Converting Existing LCD TV to Smart TV using BeagleBoard XM

Sujith Kumar .S, Dr. Ramachandra .A .C

Abstract— Earlier days, in LCD TV we were able to watch only channels and sometimes built in games. If we do text messaging, online gaming, MMS, video charting, Old persons who does not know how to operate computer they can directly use this application to perform video conference with their far relatives by simply switch on the device etc... In separate devices it will be wastage of area, power, time, and money. So by keeping this issue we have developed a solution in this paper. The aim of this paper is to Converting Existing LCD TV to Smart TV Using BeagleBoard XM. This processor operating at frequency of 1GHz in order to improve the execution speed. ARM Cortex A8 processor is available in BeagleBoard XM in market. Beagle Board XM is designed specifically to address the Open Source Community. Hence its hardware design and details about the processor are completely available in websites. It offers high re configurability and portability. In this paper host operating system is Linux, target operating system is android, system software using c language and application software is developed using java language. The application developed in this paper is contributed as open source code to the society. In future every common people relay on Smart TV with android.

Index Terms— **Android, BeagleBoard XM, Linux, Smart TV.**

1 INTRODUCTION

Smart TV systems includes features such as MMS, messaging, video conference, watching online video channels, google talk, browsing websites and also used as touch screen devices etc…

Android is run by company called Google.inc and it became more popular now a days. In recent years android became more efficient target operating because it is very to build android applications as compared to Linux operating. Android is open source so source code is available and source code for android application also available. These open standards are developed by open handset alliance (OHA) and application receives regular update from the android cloud. Android features are Messaging, Web browser, Voice based feature, Multi-touch, Bluetooth, External storage, Multitasking etc.

BeagleBoard XM is designed specifically to address the Open Source Community. It is a single board fan less computer with all interfacing ports as similar to our personal computer. It is a more supporting open source device for all open source code to build an android application. We customise android O.S for BeagleBoard XM and application software developed using java language for BeagleBoard XM. By utilizing standard interfaces, the BeagleBoard is highly extensible to add many features and interfaces. It is not intended for use in end products. All of the design information is freely available and can be used as the basis for a product. This is readily available in market for product development purpose.

In this paper, few of the smart TV applications such as watching online video channels and audio player are implemented. In LCD we can able to watch all online video channels.

2 EXISTING MODEL

The existing Smart TV system can perform audio player and online video channel watching. This model uses open source hardware component called BeagleBoard XM and open source operating system called Android and Linux. LCD with DVI port can use for display purpose.

2.1 Problem Defnition

At present the LCD T.V are used to watch only channels and sometimes built in games. The user can able to access all features in smart TV as like a smart phone and tablet. Whereas the smart TV provides following features are, Watching TV channels, Playing offline and online games, watching videos from YouTube, Video conference through applications like Skype, Connecting to social networking websites like Face book etc. If we use individual devices for each and every application in different devices, power, time, cost becomes more. Power, cost, area, time are more important for developing end product and also very simple to use. In order to overcome this problem we are implementing most of the features in single device.

3 PROPOSED WORK

The proposed work in this paper is using LCD TV as smart TV by using BeagleBoard XM with android operating system as taget operating system and Linux as host operating

- *Author Sujith Kumar .S is currently pursuing masters degree program in VLSI desing and Embedded Systems in Visvesvaraya technological University, India, PH- 91 7259318108. E-mail: sujithkumar969@gmail.com*
- *Co-Author Dr. Ramachandra .A .C is currently head of the department in Electronics and Communication Engineering at Alpha College of Engineering, India, PH- 91 9448201096. E-mail: ramachandra.ace@gmail.com*

system. BeagleBoard XM,USB, mouse, USB keyboard, LCD TV with DVI port or monitor with HDMI/DVI, USB hub where USB of mouse and keyboard are connect, power adopter to supply 5 volts to the BeagleBoard XM, DVI to HDMI converting cable, there are 2 audio ports to connect the speakers or head phones. The android operating system and application software for our project are stored in the memory devices such as micro SD or HDD this is in turn connected to BeagleBoard XM as shown in below figure 1.

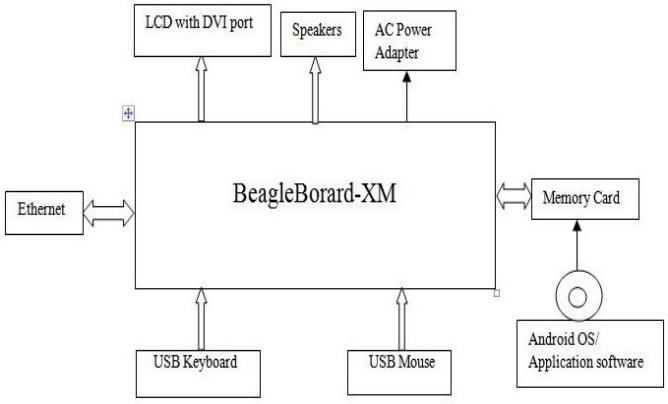

Fig. 1: Block diagram of smart TV system

Fig. 2: Overall hardware connections to BeagleBoard XM

To develop android application for Smart TV system,

Software requirements are,

- Eclipse.
- Java runtime environment (JRE).
- Android development tool (ADT).
- Android source code.Eclipse.
- Ubuntu 12.04 as a Host Operating System.
- Android Jelly Bean 4.2 as a Target Operating System.

Hardware requirements are,

- BeagleBoard XM (target machine).
- LCD with DVI port and audio speakers.
- Personal computer (host machine).
- Keyboard and Mouse.
- Stroage device (micro SD).

Board bring up: - It means giving a life to BeagleBoard XM which means android operating system is ported to target device after completion of following phases of works.

- To build boot loader: - It is welcome display which shows logo, name of company etc...
- To build kernel:- Kernel is a computer program that manages the inputs/outputs request from application software and translates to the data processing instruction for central processing unit and other computer devices.
- To build root file system:- The root file system is the file system that is contained on the same partition on which the root directory is located, and it is the file system on which all the other file systems are mounted (i.e., logically attached to the system) as the system is booted up (i.e., started up).

Booting sequence in Beagle Board XM,

ROM Code: Tries to find valid bootstrap image from various storage sources and load it into SRAM or RAM (RAM can be initialised by ROM code through a configuration header). Size limited to <64KB. No user interaction possible.

X-Loader: Runs from SRAM. Initialises the DRAM, NAND or MMC controller, and loads the secondary bootloader into RAM and starts it. This file is called MLO.

U-Boot: Runs from RAM. Initialises some other hardware devices (network, USB, etc...). Loads the kernel image from storage or network to RAM and starts it. Shell with commands provided. This file is called u-boot.bin.

Linux Kernel: Runs from RAM. Takes over the system completely (boot loaders no longer exists).

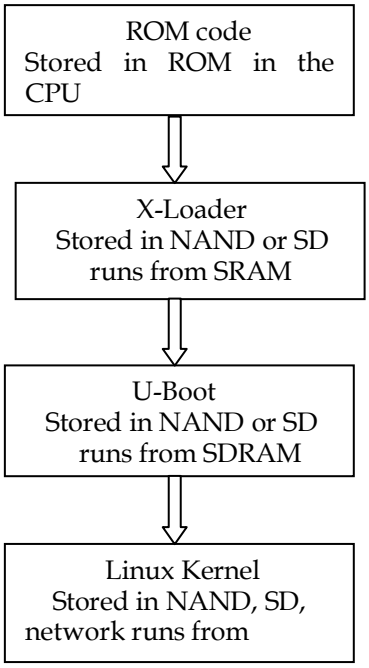

Fig. 3: Booting sequence in Beagle Board XM

Android Boot Animation,

The Android boot animation is contained within an uncompressed zip file called bootanimation.zip that can be found in the media folder of the system partition i.e. /system/media on the internal memory of the device. This single file contains all the information required to play the boot animation, and is loaded automatically when the device boots. The Android boot animation might appear to be in a video format.

Inside the Bootanimation.Zip File we have,

- A desc.txt file
- A part0 folder
- More part1, part2 etc. folders (May or may not be present)

The animation is played simply by displaying the images in a sequence, and the text file defines how they are to be played. In essence, first the PNG or jpg files in the part0 folder are displayed one after the other and afterwards, those in the part1 file if it exists are displayed, again one after the other, and so on. All of this is defined in the desc.txt file.

The folders:-

These contain PNG images named in numbers, starting from something like 0000.jpg or 00001.jpg and proceeding with increments of 1. There has to be at least one folder, and there is no known upper limit to the number of folders.

The desc.txt file:-

This file defines how the images in the folder(s) are displayed during the boot animation, in the following format:

Width Height Frame-rate

P Loop Pause Folder1

P Loop Pause Folder2

An example of a desc.txt file is:-

480 800 30

p 1 0 part0

p 0 0 part1

- In the first line, 480 and 800 define the width and height of the boot animation in pixels for this example. This must be the same as the screen resolution of your device for the boot animation to properly play in full screen. 30 is the frame rate in fps (frames per second) i.e. number of images to display per second.
- The second and third lines have a same format, start with p, which stands for a part of the animation and end in part0 or part1, which denotes the folder in which the images for that part are present.
- The number after 'p' defines how many times this part will loop (repeat playback) before switching to the next part (if present). Specifying 0 would make the part loop indefinitely till the phone has fully booted.
- The next number is for the pause, and is expressed in the number of frames, which can be translated into time by dividing it by the frame rate. A pause of 15 for example, would mean pausing for the time it takes 15 frames to play and since the frame rate is 30 frames per second, 15 frames would take half a second.
- The boot animation will play at a resolution of 480 by 800 pixels, at a frame rate of 30 fps, starting with the contents of part0 folder and after playing them in one loop, switching to contents of part1 folder and playing them continuously till the device fully boots.
- Now select everything inside the bootanimation folder and zip them into a new uncompressed zip archive using your favorite compression utility. Here is the method using 7-zip:

1. Select everything inside the bootanimation folder.
2. Right-click on any of the selected files/folders and from the 7-zip menu, select 'Add to archive'.
3. Use 'zip' as the archive format and 'Store' as the compression level, and click OK. This will create a file called bootanimation.zip in the same folder.

Rooted/Unrootedevices: - Enter these commands:

- adb pull /data/local/bootanimation.zip c:\
- adb push bootanimation.zip /data/local/

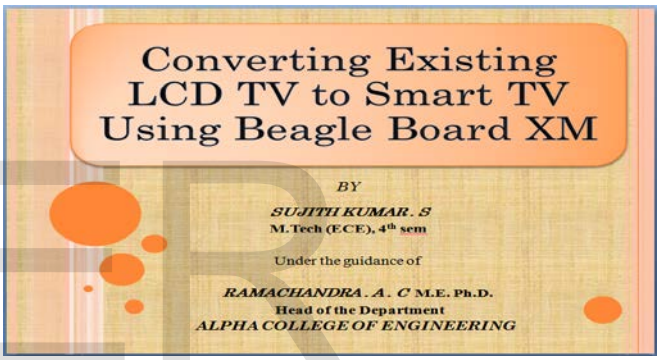

Fig. 4: Boot animation window in LCD display.

The above boot animation window is shown in display device. Similarly with same instruction procedure we can do it for any android devices or Beagle Board XM with LCD.

Commands to build the above stages are shown below.

- Tool chain setup:

To setup tool chain setup the tool chain path to point /home/pusstak/training/toolchain/arm-2009q1/bin. Execute the below command in command window.

$ export PATH=/home/Pusstak/Training/toolchain/arm-2009q1/bin: $PATH

- Build X-Loader (MLO file):

To create x-loader, image change directory to X-loader then execute the below commands. This will build the x-loader image as X-loader.bin.

$ cd /home/pusstak/training/toolchain/arm-2009q1/bin/x-loader

$ make CROSS_COMPILE= arm-none-linux-guneabi-distclean

$ make CROSS_COMPILE= arm-none-linux-guneabi-omap3beagle_config

$ make CROSS_COMPILE= arm-none-linux-guneabi

To create the MLO file which is used for booting from a MMC/SD card, sign the x-loader image using the signGP tool.

$ ./signGP ./x-load.bin

You will need to copy the signGP tool from the TI_Android_Utilities/signGP directory to the directory that contains the x-load.bin file. The signGP tool will create an .ift file; rename the x-load.bin.ift to MLO.

$ mv x-load.bin.ift MLO

- Boot Loader (U-boot.bin file):

Change directory to u-boot then execute following commands. Build will generate u-boot.bin file.

$ cd /home/pusstak/training/toolchain/arm-2009q1/bin/u-boot

$ make CROSS_COMPILE= arm-none-linux-guneabi-distclean

$ make ARCH=arm CROSS_COMPILE= arm-none-linux-guneabi- omap3beagle_config

$ make ARCH=arm CROSS_COMPILE= arm-none-linux-guneabi-distclean

- UImage (Kernel):

Execute following commands. This will generate uImage (kernel image) in kernel/arch/arm/boot folder.

$ cd /home/pusstak/training/toolchain/arm-2009q1/bin/kernel

$ make ARCH=arm CROSS_COMPILE= arm-none-linux-guneabi-distclean

$ make ARCH=arm CROSS_COMPILE= arm-none-linux-guneabi-omap3beagle_android_defconfig

$ make ARCH=arm CROSS_COMPILE= arm-none-linux-guneabi-distclean-uImage

- Root file system:

To build the root file system for Beagle Board execute this command.

$ make beagleboard-xm= beagleboard OMAPES=5.x –j

After porting android operating system into storage device (micro SD), the storage device has partioned as shown below figure 3.

MLO File	U-boot.bin	UImage Kernel	boot.scr Root file system

Fig. 5: Partitioning inside the storage device.

MLO:- Beagle board's firmware contains a first stage boot-loader called X-loader. X-loader can also be loaded from SD card in single file called MLO.

UImage:- It's a small kernel image with modified header for u-boot, enabling u-boot to load this kernel image.

Boot.scr:- It is a user defined image file that is read before loading UImage, allowing the user to supersede the loading of UImage, preventing the user from recompiling UImage.

U-boot.bin:- Which passes the control to the Linux system. It retriew kernel from the flash. U-boot boots the Linux kernel.

Finally we develop android application for watching online video channels and audio player.

3.1 Developing Android Applications for Watching Online Video Channel and Audio Player

The lifecycle of the Audio Player is not always straight forward and the order in which it takes on various states is best explained diagrammatically. In Audio player we have used java language to develop both Audio player and online video player. Using eclipse the java code has written. Below programming flow is for simple Audio player. Once the device is on after boot animation window we can see list of android programs in which the audio player is selected thus program is initialised. Then what all the mp3 files we have selected that is pushed to beagle board for preparing to play those files. After preparing it started to play those files that can be listen through head phones or speakers. While playing, in control panel if pause button is enabled then the program execution is delayed until again start button is pressed. In case if user enabled the stop button, then program control goes to prepare for next mp3 file to play. In other case, if mp3 song is allowed to complete with full duration, once the song completes the audio player is stopped. A new mp3 is prepared to play and continues to play until user press play button. The programming flow for audio player is shown in figure 6.

Similarly, for online video channel Ethernet cable is connected to beagle board. Then driver for that is initialised when device is switched on. Then programming flow for watching online video channel is same as audio player but URL of the selected channel is considered as playing list rest of the controllings are same. Only for watching videos but not for downloading videos while playing. When application is selected the list of sports channel is listed. Then particular URL is loaded to media player. The programming flow for watching online video channels is in figure 7.

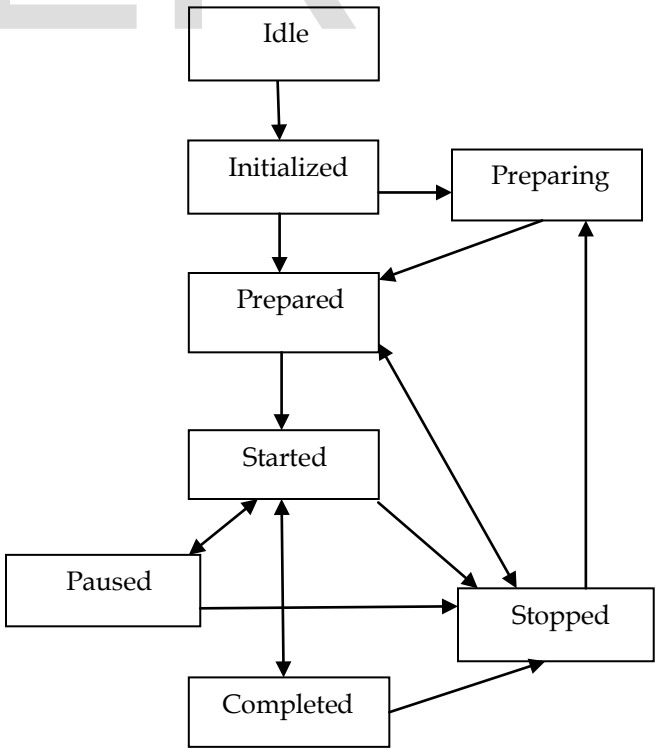

Fig. 6: Programming flow for Audio Player.

IJSER

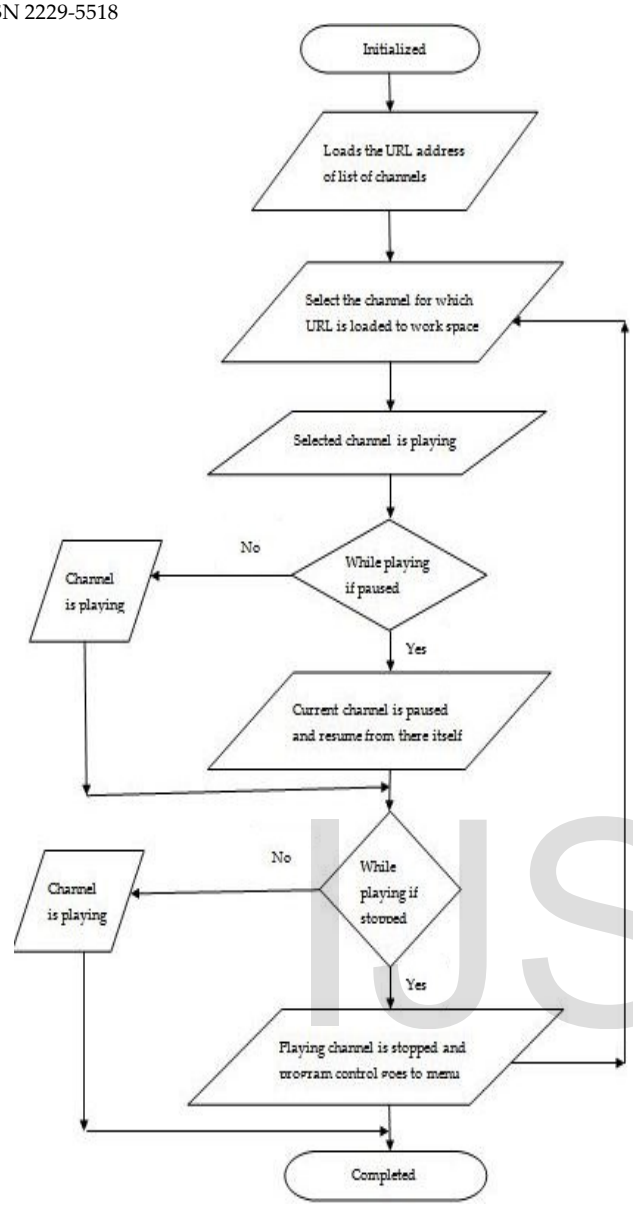

Fig. 7: Programming flow for Watching Online Video Channels.

4 RESULTS

Fig. 8: Snapshot of Audio Player Application.

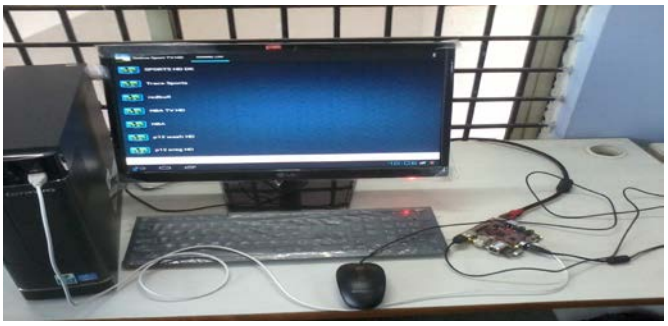

Fig. 9: Snapshot of Watching Online Video Channels Application.

Fig. 10: Snapshot of an online sport channel.

After porting system software and application software such as Watching Online Video Channel Application and Audio Player Application on BeagleBoard XM, when development board is switched on both system software and application software are loaded from memory card and display acceptable results. Figure 8 shows Audio player Application in LCD display. Figure 9 shows Watching Online Video Channels Application. Figure10 shows online sport channel after selecting particular channel in list. It can also enhance it for video player too. Figure 8 shows the Audio Player Application in LCD display with controlling buttons such as Play, Pause, Stop, Shuffle, Rewind and Forward. This java code can be contributed as open source code to the society.

5 CONCLUSION

The proposed system was built and was executed on the BeagleBoard XM development board successfully with expected results. We built android operating system and android application software for BeagleBoard-xM. We can also improve it for touch screen devices and also include text to speech converter, Video conversation, MMS, Google talk, SMS etc… Since we have used ARM cortex A8 whose operating frequency is 1GHz, it will be more convenient for user to access all features in single screen with considerable speed instead of using separate devices. We can also remove unwanted standard interfaces to optimise the area of development board at end product by the manufacturer according to application needs. In this paper it can enhance the features to develop android application s/w for touch screen devices, Instead of using wired USB mouse and key board.

IJSER

ACKNOWLEDGMENT

I am very much thankful to Dr. RAMACHANDRA .A .C, HOD, Electronics and Communication Engineering. I also express sincere thanks to all faculty members of Department for the valuable advice and assistance. This project is successfully carried out under the guidance of VISHWAKIRAN who is working in Pushkala Technologies Pvt Ltd. I also thank all my family members and friends for their kind cooperation.

REFERENCES

[1] Kyle Merrifield Mew. "Android 3.0 Application Development Cookbook". July 2011. 5-33, 191-209.
[2] Thomas Thurman. "MeeGO 1.0 Mobile Application Development Coockbook". 2012.
[3] Nirav Mehta. "Mobile Web Development". 2010.
[4] "BeagleBoard-xM System Reference Manual". Revision C.1.0. April 4, 2010. 22-31, 33-47, 116-141.
[5] Alan Cox. Video4 Linux Programming [EB/OL]. (2000)[2011-01-20].alan@redhet.com.
[6] Vivek G Gite. "Linux Shell Scripting Tutorial". Ver.1.0. Aug 2001. 2-17.
[7] Herbert Schildt. "Java the Complete Reference". Seventh Edition.
[8] Liggesmeyer P., Trapp M. Trends in Embedded Software Engineering [J]. IEEE Software, 2009, 26(3):19-25.
[9] Information on: http://android-developers.blogspot.in/
[10] Information on: http://source.android.com/source/

IJSER