**ESE-3005 Embedded Systems Architecture II**

**LAB 6**

**GROUP No. 2**

|  |  |
| --- | --- |
| **Group Members** | **Student ID** |
| Christy Rachel Philip | C0765535 |
| Fahad Rahman | C0769871 |
| James M Chacko | C0777192 |
| Premil Presannan | C0777191 |
| Vy Nguyen | C0776242 |

**Introduction**

In this lab we are creating yocto for Beaglebone and Qemulator

**Step 1**

Change the working directory using cd command and start the build environment

$ cd Poky

|  |
| --- |
| $ source oe-init-build-env build\_bbb |

Result:

|  |
| --- |
| You had no conf/local.conf file. This configuration file has therefore been created for you with some default values. You may wish to edit it to, for example, select a different MACHINE (target hardware). See conf/local.conf for more information as common configuration options are commented.  You had no conf/bblayers.conf file. This configuration file has therefore been created for you with some default values. To add additional metadata layers into your configuration please add entries to conf/bblayers.conf.  The Yocto Project has extensive documentation about OE including a reference manual which can be found at:  http://yoctoproject.org/documentation For more information about OpenEmbedded see their website:  http://www.openembedded.org/ ### Shell environment set up for builds. ### You can now run 'bitbake <target>'  Common targets are:  core-image-minimal  core-image-sato  meta-toolchain  meta-ide-support  You can also run generated qemu images with a command like 'runqemu qemux86'  Other commonly useful commands are:  - 'devtool' and 'recipetool' handle common recipe tasks  - 'bitbake-layers' handles common layer tasks  - 'oe-pkgdata-util' handles common target package tasks |

**Step 2**

Switch to conf subfolder and open the local.conf using the nano editor

|  |
| --- |
| $cd conf $nano local.conf |

**Step 3**

Select beaglebone-yocto as the target machine by uncommenting it

Editing this part:

|  |
| --- |
| # There are also the following hardware board target machines included for  # demonstration purposes: # MACHINE ?= "beaglebone-yocto" #MACHINE ?= "genericx86" #MACHINE ?= "genericx86-64" #MACHINE ?= "edgerouter" |

**Step 4**

Also, in your local.conf file ensure the following statements are uncommented. Why?

To enable download directory, sstate-cache directory, distribution policy control, package management format.

|  |
| --- |
| DL\_DIR ?= "${TOPDIR}/downloads" SSTATE\_DIR ?= "${TOPDIR}/sstate-cache DISTRO ?= "poky" PACKAGE\_CLASSES ?= "package\_rpm |

**Step 5**

Once the file is ready, save the file and then run the bitbake command

|  |
| --- |
| $bitbake core-image-minimal |

**Step 6**

Now check the path

poky/build\_bbb/tmp/deploy/images/beaglebone-yocto. What files can you see there?

|  |
| --- |
| vy@vy-X550LN:~/yocto/poky/build\_bb/tmp/deploy/images/beaglebone-yocto$ ls am335x-bone--5.8.2+git0+b976de4f41\_912adf166e-r0-beaglebone-yocto-20210401022941.dtb am335x-bone-beaglebone-yocto.dtb am335x-boneblack--5.8.2+git0+b976de4f41\_912adf166e-r0-beaglebone-yocto-20210401022941.dtb am335x-boneblack-beaglebone-yocto.dtb am335x-boneblack.dtb am335x-bone.dtb am335x-bonegreen--5.8.2+git0+b976de4f41\_912adf166e-r0-beaglebone-yocto-20210401022941.dtb am335x-bonegreen-beaglebone-yocto.dtb am335x-bonegreen.dtb core-image-minimal-beaglebone-yocto-20210401022941.qemuboot.conf core-image-minimal-beaglebone-yocto-20210401022941.rootfs.jffs2 core-image-minimal-beaglebone-yocto-20210401022941.rootfs.manifest core-image-minimal-beaglebone-yocto-20210401022941.rootfs.tar.bz2 core-image-minimal-beaglebone-yocto-20210401022941.rootfs.wic core-image-minimal-beaglebone-yocto-20210401022941.rootfs.wic.bmap core-image-minimal-beaglebone-yocto-20210401022941.testdata.json core-image-minimal-beaglebone-yocto.jffs2 core-image-minimal-beaglebone-yocto.manifest core-image-minimal-beaglebone-yocto.qemuboot.conf core-image-minimal-beaglebone-yocto.tar.bz2 core-image-minimal-beaglebone-yocto.testdata.json core-image-minimal-beaglebone-yocto.wic core-image-minimal-beaglebone-yocto.wic.bmap core-image-minimal.env MLO MLO-beaglebone-yocto MLO-beaglebone-yocto-2020.07-r0 modules--5.8.2+git0+b976de4f41\_912adf166e-r0-beaglebone-yocto-20210401022941.tgz modules-beaglebone-yocto.tgz u-boot-beaglebone-yocto-2020.07-r0.img u-boot-beaglebone-yocto.img u-boot.img u-boot-initial-env u-boot-initial-env-beaglebone-yocto u-boot-initial-env-beaglebone-yocto-2020.07-r0 zImage zImage--5.8.2+git0+b976de4f41\_912adf166e-r0-beaglebone-yocto-20210401022941.bin zImage-beaglebone-yocto.bin |

**Step 7:**

Next,insert the microSD card into the host machine. Once inserted, check if the microSD card has been mounted.Use the dmesg command as follows. What is this command for? What is the name of your card?

|  |
| --- |
| vy@vy-X550LN:~$ dmesg | tail [26850.579329] mmcblk0: p1 [26850.955042] EXT4-fs (mmcblk0p1): mounted filesystem with ordered data mode. Opts: (null) [26914.097722] EXT4-fs (mmcblk0p1): mounted filesystem with ordered data mode. Opts: (null) [26920.013193] mmc0: card aaaa removed [26934.214172] mmc0: cannot verify signal voltage switch [26934.334589] mmc0: new ultra high speed SDR104 SDXC card at address aaaa [26934.335572] mmcblk0: mmc0:aaaa SN64G 59.5 GiB  [26934.350233] mmcblk0: p1 [26934.565852] EXT4-fs (mmcblk0p1): mounted filesystem with ordered data mode. Opts: (null)  [27124.410676] perf: interrupt took too long (2509 > 2500), lowering kernel.perf\_event\_max\_sample\_rate to 79500 |

Dmesg Command is used to check all the external connections to the host machine, like: USB devices, SD cards,...

The name of SD card here is mmcblk0

**Step 8:**

Unmount the probable existing partitions using the “unmount” command . Why do we need to do it?

|  |
| --- |
| vy@vy-X550LN:~$ sudo unmount /dev/mmcblk0 |

We need this command in order to unmount the device safely.

**Step 9:**

Launch fdisk utility for partitioning

|  |
| --- |
| vy@vy-X550LN:~$ sudo fdisk /dev/mmcblk0 Welcome to fdisk (util-linux 2.34). Changes will remain in memory only, until you decide to write them. Be careful before using the write command.   Command (m for help): m  Help:   DOS (MBR)  a toggle a bootable flag  b edit nested BSD disklabel  c toggle the dos compatibility flag   Generic  d delete a partition  F list free unpartitioned space  l list known partition types  n add a new partition  p print the partition table  t change a partition type  v verify the partition table  i print information about a partition   Misc  m print this menu  u change display/entry units  x extra functionality (experts only)   Script  I load disk layout from sfdisk script file  O dump disk layout to sfdisk script file   Save & Exit  w write table to disk and exit  q quit without saving changes   Create a new label  g create a new empty GPT partition table  G create a new empty SGI (IRIX) partition table  o create a new empty DOS partition table  s create a new empty Sun partition table |

**Step 10:**

Delete a partition

|  |
| --- |
| Command (m for help): d Partition number (1,2, default 2): 1  Partition 1 has been deleted. |

Create a new partition:

|  |
| --- |
| Command (m for help): n Partition type  p primary (1 primary, 0 extended, 3 free)  e extended (container for logical partitions) Select (default p): p Partition number (1,3,4, default 1): 1 First sector (2048-7372799, default 2048): 2048 Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-7372799, default 7372799): +2048  Created a new partition 1 of type 'Linux' and of size 1 MiB.  Command (m for help): w The partition table has been altered. Calling ioctl() to re-read partition table. Syncing disks. |

Changing type of this partition to FAT32:

|  |
| --- |
| Device Boot Start End Sectors Size Id Type /dev/mmcblk0p1 2048 4096 2049 1M 83 Linux /dev/mmcblk0p2 7372800 124735487 117362688 56G c W95 FAT32 (LBA)  Command (m for help): t Partition number (1,2, default 2): 1 Hex code (type L to list all codes): L   0 Empty 24 NEC DOS 81 Minix / old Lin bf Solaris   1 FAT12 27 Hidden NTFS Win 82 Linux swap / So c1 DRDOS/sec (FAT-  2 XENIX root 39 Plan 9 83 Linux c4 DRDOS/sec (FAT-  3 XENIX usr 3c PartitionMagic 84 OS/2 hidden or c6 DRDOS/sec (FAT-  4 FAT16 <32M 40 Venix 80286 85 Linux extended c7 Syrinx   5 Extended 41 PPC PReP Boot 86 NTFS volume set da Non-FS data   6 FAT16 42 SFS 87 NTFS volume set db CP/M / CTOS / .  7 HPFS/NTFS/exFAT 4d QNX4.x 88 Linux plaintext de Dell Utility   8 AIX 4e QNX4.x 2nd part 8e Linux LVM df BootIt   9 AIX bootable 4f QNX4.x 3rd part 93 Amoeba e1 DOS access   a OS/2 Boot Manag 50 OnTrack DM 94 Amoeba BBT e3 DOS R/O   b W95 FAT32 51 OnTrack DM6 Aux 9f BSD/OS e4 SpeedStor   c W95 FAT32 (LBA) 52 CP/M a0 IBM Thinkpad hi ea Rufus alignment  e W95 FAT16 (LBA) 53 OnTrack DM6 Aux a5 FreeBSD eb BeOS fs   f W95 Ext'd (LBA) 54 OnTrackDM6 a6 OpenBSD ee GPT  10 OPUS 55 EZ-Drive a7 NeXTSTEP ef EFI (FAT-12/16/ 11 Hidden FAT12 56 Golden Bow a8 Darwin UFS f0 Linux/PA-RISC b 12 Compaq diagnost 5c Priam Edisk a9 NetBSD f1 SpeedStor  14 Hidden FAT16 <3 61 SpeedStor ab Darwin boot f4 SpeedStor  16 Hidden FAT16 63 GNU HURD or Sys af HFS / HFS+ f2 DOS secondary  17 Hidden HPFS/NTF 64 Novell Netware b7 BSDI fs fb VMware VMFS  18 AST SmartSleep 65 Novell Netware b8 BSDI swap fc VMware VMKCORE  1b Hidden W95 FAT3 70 DiskSecure Mult bb Boot Wizard hid fd Linux raid auto 1c Hidden W95 FAT3 75 PC/IX bc Acronis FAT32 L fe LANstep  1e Hidden W95 FAT1 80 Old Minix be Solaris boot ff BBT    Hex code (type L to list all codes): b   Changed type of partition 'Linux' to 'W95 FAT32'.  Command (m for help): w The partition table has been altered. Calling ioctl() to re-read partition table. Syncing disks. |

Enable BOOT on first partition:

|  |
| --- |
| Command (m for help): a Partition number (1,2, default 2): 1  The bootable flag on partition 1 is enabled now.  Command (m for help): w The partition table has been altered. Calling ioctl() to re-read partition table. Syncing disks |

**Step 11**

Format the first partition using the following command. What is the impact of this command?

|  |
| --- |
| $vy@vy-X550LN sudo mkfs.vfat -n "BOOT" /dev/mmcblk0p1  mkfs.fat 4.1 (2017-01-24) |

Changing system label of partition 1 to BOOT

**Step 12**

format the second partition using the following command. What is the impact of this command?

|  |
| --- |
| $vy@vy-X550LN sudo mkfs.ext4 -n "ROOT" /dev/mmcblk0p2  mkfs.ext4: invalid blocks '/dev/mmcblk0p2' on device 'ROOT' |

Changing system label of partition 2 to ROOT

Alternative method is using gparted for both step 11 and step 12

**Step 13**

Now we can copy the files into the SD card by first mounting both the partitions using the mount command . At first change directory to the path “poky/build\_bbb/tmp/deploy/images/beaglebone-yocto” and then try to copy the MLO, U-Boot into FAT32 and the kernel image in the boot partition along with .dtb file (which is necessary only for core-image-minimal):

|  |
| --- |
| vy@vy-X550LN:~/yocto/poky/build\_bb/tmp/deploy/images/beaglebone-yocto$ sudo cp MLO /media/$USER/BOOT  vy@vy-X550LN:~/yocto/poky/build\_bb/tmp/deploy/images/beaglebone-yocto$ sudo cp u-boot.img /media/$USER/BOOT  vy@vy-X550LN:~/yocto/poky/build\_bb/tmp/deploy/images/beaglebone-yocto$ sudo cp zImage /media/$USER/BOOT  vy@vy-X550LN:~/yocto/poky/build\_bb/tmp/deploy/images/beaglebone-yocto$ sudo cp MLO-beaglebone-yocto /media/$USER/BOOT vy@vy-X550LN:~/yocto/poky/build\_bb/tmp/deploy/images/beaglebone-yocto$ sudo tar -xf core-image-minimal-beaglebone-yocto.tar.bz2 -C/media/$USER/ROOT |

**Step 14**

Insert the microSD card in the beaglebone black and then start it to boot through the SD card. Once booted through the SD card we can communicate with beagle bone through the serial communication such as Minicom or screen. Try Minicom. For this purpose, do you need to install Minicom? Which cable shall you use then?

On the host machine type:

|  |
| --- |
| vy@vy-X550LN:~ sudo minicom |

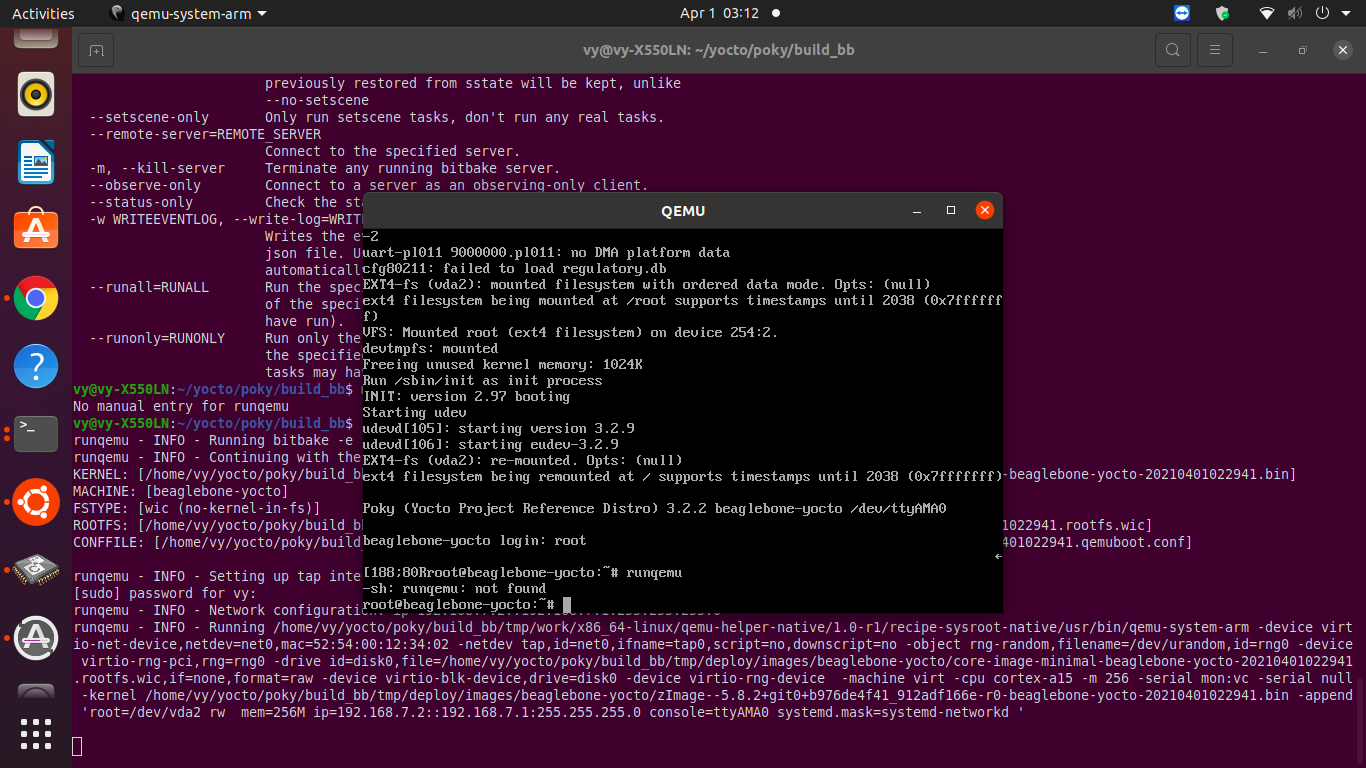
We do need to install minicom for this purpose. FTDI cable is used in this case.

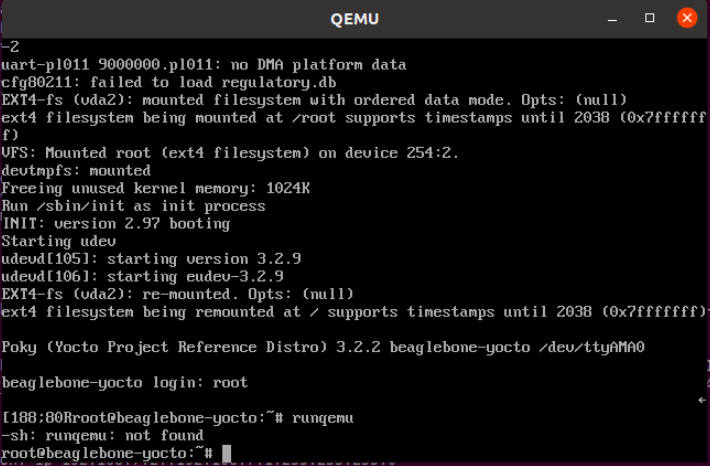
**Step 15**

Now login into the beagle bone, the username is root and the password is blank

|  |
| --- |
| zImage /media/$USER/BOOT  Poky (Yocto Project Reference Distro) 3.2.2 beaglebone-yocto /dev/ttyS0  beaglebone-yocto login: root root@beaglebone-yocto:~# |

**Another method is using qemu on the host machine to run yocto on BBB:**





**Step 16**

Try to see what commands are available in the image. How? Why do we have such a limited number of commands there?

|  |
| --- |
| Poky (Yocto Project Reference Distro) 3.2.2 beaglebone-yocto /dev/ttyS0  beaglebone-yocto login: root root@beaglebone-yocto:~# ls root@beaglebone-yocto:~# cd .. root@beaglebone-yocto:/home# cd .. root@beaglebone-yocto:/# ls bin dev home lost+found mnt run sys usr boot etc lib media proc sbin tmp var root@beaglebone-yocto:/# pwd / root@beaglebone-yocto:/# cd root@beaglebone-yocto:~# pwd /home/root root@beaglebone-yocto:~# |

The commands are limited because we are using the minimal build of yocto.

**Part B**

Create an Image for BeagleBone Black preferably sato image using the Yocto project which has the GUI and make the Beaglebone run from the image flashed on the MicroSD Card and display the GUI using a screen.

**The second part of the lab, from step 1 to step 6 are similar to part A.**

**Step 1**

Change your working directory first to poky and then type the following command.

|  |
| --- |
| $ cd Poky |

|  |
| --- |
| $ source oe-init-build-env build\_bbb |

**Step 2**

Switch to conf subfolder and open local.conf file.

|  |
| --- |
| $cd conf $nano local.conf |

**Step 3**

Try to select beaglebone-yocto as your target machine there. in the local.conf file ensure the following statements are uncommented.

|  |
| --- |
| DL\_DIR ?= "${TOPDIR}/downloads" SSTATE\_DIR ?= "${TOPDIR}/sstate-cache DISTRO ?= "poky" PACKAGE\_CLASSES ?= "package\_rpm" |

**Step 4**

Once the file is ready, save the file and then run the bitbake command

|  |
| --- |
| vy@vy-X550LN:~/yocto/poky/build\_bbb$ bitbake core-image-sato Loading cache: 100% | | ETA: --:--:-- Loaded 0 entries from dependency cache. Parsing recipes: 100% |##########################################################################################################################| Time: 0:01:01 Parsing of 784 .bb files complete (0 cached, 784 parsed). 1354 targets, 61 skipped, 0 masked, 0 errors. NOTE: Resolving any missing task queue dependencies  Build Configuration: BB\_VERSION = "1.48.0" BUILD\_SYS = "x86\_64-linux" NATIVELSBSTRING = "ubuntu-20.04" TARGET\_SYS = "arm-poky-linux-gnueabi" MACHINE = "beaglebone-yocto" DISTRO = "poky" DISTRO\_VERSION = "3.2.2" TUNE\_FEATURES = "arm vfp cortexa8 neon callconvention-hard" TARGET\_FPU = "hard" meta  meta-poky  meta-yocto-bsp = "gatesgarth:d5d6286a66f46f4523e35e0e3f20cd7396195fdc"  NOTE: Fetching uninative binary shim http://downloads.yoctoproject.org/releases/uninative/2.10/x86\_64-nativesdk-libc.tar.xz;sha256sum=04333677f81990ce2cf55c3bc256cd84a66085d18fc95ccddfab8581e4aec014 (will check PREMIRRORS first) Initialising tasks: 100% |#######################################################################################################################| Time: 0:00:05 Sstate summary: Wanted 2694 Found 0 Missed 2694 Current 0 (0% match, 0% complete) NOTE: Executing Tasks NOTE: Tasks Summary: Attempted 6792 tasks of which 1 didn't need to be rerun and all succeeded. |

**Step 5**

check the path “poky/build\_bbb/tmp/deploy/images/beaglebone-yocto.” What files can you see there?

|  |
| --- |
| vy@vy-X550LN:~/yocto/poky/build\_bbb$ cd tmp/deploy/images/beaglebone-yocto vy@vy-X550LN:~/yocto/poky/build\_bbb/tmp/deploy/images/beaglebone-yocto$ ls am335x-bone--5.8.2+git0+b976de4f41\_912adf166e-r0-beaglebone-yocto-20210401072813.dtb am335x-bone-beaglebone-yocto.dtb am335x-boneblack--5.8.2+git0+b976de4f41\_912adf166e-r0-beaglebone-yocto-20210401072813.dtb am335x-boneblack-beaglebone-yocto.dtb am335x-boneblack.dtb am335x-bone.dtb am335x-bonegreen--5.8.2+git0+b976de4f41\_912adf166e-r0-beaglebone-yocto-20210401072813.dtb am335x-bonegreen-beaglebone-yocto.dtb am335x-bonegreen.dtb core-image-sato-beaglebone-yocto-20210401072813.qemuboot.conf core-image-sato-beaglebone-yocto-20210401072813.rootfs.jffs2 core-image-sato-beaglebone-yocto-20210401072813.rootfs.manifest core-image-sato-beaglebone-yocto-20210401072813.rootfs.tar.bz2 core-image-sato-beaglebone-yocto-20210401072813.rootfs.wic core-image-sato-beaglebone-yocto-20210401072813.rootfs.wic.bmap core-image-sato-beaglebone-yocto-20210401072813.testdata.json core-image-sato-beaglebone-yocto.jffs2 core-image-sato-beaglebone-yocto.manifest core-image-sato-beaglebone-yocto.qemuboot.conf core-image-sato-beaglebone-yocto.tar.bz2 core-image-sato-beaglebone-yocto.testdata.json core-image-sato-beaglebone-yocto.wic core-image-sato-beaglebone-yocto.wic.bmap core-image-sato.env MLO MLO-beaglebone-yocto MLO-beaglebone-yocto-2020.07-r0 modules--5.8.2+git0+b976de4f41\_912adf166e-r0-beaglebone-yocto-20210401072813.tgz modules-beaglebone-yocto.tgz u-boot-beaglebone-yocto-2020.07-r0.img u-boot-beaglebone-yocto.img u-boot.img u-boot-initial-env u-boot-initial-env-beaglebone-yocto u-boot-initial-env-beaglebone-yocto-2020.07-r0 zImage zImage--5.8.2+git0+b976de4f41\_912adf166e-r0-beaglebone-yocto-20210401072813.bin zImage-beaglebone-yocto.bin |

**We can see all the files that are contained in this build.**

**Step 6**

insert the microSD card into the host machine Use the dmesg command as follows.

|  |
| --- |
| vy@vy-X550LN:~$ dmesg | tail [102004.840023] rndis\_host 2-1:1.0 enx0cb2b7d5757c: renamed from eth0 [102005.695369] cdc\_ether 2-1:1.2 enx0cb2b7d5757e: renamed from eth1 [102037.131930] usb 2-1: USB disconnect, device number 12 [102037.132040] rndis\_host 2-1:1.0 enx0cb2b7d5757c: unregister 'rndis\_host' usb-0000:00:14.0-1, RNDIS device [102037.170703] cdc\_ether 2-1:1.2 enx0cb2b7d5757e: unregister 'cdc\_ether' usb-0000:00:14.0-1, CDC Ethernet Device [103535.684286] mmc0: cannot verify signal voltage switch [103535.806381] mmc0: new ultra high speed SDR104 SDXC card at address aaaa [103535.807264] mmcblk0: mmc0:aaaa SN64G 59.5 GiB  [103535.820829] mmcblk0: p1 p2 [103536.307847] EXT4-fs (mmcblk0p2): mounted filesystem with ordered data mode. Opts: (null) |

**Step 7**

use dd command whose primary purpose is to convert and copy the files and make an image on the SD card. Explain about the following command in more details

|  |
| --- |
| vy@vy-X550LN:~/yocto/poky/build\_bbb/tmp/deploy/images/beaglebone-yocto$ sudo dd if=core-image-sato-beaglebone-yocto.wic of=/dev/mmcblk0 1264958+0 records in 1264958+0 records out 647658496 bytes (648 MB, 618 MiB) copied, 69.8319 s, 9.3 MB/s |

-dd command is used to convert and copy file

- if=FILE option is to read from FILE instead of stdin

-of=FILE write to FILE instead of stdout

**Step 8**

Once all the steps are completed insert the microSD card in the beaglebone black and then start it to boot through the SD card. Once booted through the SD card and connect it through the micro HDMI to HDMI cable to a screen like a device such as a monitor or a television screen. Explain about this cable in more detail. What can you see on the screen now?

Since we do not have the micro HDMI to HDMI cable and the monitor, we login into the BBB via minicom on the host machine.

|  |
| --- |
| Poky (Yocto Project Reference Distro) 3.2.2 beaglebone-yocto /dev/ttyS0  beaglebone-yocto login: root root@beaglebone-yocto:~# ls root@beaglebone-yocto:~# cd .. root@beaglebone-yocto:/home# cd .. root@beaglebone-yocto:/# ls bin dev home lost+found mnt run sys usr boot etc lib media proc sbin tmp var root@beaglebone-yocto:/# cd boot root@beaglebone-yocto:/boot# ls MLO am335x-bone.dtb am335x-boneblack.dtb am335x-bonegreen.dtb extlinux u-boot.img zImage root@beaglebone-yocto:/boot# |