BSP CUSTOMIZATION AND PORTING OF LINUX ON ARM CORTEX BASED I.MX6 PROCESSOR WITH YOCTO BUILD ENVIRONMENT

Manjunath Joshi[1], Harsha B.K.[2], Vivek Kaushik[3], Harish Mekali[4]

[1]*Student, Electronics and communication Department, CMRIT, Karnataka, India*
[2]*Assistant Professor, Electronics and communication Department, CMRIT, Karnataka, India*
[3]*Project Manager, Software Department, TES Electronic Solutions, Karnataka, India*
[4]*Assistant Professor, Electronics and communication Department, BMSCE, Karnataka, India*

Abstract

In the last few years, the need for compact and embedded systems has expanded in all fields. With regard to this development, ARM development platform is the ideal and practical answer for planning a new product design. ARM platforms carry a generally positive result regarding speed, accuracy, adaptability, size and cost. Every new embedded design, at the back an operating system must be modified appropriately for the specifications of the embedded target. It would be a huge added advantage to have an embedded design running an OS. In particular, this paper underlines the BSP (Board Support Package) customization of Linux operating system and porting mechanism to FreeScale SabreSD an i.MX6 processor based embedded board with the help of Yocto build environment. The usual method of going for Linux Target Image Builder (LTIB) is considered old and Yocto is specifically preferred as Freescale is emphasizing on lighter and easier version. Yocto is a dream project by Freescale. Successful build of Yocto environment enables customization of kernel, through which images of OS are built and they are ported to target platform.

Keywords: *ARM, Bootloader, BSP Customization, Embedded Systems, Filesystem, Linux Ubuntu, Porting OS. SabreSD, Yocto Build Environment.*

---*--**

1. INTRODUCTION

Linux has been available for the ARM architecture for many years now. The original "port" was done by Russell King and he is still the maintainer through whom all ARM kernel patches generally pass [7]. Linux is now the preferred operating system for many embedded devices - mainly due to the efficient and portable design of the Linux kernel.

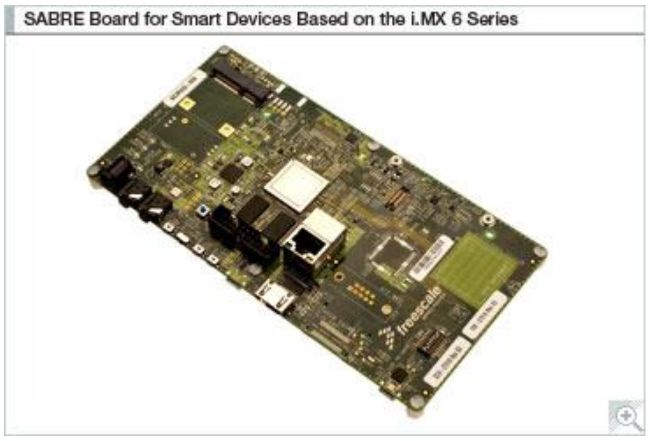


Fig-1: Freescale SabreSD board (courtesy: Freescale Semiconductor, Inc)

An Embedded system is application oriented special computer system which is accessible on both software and hardware. It can satisfy the strict necessity of functionality, consistency, cost, size, and power consumption of the specific application. With the extremely fast development of IC design and manufacture, CPUs became inexpensive. Lots of consumer electronics have embedded CPU and thus embedded systems became more popular. For example, Tablets, point-of-sale devices, industrial control panels, or even your washing machine can be embedded system. There is a greater extent demand on the embedded system market. According to the present scenario, the demand on embedded CPUs is more times as large as general purpose CPUs. As applications of the embedded systems become more multifaceted, to build the operating system and preparing development environment became crucial [9].

1.1 Embedded Linux System

Figure 1 shows the layered architecture based upon the OS directory structure, and also indicates the how the application in the device accessing the hardware. The main concentration is only the Board Support Packages. It depends on the architecture of that OS. If the architecture is ARM, then the corresponding will be created according to the target platform. The BSP is in detailed as follows.

Board Support Packages *(*BSP) is a collection of the binary, code, and support files that can be used to create a Linux kernel firmware and Filesystem images, for a particular target. In other words a Board Support Package (BSP) is an implementation specific support code for a given board that conforms to a given operating system. It has commonly had a boot loader that contains the minimal device support to load the operating system and device drivers for all the devices in a target system.

BSP in a layman's words can be a menu from which he can choose. The menu contains all the essential features that a board can have. Customization is specific to a board as to whether it supports that particular feature or not.

Porting on the other hand is making sure that the software which was customized, runs on the target platform.

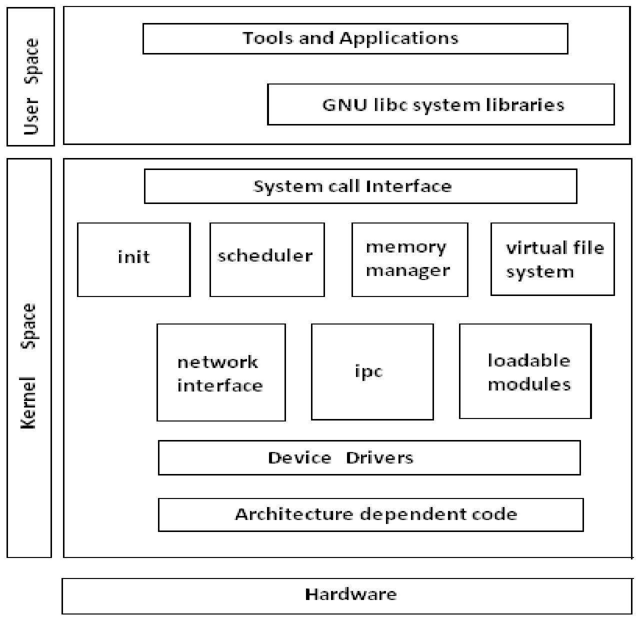


Fig-2: Embedded Linux System layered architecture.

2. THE EMBEDDED OPERATING SYSTEM

The term Operating system is referred to as; it's a special code that acts as an intermediate between the hardware and the user[5]. The main goals of the operating system is to make the system is convenient to use (Hiding the hardware details) and utilizing the resources in efficient manner[1]. The following are the most important factors to choose an Embedded Operating System[4].

Full source availability, Technical support, real-time performance, compatibility, customizable, open source, the processor it supports, purchase price, simplicity/easy to use, availability of the software development tools, small memory footprint, middleware/software/drivers and finally it is also supports the other architectures also. The layered architecture of basic Embedded Linux is shown in the Figure 2.

The operating system can be divided into three modules. They are Bootloader, Kernel, Filesystem.

Bootloader is an initializing code for a particular board, which is executed at the power on or reset the board. Here the proposed boot loader is U-Boot boot loader. To boot the Linux, the boot loader has to load the modules into the memory, one is the Linux kernel and another one is the Filesystem [6].

Kernel is a software layer that interfaces with the hardware. It is responsible for interfacing all peripherals that are currently connected to the system and running in "user mode" down to the physical hardware, and allowing processes, to get information from each other using inter-process communication[9].

Filesystem is the way in which files are named and where they are placed logically for storage and retrieval. The DOS, Windows, Macintosh, and UNIX-based operating systems all have Filesystems in which files are placed somewhere in a hierarchical (tree) structure. A file is placed in a directory or subdirectory at the preferred place in the tree structure. Filesystems require conventions for naming files. A Filesystem also includes a format for identifying the path to a file through the structure of directories.

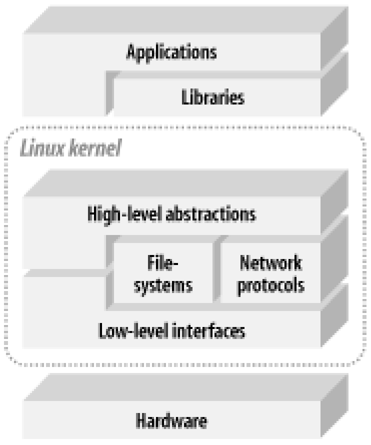


Fig-3: Basic Embedded Linux Structure

3. BUILDING LINUX KERNEL PLATFORM

This section incorporates to set up the building environment, install and run the Yocto, and finally generate the output binary files to prepare the SD card bootable image compatibles. The main objective of this paper is to make the Embedded Linux OS according to the target platform; in this instance, it is the I.MX6 application processor platform, which is developed by the Freescale Semiconductors. The required components to build the OS are bootloader, kernel and Filesystem, Of course the development of the Operating System image individually by selecting the bootloader, kernel and Filesystem, but it is very tedious job to do such kind of selection, as per time to market constraint, vendors are developing the target image builders. In this paper proposed building tool is the Yocto. The Yocto project can be used to develop and deploy the Board Support Packages for various target platforms.

3.1 Setting up the Host PC

Requirements:

- Ubuntu 12.04 (32/64 bit) OS version on host machine.

- 100 gigabytes of free disk space for building images.

Note: The Open Embedded build system should be able to run on any modern distribution that has the following versions for Git, tar, and Python.

- Git 1.7.5 or greater
- tar 1.24 or greater
- Python 2.7.3 or greater excluding Python 3.x, which is not supported.

Packages: sudo apt-get install #required packages

3.2 Yocto Build Environment Latest Release

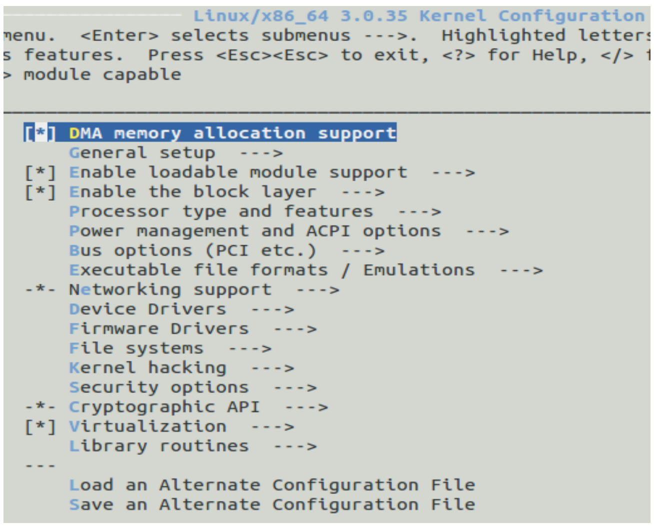
The latest *Yocto* build can be installed locally on development system. The user manual can be referred from Yocto reference manual [10]. This manual explains how to set up build environment, to configure and to compile.

3.3 Configuration

To fit the Embedded Linux on the hardware platform, the configuration must be changed according to the type of application, so that drivers must be included but default kernel configurations. It also effects the boot time of the device.

In the kernel some of the configuration changes may required to the target device. It depends on the application running on that device and it affects the final footprint of the binary image. Here some of the configurations like in the Figure 5, in those some are ‘Enable the loadable module support’, System type: select imx233, Boot options: This is the one of the significant configuration, for example ***console=ttyS0 115200n8:*** For displaying log screen while booting the board with the baud rate of 115200, ***rootwait:*** for detecting the devices asynchronously like USB or MMC, ***rw:*** mount root device read write on boot, ***initrd:*** to specify the location of the initial ramdisk, ***rootfstype:*** to select the type of root filesystem .

It depends on the target boot device, if the device is an SD card, it supports Extended filesystem (ext2) or it may be the NAND flash it supports Journaling flash Filesystem (jffs2). lcd_panel=lms430: which shows the type of LCD used and its resolution. The configured drivers are the Memory Technology device drivers (MTD), Block Devices, I2C Support, GPIO Support, Multimedia support, USB Support, MMC Card support, Real time clock, Sound card support, Power supply Class support, Watchdog timer support. These are the required drivers for customizing. The selection of options either <*> or <M>, modules as per our requirement. Figure 5 represents the configuration of kernel.



```
                    Linux/x86_64 3.0.35 Kernel Configuration
nenu.  <Enter> selects submenus --->.  Highlighted letter
s features.  Press <Esc><Esc> to exit, <?> for Help, </> f
> module capable

 [*] DMA memory allocation support
     General setup  --->
 [*] Enable loadable module support  --->
 [*] Enable the block layer  --->
     Processor type and features  --->
     Power management and ACPI options  --->
     Bus options (PCI etc.)  --->
     Executable file formats / Emulations  --->
 -*- Networking support  --->
     Device Drivers  --->
     Firmware Drivers  --->
     File systems  --->
     Kernel hacking  --->
     Security options  --->
 -*- Cryptographic API  --->
 [*] Virtualization  --->
     Library routines  --->
 ---
     Load an Alternate Configuration File
     Save an Alternate Configuration File
```

Fig-4: kernel configuration by menuconfig

Configuration is to set up the kernel and u-boot as per the board specifications of our interest. U-boot is open source and is available at http://sourceforge.net/projects/uboot/ or can be downloaded from Freescale community website if you are a registered user.

4. COMPILATION

4.1 Compiling Kernel Source

We need to set the path to cross compile. Tool chain is a cross compiler with necessary libraries, binary utilities.

We are using fsl-linaro-toolchain as a cross compiler.
$ export ARCH=arm
$ export CROSS_COMPILE=arm-none-linux-gnueabi-
$ export PATH=$PATH: /opt/freescale/usr/local/gcc-4.6.2-glibc-2.13-linaro-multilib-2011.12/fsl-linaro-toolchain/bin
$ make sabresd_defconfig
$ make uImage
Output file uImage will be under arch/arm/boot/uImage

4.2 Compiling u-boot

Cross-compiler for arm v9: arm-none-linux-gnueabi-gcc

4.3 Compiling Filesystem

Ubuntu filesystem is freely available on the freescale community.

Select machine and prepare the bitbake’s environment
$ MACHINE=imx6qsabresd source setup-environment build-fb

• bake the image recipe

$ bitbake #Linux image. Bake! The first time can take several hours.

5. SABRESD BOOT SET UP

5.1 Booting the i.MX 6Dual/6Quad SABRE-SD Board

The boot modes of the i.MX 6Dual/6Quad SABRE-SD board are controlled by the boot configuration DIP switches on the board.

Serial Download Mode for MFGTool

No dedicated boot dips are reserved for serial download mode on i.MX 6 SABRE-SD boards. Therefore, a tricky method is used to enter serial download mode. After "HID-Compliant device" is detected, it means that the board has entered serial download mode. Insert the SD card into SD3 slot.

Another way to do this is to configure an invalid boot switch setting, for example, set all the dips of SW6 to off. [11]

5.2 Booting from SD Card from Slot2

The following tables show the dip switch settings for SD2 boot.

Table-1: Boot switch setup for SD2 boot (J500) Switch:

S	D1	D2	D3	D4	D5	D6	D7	D8
S6	ON	OFF	OFF	OFF	OFF	OFF	ON	OFF

Note: *S stands for Switch and S6 is Switch 6.*

Table-2: Boot switch set up for SD Card from Slot3

S	D1	D2	D3	D4	D5	D6	D7	D8
S6	OFF	ON	OFF	OFF	OFF	OFF	ON	OFF

Table-3: Boot switch set up for SD Card from eMMC 4.4

S	D1	D2	D3	D4	D5	D6	D7	D8
S6	ON	ON	OFF	ON	OFF	ON	ON	OFF

Table-4: Boot switch set up for SD Card from SATA

S	D1	D2	D3	D4	D5	D6	D7	D8
S6	ON	ON	OFF	ON	OFF	ON	ON	OFF

6. RESULTS

The successful building of the Yocto generates a file, which is an encrypted file that is bootable on i.mx6 series (i.e. ARM9 based development boards).

Next few steps explain flashing images and root file-system to SD card.

6.1 Arm Board Serial Terminal Set-Up

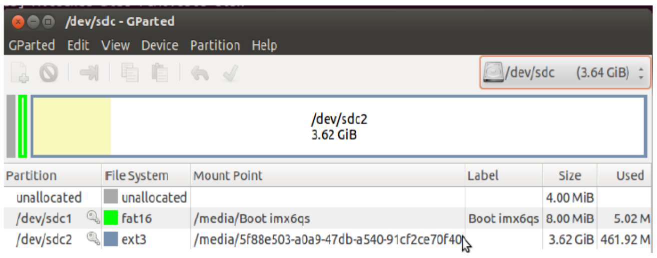
For a Linux machine, a serial terminal such as Minicom can be used. Minicom is a text-based modem control and terminal emulation program for Unix-like operating systems. Gtkterm is a terminal emulator written with GTK+. It is lightweight and simple that drives serial ports Ubuntu users wanting a graphical terminal program can install gtkterm. Later versions which include many bug fixes can be obtained from the current maintainer's website. On a Windows PC, you could use HyperTerminal or TeraTerm. Insert the SD card into the SD card slot of the target board. Connect the target board to host machine using RS232 serial cable. Apply power supply to the target board[7].

6.2 Creating Partition in SD Card

Pre-requisite: 4GB SD/MMC card, Card Reader
Flashing Utility: Gparted

Install 'Gparted' utility and run it after installation.
$ sudo apt-get install Gparted

Select 4 GB SD/MMC Card from drop-down menu (shown in Fig-5 as 3.64 GiB)



Partition	File System	Mount Point	Label	Size	Used
unallocated	unallocated			4.00 MiB	
/dev/sdc1	fat16	/media/Boot imx6qs	Boot imx6qs	8.00 MiB	5.02 M
/dev/sdc2	ext3	/media/5f88e503-a0a9-47db-a540-91cf2ce70f40		3.62 GiB	461.92 M

Fig-5: Creating partitions in SD card using Gparted

The card will be mounted and any existing partitions will be visible. First unmount all the partitions and then delete those partitions.

Create new partitions.

- Leave 100 MB as Free Space preceding
- Choose Filesystem as **ext4**
- Give label as **rootfs**
- Click Add to create the partition

6.3 Requirement

An SD/MMC card reader, it will be used to transfer the boot loader and kernel images to initialize the partition table and copy the root filesystem. To simplify the instructions, it is assumed that a 4GB SD/MMC card is used.

The Linux kernel running on the Linux host will assign a device node to the SD/MMC card reader.

To identify the device node assigned to the SD/MMC card, enter the command:

```
$ cat /proc/partitions
```

6.4 Copying Root Filesystem to SD Card

First, a partition table must be created. Create a partition; at offset 16384 (in sectors of 512 bytes) enter command:

$ sudo fdisk /dev/sdb

The Filesystem format ext3 or ext4 is a good option for removable media due to the built in journaling. To format the partition:

$ sudo mkfs.ext3 /dev/sdb1Or $ sudo mkfs.ext4 /dev/sdb1

6.5 Copy the Target Filesystem to the Partition:

$ mkdir /home/user/mountpoint

$ sudo mount /dev/sdb1 /home/user/mountpoint

Extract sabresd_rootfs package to certain directory: extract sabresd_rootfs.tar.bz2 to /home/user/ mountpoint

$ tar –xvjf sabresd_rootfs.tar.bz2 –C /home/user/mountpoint

Note:Ubuntu filesystem may be freely available for demo purpose on websites.As an example OS it can be used along with our customized kernel and U-boot.

6.6 Copying Boot Loader Image to SD Card

Enter the following command to copy the U-Boot image to the SD/MMC card:

For non padded U-Boot image:

$ sudo dd if=u-boot.imx of=/dev/sdb bs=1k seek=1

For padded U-Boot image:

$ sudo dd if=u-boot.bin of=/dev/sdb bs=1k seek=1 skip=1

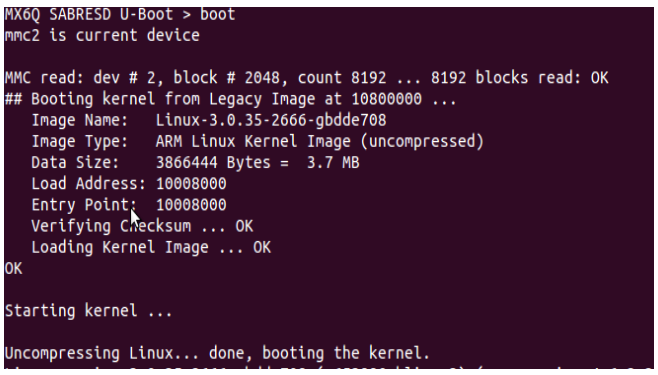
The first 1 KB of the SD/MMC card, that includes the partition table, will be preserved.

6.7 Copying Kernel Image to SD Card

Copy the kernel image to the SD/MMC card:

$ sudo dd if=uImage of=/dev/sdb bs=1M seek=1

This will copy uImage to the media at offset 1 MB (bs x seek = 1 M x 1 = 1MB).



```
MX6Q SABRESD U-Boot > boot
mmc2 is current device

MMC read: dev # 2, block # 2048, count 8192 ... 8192 blocks read: OK
## Booting kernel from Legacy Image at 10800000 ...
   Image Name:   Linux-3.0.35-2666-gbdde708
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    3866444 Bytes =  3.7 MB
   Load Address: 10008000
   Entry Point:  10008000
   Verifying Checksum ... OK
   Loading Kernel Image ... OK
OK

Starting kernel ...

Uncompressing Linux... done, booting the kernel.
```

Fig-6: Kernel image in host system

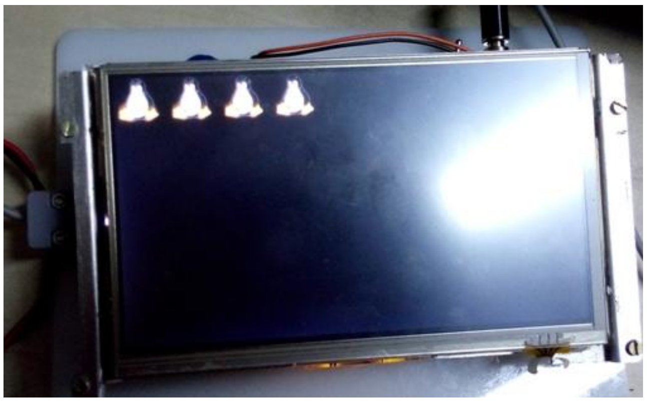


Fig-7: Linux boot process on Target board (4 penguins for quad core processor).

The *boot time* is an important factor. It depends on the number of features supported by the board which extends size of the kernel and the filesystem. The kernel boot with filesystem usually takes less than a minute.

Fig-8: Example filesystem linux ubuntu on target board.

7. CONCLUSIONS

This paper discussed about Yocto build environment and was successfully installed on the host system Ubuntu 12.04 LTS. U-boot, Kernel and filesystem were successfully configured and cross-compiled as per Freescale Sabresd embedded board specifications. At the end as an example Ubuntu filesystem was ported to the same board. The boot time recorded was 32 seconds.

ACKNOWLEDGEMENTS

I would like to thank my master and my parents.

REFERENCES

[1]. De Goyeneche, J.-M, De Sousa, E.A.F, "*Loadable Kernel Modules*", Software IEEE, Vol16, Issue1, pp:65-71, Jan/Feb- 1999
[2]. Wooking and Tak-Shing, "Porting the Linux kernel to a new ARM Platform", Aleph One, vol. 4, summer 2002.
[3]. Vincent Sanders, "Booting ARM Linux", rev.1.10,June2004.http://www.simtec.co.uk/products/SWLINUX/files/booting_article.html

[4]. Hu Jie ; Zhang Gen-bao, "Research transplantation method of embedded linux kernel based on ARM platform", *Information Science and Management Engineering* (ISME), 2010 International Conference, Vol2, pp:35-38, 7-8. Aug 2010.
[5]. Chun-yue Bi; Yun-peng Liu; Ren-fang Wang; "Research of key technologies for embedded Linux based on ARM", Computer Application and System Modeling (ICCASM),2010 International Conference, 22-24 Oct. 2010, E-ISBN : 978-1-4244-7237-6.
[6]. The DENX U-Boot and Linux Guide, available at www.denx.de
[7]. Pratyusha Gandham, Ramesh N.V.K "Porting the linux kernel to an arm based development board", International Journal of Engineering Research and Applications (IJERA), Vol. 2, Issue 2,Mar-Apr 2012, pp.1614-1618.
[8]. Divya Sharma, Kamal kanth, "Porting the Linux Kernel to Arm System-On-Chip And Implementation of RFID Based Security System Using ARM", International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSC), Vol3, issue5, May-2013.
[9]. K.Eshwar Kumar, M.Kamaraju, Ashok Kumar Yadav " Porting and BSP Customization of Linux on ARM Platform"International Journal of Computer Applications (0975 – 8887) Volume 80 – No 15, October 2013.
[10]. http://www.yoctoproject.org/docs/1.4/ref-manual/ref manual.html#required-packages-for-thehost-development-system.
[11]. i.MX 6Dual/6Quad BSP Porting Guide , Rev. L3.0.35_1.1.0, 01/2013, Freescale Semiconductor,inc.

BIOGRAPHIES

Manjunath Joshi completed his B.E. from VTU Belgaum, Karnataka, India. He is currently pursuing Masters at CMRIT Bangalore and is working as an intern at TES Electronic Solutions, a German based Embedded Design Company.

Harsha B.K is currently working as an assistant professor at CMRIT Bangalore. He did his masters from VTU Belgaum.

Vivek Kaushik completed his bachelors from Chandigarh. He has ten years of experience in embedded field and is currently Project Manager at TES Electronic Solutions.

Harish Mekali has graduated from PESIT Bangalore. He has been key figure in many research works at BMSCE