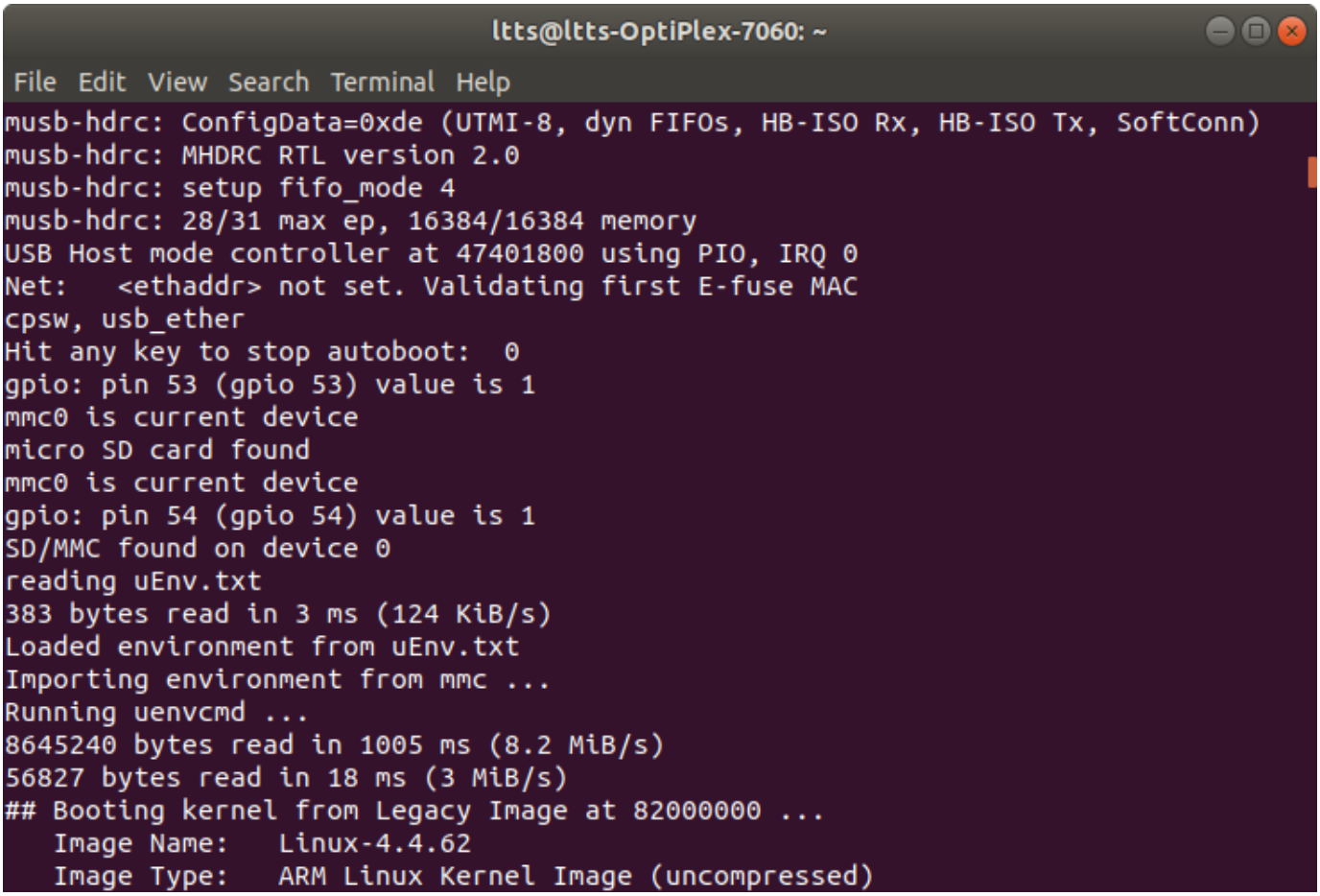
bootcmd=echo "TEAM-1--Nalanda";load mmc 0:2 0x82000000 /boot/uImage;load mmc 0:2 0x88000000 /boot/am335x-boneblack.dtb;setenv bootargs console =ttyO0,115200 root=/dev/mmcblk0p2 rw;bootm 0x82000000 - 0x88000000;

Here we are printing the message "TEAM-1 -- Nalanda" into the terminal which will be reflect before the booting compilation will take place.Also setting of the server ip using the environmental value serverip will give us a a new set of server ip.

load mmc 0:2 that is {MMC 0 is the SD card and the 0:2 is the second partion where we are having the booting file}

addressspace of 0x82000000 conatians the uImage.

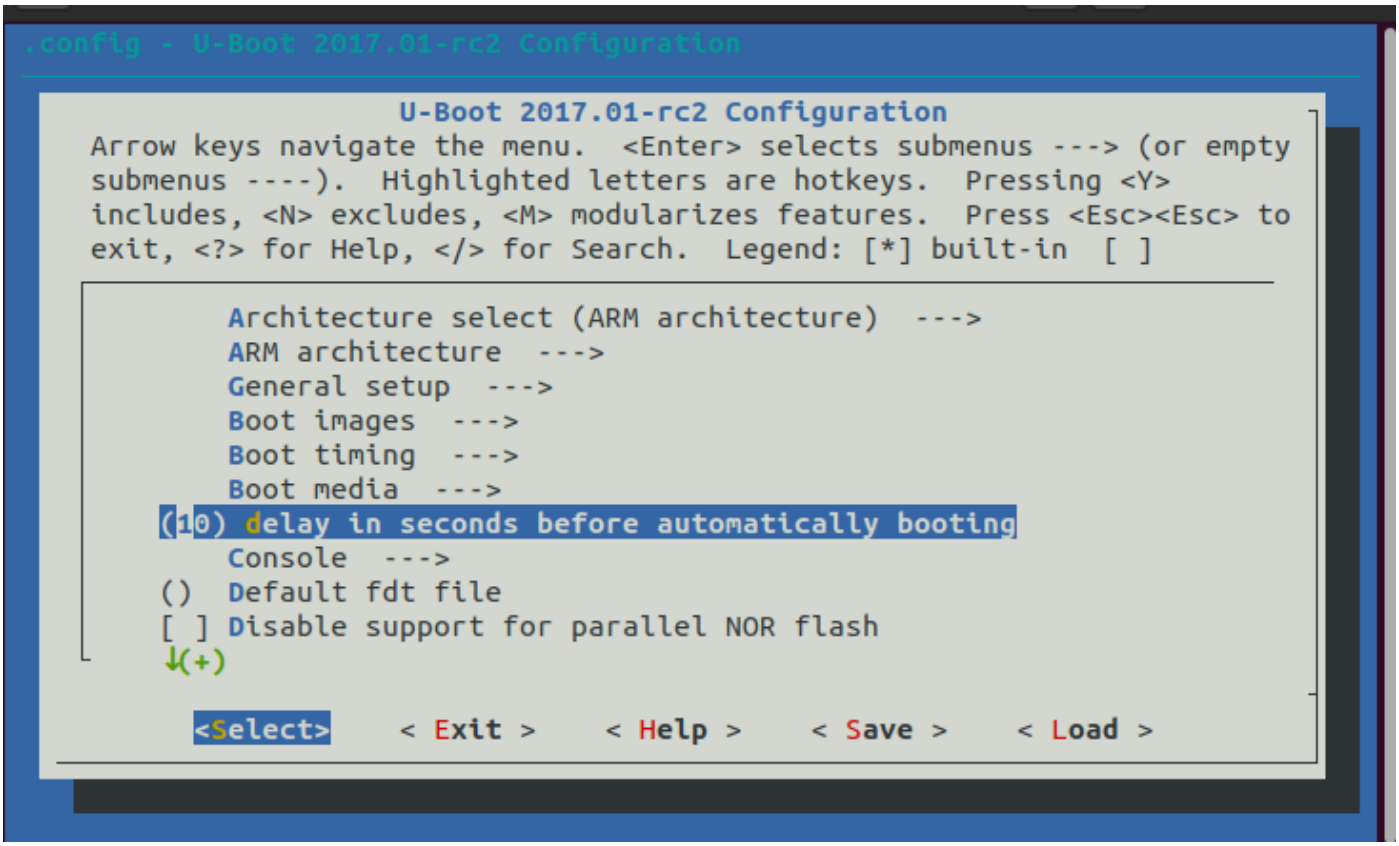
addressspace of 0x88000000 contains the arm335x-boneblack.dtb



Challenges 2

Increase the AUTOLOAD timings . you might have observed that , when the uboot boots it just
waits for 5 seconds before going to auto loading mode(reading uEnv.txt , loading uImage, etc).
Within 5 seconds if you press "space" bar then you will get the uboot command prompt.
Challenge for you is to increase that autoload timing to 10 seconds and confirm this change by
testing on the hardware. you have to change the uboot source code and then rebuild it to generate the uboot image.

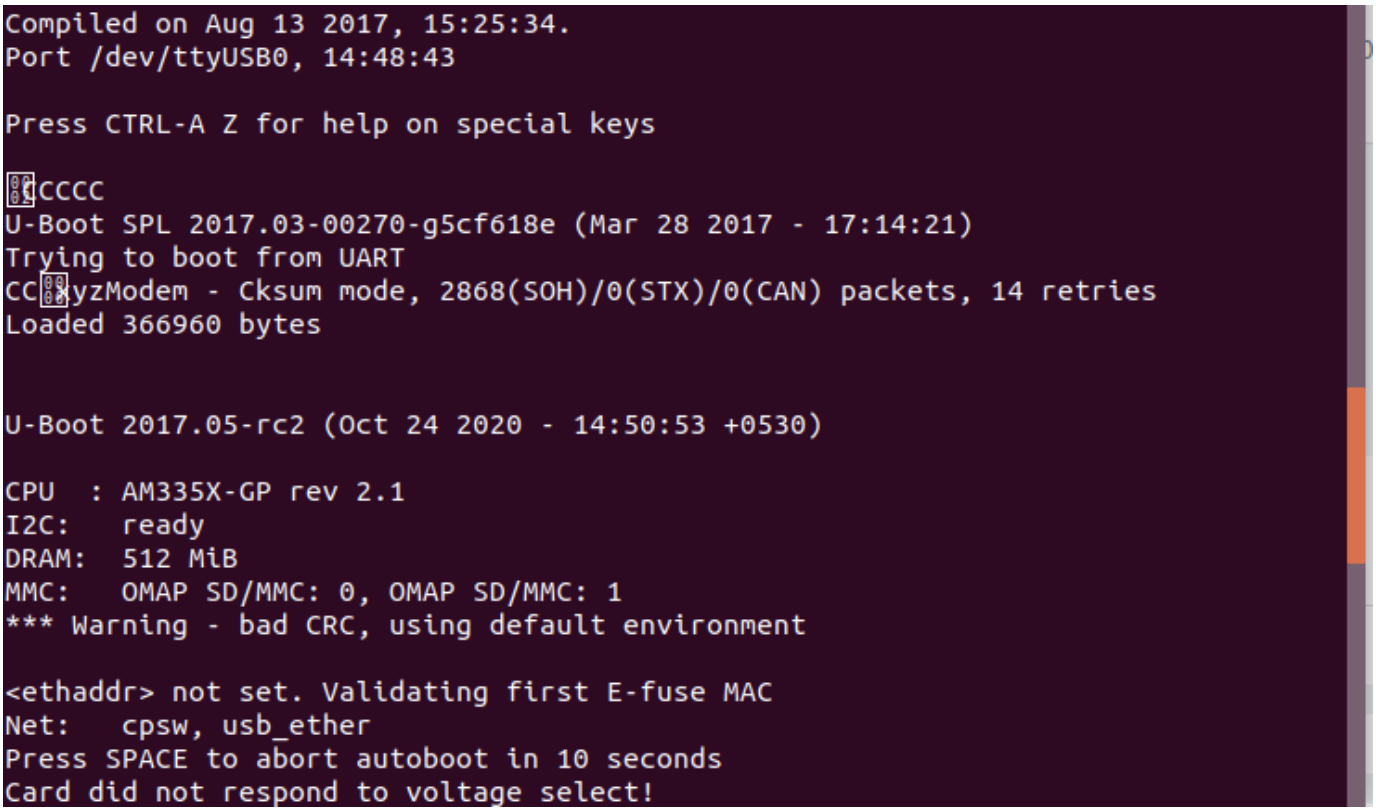
- Run the command make terminal.
  ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- menuconfig\
- It opens the below window and change the delay setting.



- Check how many core in the host PC
- Run the command

make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- -j< No. Of cores>

Boot



Challenges 3

Busybox "Dynamic" Compilation In the entire "Busybox" lectures we have used "Static" binaries. That means all the generated utilities/commands of busybox are "statically linked" binaries. if you want to test any applications which are cross compiled by "Dynamic linking "
then those applications wont execute on your Busybox file system. The challenge for you is to
reconfigure and recompile the busybox to generate "dynamically linked " binaries/utilities and
you should also able to test any applications which are dynamically linked.

1. Install all the packages such as:
Build-essential, Flex, Busy Box
For u-boot compilation
2. Open the terminal and navigate to the directory U-boot2017.01-rc2
3. Run the command:

➢ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- distclean

➢ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- am335x_boneblack_defconfig

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

➢ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- -j8

4. Download kernel from https://github.com/beagleboard/linux Go to that directory.
5. Run the commands

➢ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- distclean

➢ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- bb.org_defconfig

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

➢ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- uImage dtbs LOADADDR=0x80008000 -j8

➢ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- -j8 modules

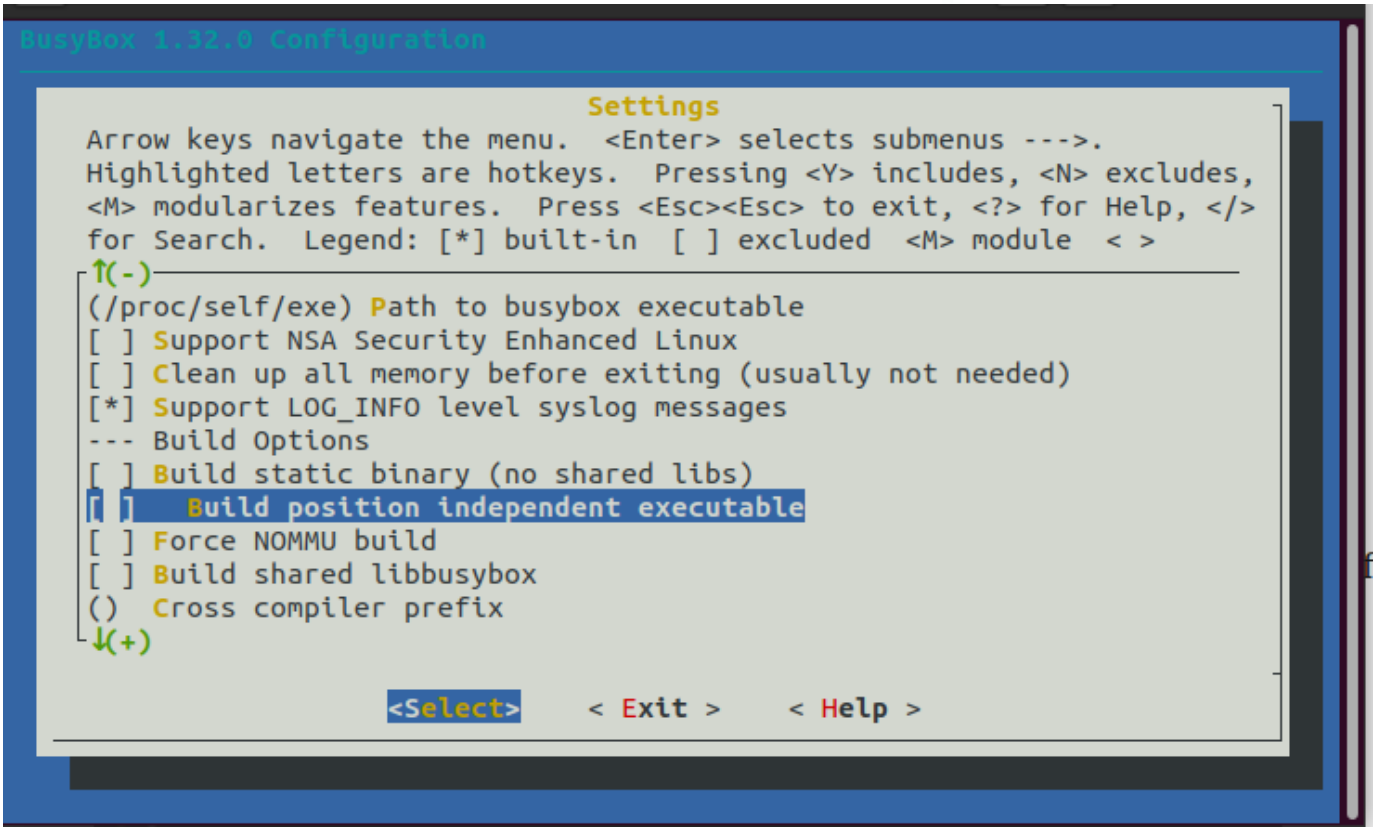
➢ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- INSTALL_MOD_PATH=<destination_path> modules_install

6. Open the busybox directory
7. Run the commands:

➢ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- defconfig

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

In the settings, select dynamic binary by deselecting ‘*’ of the build static binary



- make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- CONFIG_PREFIX=<destination_path > Install
- make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- CONFIG_PREFIX=< destination _path> install

Challenge 4

Increase the AUTOLOAD timings . You might have observed that , when the uboot boots it just waits for 5 seconds before going to auto loading mode(reading uEnv.txt , loading uImage, etc). Within 5 seconds if you press "space" bar then you will get the uboot command prompt. Challenge for you is to increase that autoload timing to 10 seconds and confirm this change by testing on the hardware. you have to change the uboot source code and then rebuild it to generate the uboot image.

----Solution----

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

- distclean : deletes all the previously compiled/generated object files. ***make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- distclean***
- apply board default configuration for uboot ***make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- am335x_boneblack_defconfig***
- Run menuconfig, This is where we can change the auto load timing.
- After that select the “delay in seconds before aromatically booting” and press space bar. Enter 10s as the delay and save it.
- Compile make ***ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- -j4***
- Here -j4 meaning 4 core machine it 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.

Challenge 5

Busybox "Dynamic" Compilation. In the entire "Busybox" lectures we have used "Static" binaries. That means all the generated utilities/commands of busybox are "statically linked" binaries. if you want to test any applications which are cross compiled by "Dynamic linking " then those applications wont execute on your Busybox file system.
The challenge for you is to reconfigure and re-compile the busybox to generate "dynamically linked " binaries/utilities and you should also able to test any applications which are dynamically linked.

----Solution----

- download busybox https://busybox.net/
- Apply default configuration ***make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- defconfig***
- change default settings if you want ***make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- menuconfig***
- generate the busy box binary and minimal file system ***make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- CONFIG_PREFIX=<install_path> install***

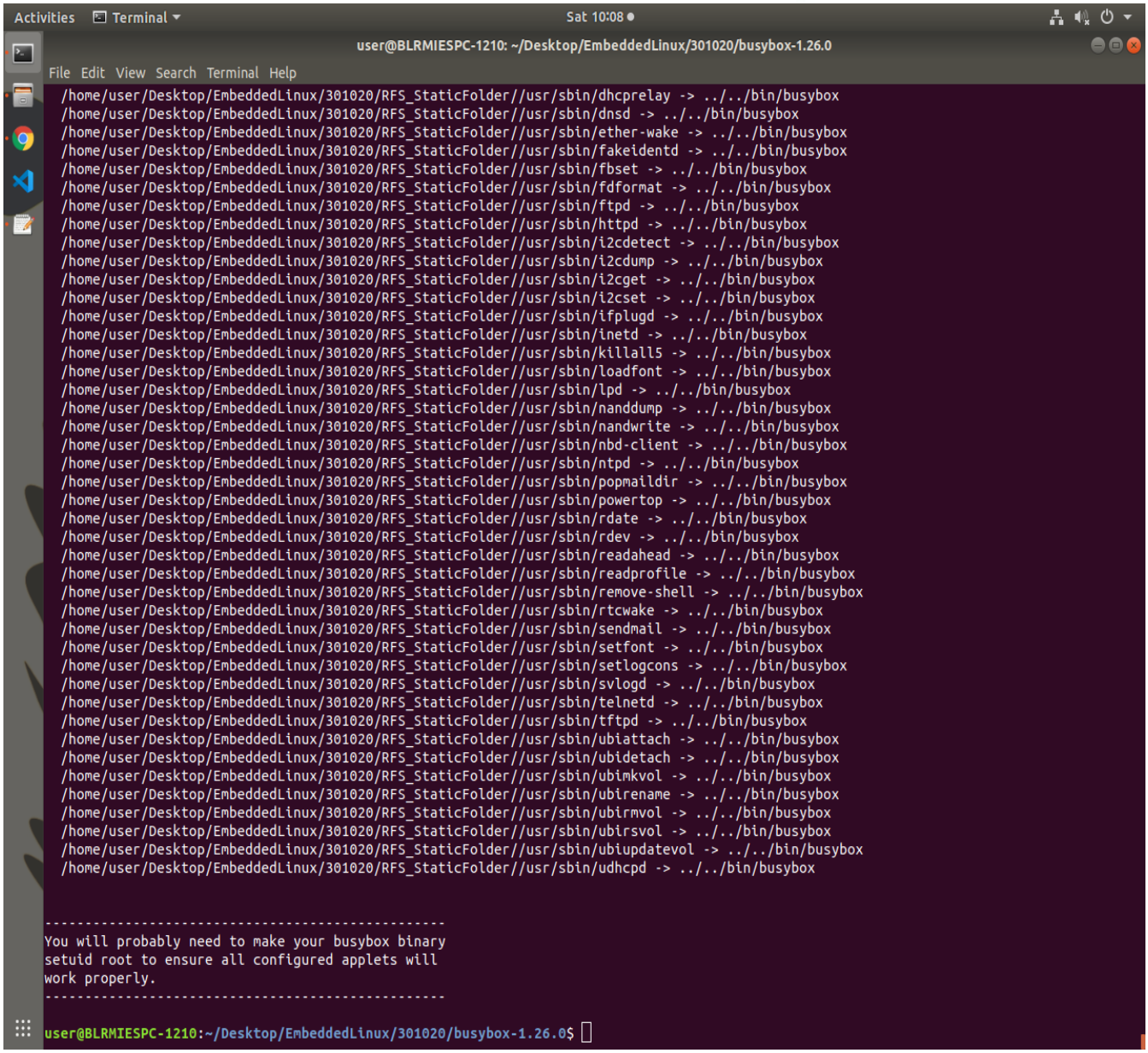


Illustration 4: Busy box compilation image

References

- https://en.wikipedia.org/wiki/BeagleBoard
- https://beagleboard.org/boards
- https://www.nxp.com/design/development-boards/i-mx-evaluation-and-development-boards/sabre-board-for-smart-devices-based-on-the-i-mx-7dual-applications-processors:MCIMX7SABRE
- https://www.educba.com/raspberry-pi-3-vs-beaglebone-black/
- https://components101.com/microcontrollers/beaglebone-black-pinout-datasheet
- "*USB-powered Beagle Board from Digi-Key Unleashes Community Development with Laptop-like Performance and Expansion for $149*" (Press release). Digi-Key. July 28, 2008. Archived from the original on October 4, 2011. Retrieved September 15, 2017.
- "*Digi-Key Announces New Open Source BeagleBoard Development Board*" (Press release). Digi-Key. May 13, 2009. Archived from the original on October 4, 2011. Retrieved September 15, 2017.
- "*BeagleBoard-xM*". BeagleBoard.org. Kridner, Jason (May 4, 2017). Texas Instruments. Retrieved September 15, 2017.
- "*Meet BeagleBone, the new $89 open source hardware platform, giving electronic enthusiasts a smaller, friendlier and more affordable treat*" (Press release). BeagleBoard.org. PR Newswire. October 31, 2011. Retrieved September 15, 2017.
- "*Digi-Key Continues Support of Innovative Line of TI-based ARM Development Boards from BeagleBoardorg*" (Press release). Digi-Key. April 23, 2013. Retrieved September 15, 2017.
- "*BeagleBoard:BeagleBoard-X15*". Coley, Gerald (February 24, 2017). eLinux. Retrieved September 15, 2017