International Journal of Computer Applications Technology and Research
Volume 3– Issue 5, 299 - 303, 2014, ISSN: 2319–8656

Remote Control based Audio-Video Content Filter Application for Philips 2K10 TV with JointSPACE Architecture

Mohan Kumar J
School of Information Sciences
Manipal University, Manipal.
India

Sandhesh Gowda
School of Information Sciences
Manipal University, Manipal.
India

Rahul Devi Reddy
School of Information Sciences
Manipal University, Manipal.
India

Harishchandra Hebbar
School of Information Sciences
Manipal University, Manipal.
India

Sundaresan C
School of Information Sciences
Manipal University, Manipal.
India

Chaitanya C.V.S
School of Information Sciences
Manipal University, Manipal.
India

Abstract: JointSPACE is an Open Source project that allows every user/supplier to develop applications for Philips TV displays. JointSPACE is based on the SPACE architecture which was developed by Philips to ease internal development. At a certain point of time, Philips decided to open its architecture to allow everyone developing code for the TV target. In this paper we propose and implement a Remote control application for Audio-Video Content Filter Application. One of the major issues with people watching the Television shows, as a family with children, especially in Asian context, there may be some inappropriate visuals with audio may appear. These contents affect the children mental health. So a remote application is created to mask the video and mute the audio for the required time, by the user. Even the change of channel may have the similar type of content. So this application is not providing the complete solution for this problem, but helps in certain scenarios.

Keywords: Philips JointSPACE TV, Directfb, Voodoo, Mobile Applications

1. INTRODUCTION

Television (TV) is an essential entertainment media in common man's life. Presently smartTv started emerging and in the next decade, the user space will be more. These smart TV run a particular software platform for running the applications. One such software platform is JointSPACE from Philips. JointSPACE is an Open Source project that will allow every user/supplier to develop applications for Philips TV displays. JointSPACE is based on the SPACE architecture which was developed by Philips to ease internal development. At a certain point of time, Philips decided to open its architecture to allow everyone developing code for the TV target. [1][2].

2. JointSPACE

JointSPACE facilitates mainly 2 aspects:

i. Integration of applications made by suppliers.
ii. Integration of applications made by customers. JointSPACE addresses this by opening and extending the current TV architecture. [1]

Some of the features of Jointspace are:

JointSPACE proposes a single platform to develop applications. The platform may be any Linux PC or device capable of running Linux/DirectFB technologies. JointSPACE publishes the essential TV APIs used in the SPACE architecture. JointSPACE provides portable prototyping software that includes and illustrate the essential of the SPACE architecture. [1]

JointSPACE extends the TV architecture to allow:

1. Executing TV applications on a remote system, rendering and being controlled on the TV
2. Executing application on a remote system, controlling the TV APIs remotely [1]

JointSPACE continuously provide new technologies/libraries to ease and improve the development of new applications. JointSPACE extends the TV API to allow controlling more TV functionalities. As JointSPACE is based on DirectFB technologies; following DirectFB packages are used

- DirectFB 1.4
- SaWMan 1.4
- FusionDale 0.8.1.

The JointSPACE exposes 3 core APIs apart from these packages for developers, as shown in Figure 1.

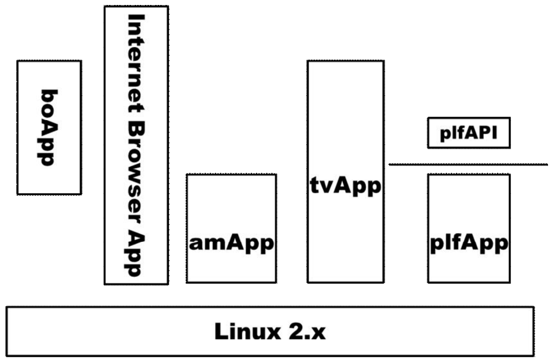


Figure 1. JointSPACE Architecture

plfApi is the API for control various platform devices, like Audio/video setup and playback, demux, tuner control, Picture and sound properties (Brightness, contrast, volume etc.)

plfApi is the key API which needs to be explicitly managed because multiple application will need to access it. To reduce complexity of resource management plfApi is divided into a number of resource groups

- Source
- Front End
- Audio featuring
- Video Featuring
- Connectivity
- Mute

2.1 Resource Management – plfApi

Management of plfApi is done by application manager (amApp).All applications needing to control any of plfApi interfaces must create these special windows of directfb and use the window Id as an identifier to call plfApi.Audio node window for calling plfApi functions related to audio control. Border window for calling any other plfApi functions.

All plfApi calls are routed through a resource gating layer within the plfApp which only allows calls with the owned window ID. Application request resource groups of plfApi that they need to call towards amApp. AmApp ensures that the requesting application's window ID is set into resource gate of plf.

2.2 Application Manager API

Application Manager API (amApi) is the API used for communication between clients and application manager.Some of the important functions available are:

- Power related functions: PreparePowerState, RequestPowerState, ConfirmPowerState.
- Platform Resources: RequestPlfApi, PlfApiDenied.
- Key grabbing: RequestKey.
- Activities: Requesting various activities.
- Focus management: RequestOverlay, SetFocus
- Multiview: AddToLayout, MoveToLayout.
- System event management: DisableEvents.

2.3 Connection Management

The connection between the remote device and the television is established by a set of processes. Firstly the application manager api, amapp, starts the communication, after the device discovery. Then the remote device calls the directfbinit function, so that the jsapp master can fork another process jsapp1. Also the process will be added to the application manager. Then the remote device initiates directfbcreate and createwindow functions. Then a window will be created inside the application manager for the particular process. Then the remote device requests requestfocus, through jsapp1 so that the application manager focuses to the amapp.

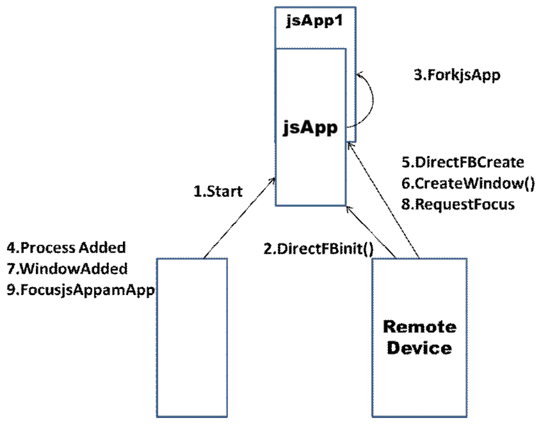


Figure 2. Connection Management

2.4 DirectFB – Direct Frame Buffer

A thin library that provides hardware graphics acceleration, input device handling and abstraction, integrated windowing system with support for translucent windows and multiple display layers, not only on top of the Linux Frame buffer Device. It is a complete hardware abstraction layer with software fallbacks for every graphics operation that is not supported by the underlying hardware. DirectFB adds graphical power to embedded systems and sets a new standard for graphics under Linux.[5]

2.5 JointSPACE Simulator

The jointSPACE simulator allows to experiment with the SPACE architecture on a Linux PC[3][4]. It is splitted into various packages organised into sub-directories.

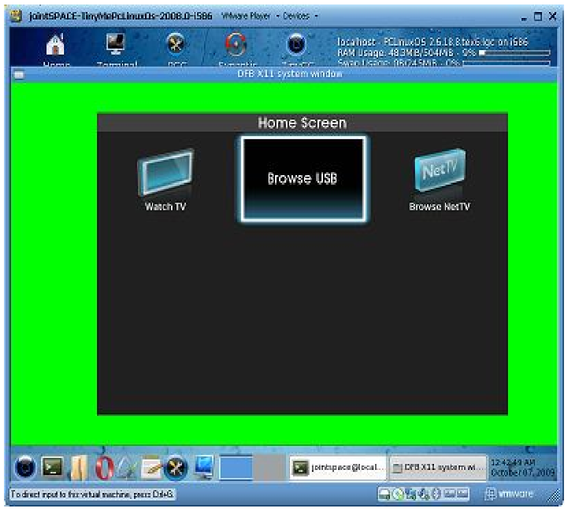


Figure 3. JointSPACE Simulator[4]

The package currently includes: DirectFB (in the form of installation scripts and patches, as DirectFB itself is hosted at directfb.org), the basic SPACE applications (amapp, plfapp, homeapp, tvapp other examples in the form of source code), the basic libraries required by SPACE (presenting amApi and plfApi to all applications, across processes' boundary using FusionDale IPC; in the form of dynamic libraries and header files), the platform porting glue header files (papi), mainly required by HW suppliers to integrate their platform in the architecture (the default plfapp delivered is based on a papi implementation delivered in source code but mainly consisting in stubs that could be mapped on standard Linux device drivers as v4l or other higher level packages as SDL or mplayer.)

These packages represent the minimum required by the simulator. New packages will be added during the life time of this project (new libraries, new APIs etc) to ease the development of future applications.

3. REMOTE APPLICATIONS

Remote applications are applications running on external devices, making use of TV capabilities to render GFX and media. External devices can be any computing device including iPod/iPhone, game consoles, PDA, lightweight PC, PC servers, MAC. Remote applications are using the IP network (wired or wireless) to communicate with the TV. It can be used to extend TV functionality with customizable features, integrated together with "standard" TV applications. Remote applications can also be controlled with the TV remote[6]. They are making use of DirectFB/VooDoo technologies and follow (joint)SPACE architecture rules.

Two kind of remote applications are currently supported:

1. Applications to control the TV remotely (inject events like keys over the network)
2. Applications making use of the TV resources to render GFX and media content.[6]

3.1 Block Diagram

A remote based content filter application has been created for the television, which can be used at the any remote device. Here the users have to press a key in a remote device, which can be a laptop or a mobile phone, pressing the key will block the content and wait for the another key press for releasing the content. For the implementation a laptop is considered as a remote device. This can be extended to mobile phones. Instead of the Television, the JointSPACE simulator is used [2], which is an open source simulator for the Philips JointSPACE Television, to test the applications.

The filter should have an application running on the Television and also another application running on the remote device. Whenever the user send a signal from the remote device, this signal will be received TV, through the Wi-Fi Connection. Once the signal is received a screen is created on top of the visual. Also the audio is muted. The simple block diagram of the implementation is shown in Figure 3.

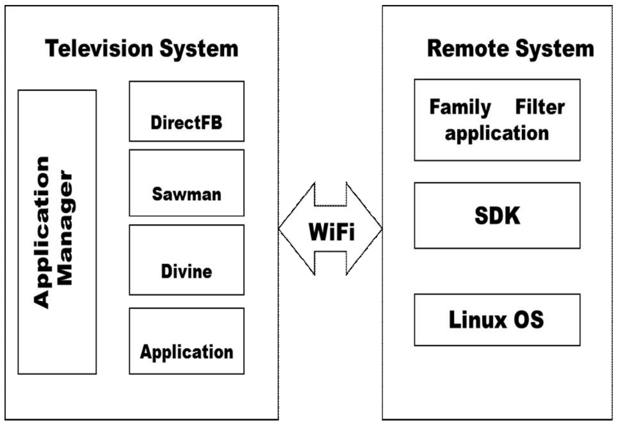


Figure 4 Block Diagram

3.2 Scenario and Scope of the Implementation

For the implementation scenario, considered certain major issues in the human life with regard to the Television. Present scenarios families face difficulty in control of children watching Television. The content broadcasted in television are sometimes not apt for children. So it is necessary to provide Parental Guidance for child. One way of avoiding is to mask the video and mute the audio, for some time. Even though change of channel can be one of the solutions, we can’t predict the content broadcasted on changed channel. So this application can be used for masking video and mute audio for the required time until the user decides to unmask.

Scope of this project extends from each and every family, who is having a television, to very large display halls, where people watch different programs on television especially as a family.

3.3 Implementation

The implementation of this project was a step by step procedure. An application is created such that a signal is send from the remote device when the users press a button. A RC code is send to the TV. On the TV this code is received and accordingly a window is drawn on the video running on the TV. Also the audio is muted. The project is implemented using C programming and linux commands. Filter.c file is written when a header file Control.h is invoked. The Control.h has two functions defined, namely dfb_start() and dfb_stop(). These functions are used to input the key for masking and unmasking the video along with muting and unmuting the audio. Two threads are created, one is to show the image continuously on TV until stop event is given from the remote device. Second thread is to wait for event to stop given from the remote device ie to unmask the screen layer and unmuting the audio. The following RC code for Mute and Demute in the program which is #defined in the filter.c program.

```
#define NOTRCSOURCEMASK        0x20

#define keySourceRc6           3

#define rc6S0AvMute            163

#define rc6S0DeMute            13
```

The flowchart of the implementation is shown in Figure 5. Figure 6 shows the user input for blocking/masking the TV content. For the prototype the program waits for the user to give an input as 1.Figure 7 shows the normal TV channel watching. Once the key is pressed a new layer is created on the currently watched TV screen blocking the content. Audio is also muted. This is shown in Figure 8.

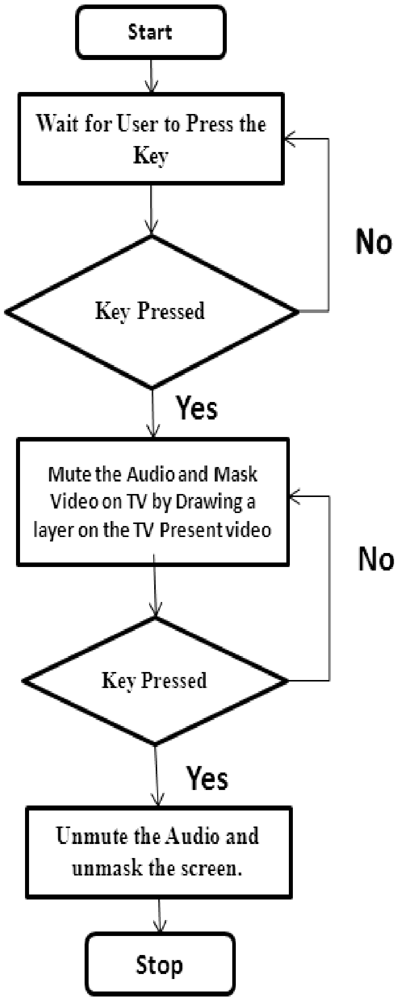
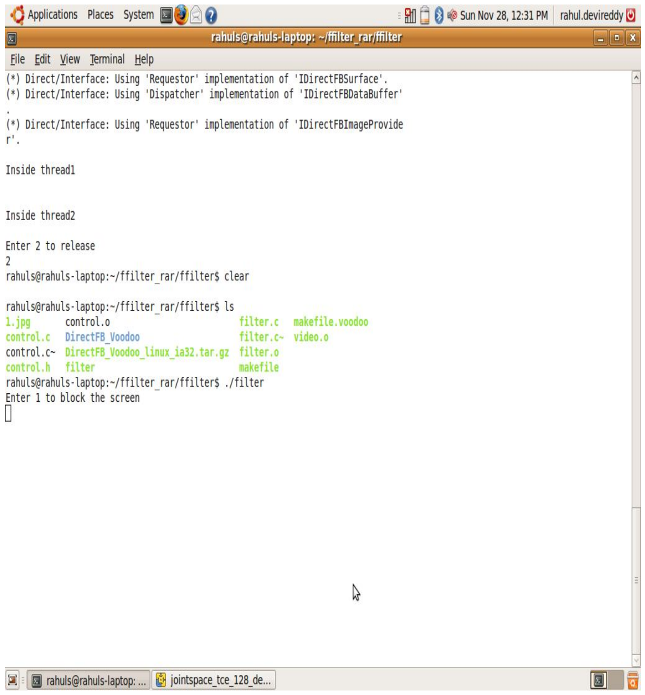

Start
Wait for User to Press the Key
Key Pressed
No
Yes
Mute the Audio and Mask Video on TV by Drawing a layer on the TV Present video
Key Pressed
No
Yes
Unmute the Audio and unmask the screen.
Stop

Figure 5 Flowchart of the Implementation


Applications Places System
Sun Nov 28, 12:31 PM rahul.devireddy
rahuls@rahuls-laptop: ~/ffilter_rar/ffilter
File Edit View Terminal Help
rahuls@rahuls-laptop: ...
jointspace_tce_128_de...

```
(*) Direct/Interface: Using 'Requestor' implementation of 'IDirectFBSurface'.
(*) Direct/Interface: Using 'Dispatcher' implementation of 'IDirectFBDataBuffer'
.
(*) Direct/Interface: Using 'Requestor' implementation of 'IDirectFBImageProvide
r'.

Inside thread1


Inside thread2

Enter 2 to release
2
rahuls@rahuls-laptop:~/ffilter_rar/ffilter$ clear

rahuls@rahuls-laptop:~/ffilter_rar/ffilter$ ls
1.jpg      control.o                       filter.c   makefile.voodoo
control.c  DirectFB_Voodoo                 filter.c~  video.o
control.c~ DirectFB_Voodoo_linux_ia32.tar.gz filter.o
control.h  filter                          makefile
rahuls@rahuls-laptop:~/ffilter_rar/ffilter$ ./filter
Enter 1 to block the screen
```

Figure 6 Running family filter application in Remote Device (laptop)

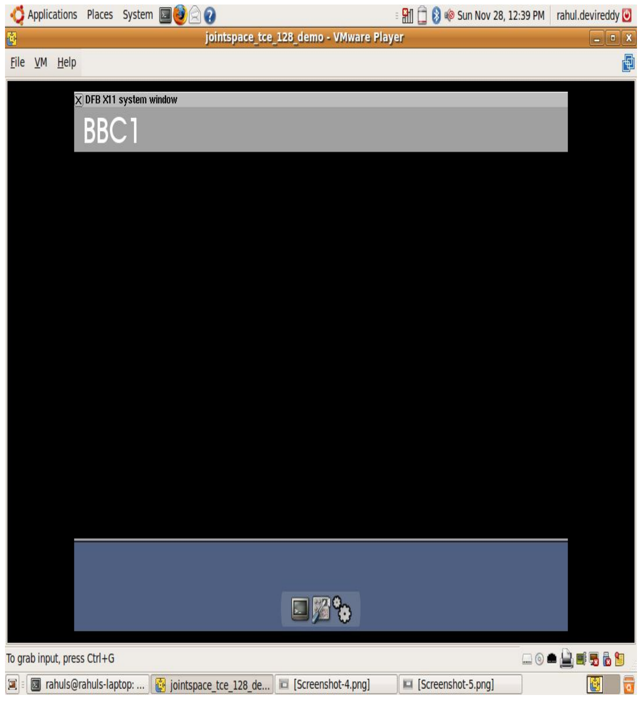

Applications Places System
Sun Nov 28, 12:39 PM rahul.devireddy
jointspace_tce_128_demo - VMware Player
File VM Help
DFB X11 system window
BBC1
To grab input, press Ctrl+G
rahuls@rahuls-laptop: ...
jointspace_tce_128_de...
[Screenshot-4.png]
[Screenshot-5.png]

Figure 7 Watching a channel in the TV

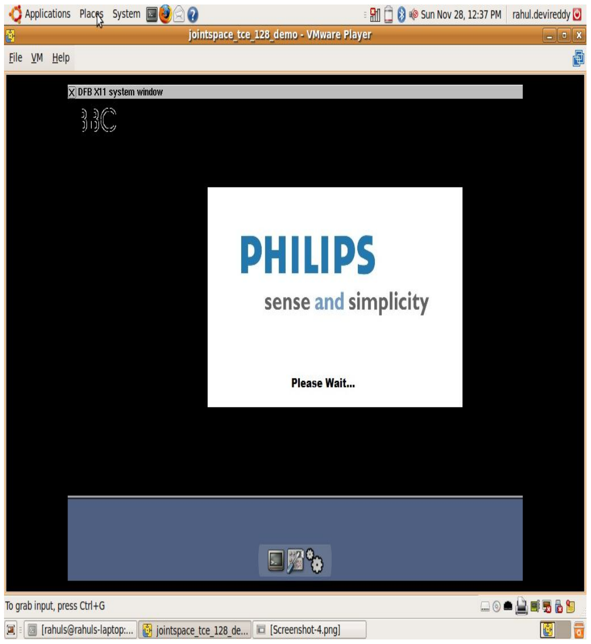
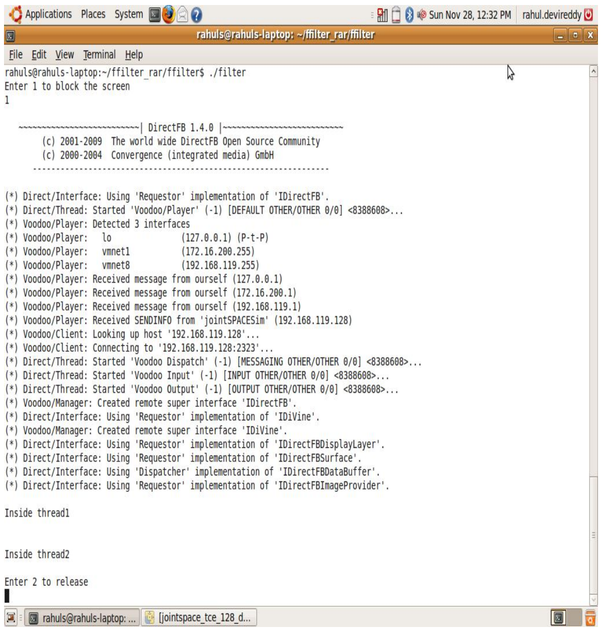

Applications Places System
Sun Nov 28, 12:37 PM rahul.devireddy
jointspace_tce_128_demo - VMware Player
File VM Help
DFB X11 system window
PHILIPS
sense and simplicity
Please Wait...
To grab input, press Ctrl+G
[rahuls@rahuls-laptop:... jointspace_tce_128_de... [Screenshot-4.png]

Figure 8 Blocks the running video for a while


Applications Places System
Sun Nov 28, 12:32 PM rahul.devireddy
rahuls@rahuls-laptop: ~/ffilter_rar/ffilter
File Edit View Terminal Help
rahuls@rahuls-laptop: ... [jointspace_tce_128_d...

```
rahuls@rahuls-laptop:~/ffilter_rar/ffilter$ ./filter
Enter 1 to block the screen
1

   ~~~~~~~~~~~~~~~~~~~~~~~~~~| DirectFB 1.4.0 |~~~~~~~~~~~~~~~~~~~~~~~~~~
        (c) 2001-2009  The world wide DirectFB Open Source Community
        (c) 2000-2004  Convergence (integrated media) GmbH
      ----------------------------------------------------------------

(*) Direct/Interface: Using 'Requestor' implementation of 'IDirectFB'.
(*) Direct/Thread: Started 'Voodoo/Player' (-1) [DEFAULT OTHER/OTHER 0/0] <8388608>...
(*) Voodoo/Player: Detected 3 interfaces
(*) Voodoo/Player:   lo             (127.0.0.1) (P-t-P)
(*) Voodoo/Player:   vmnet1         (172.16.200.255)
(*) Voodoo/Player:   vmnet8         (192.168.119.255)
(*) Voodoo/Player: Received message from ourself (127.0.0.1)
(*) Voodoo/Player: Received message from ourself (172.16.200.1)
(*) Voodoo/Player: Received message from ourself (192.168.119.1)
(*) Voodoo/Player: Received SENDINFO from 'jointSPACESim' (192.168.119.128)
(*) Voodoo/Client: Looking up host '192.168.119.128'...
(*) Voodoo/Client: Connecting to '192.168.119.128:2323'...
(*) Direct/Thread: Started 'Voodoo Dispatch' (-1) [MESSAGING OTHER/OTHER 0/0] <8388608>...
(*) Direct/Thread: Started 'Voodoo Input' (-1) [INPUT OTHER/OTHER 0/0] <8388608>...
(*) Direct/Thread: Started 'Voodoo Output' (-1) [OUTPUT OTHER/OTHER 0/0] <8388608>...
(*) Voodoo/Manager: Created remote super interface 'IDirectFB'.
(*) Direct/Interface: Using 'Requestor' implementation of 'IDiVine'.
(*) Voodoo/Manager: Created remote super interface 'IDiVine'.
(*) Direct/Interface: Using 'Requestor' implementation of 'IDirectFBDisplayLayer'.
(*) Direct/Interface: Using 'Requestor' implementation of 'IDirectFBSurface'.
(*) Direct/Interface: Using 'Dispatcher' implementation of 'IDirectFBDataBuffer'.
(*) Direct/Interface: Using 'Requestor' implementation of 'IDirectFBImageProvider'.

Inside thread1


Inside thread2

Enter 2 to release
█
```

Figure 9 Sending a signal to get back to normal audio and video

To bring back the audio and video another key is pressed. Here it is number 2. This is shown in Figure 9.

4. CONCLUSION AND SCOPE

A remote based Audio-Video Content Filter Application for Philips Joint SPACE is developed and checked under the development environment. The application has the capability to block and mute the inappropriate contents when watch with the family especially with children. The application is not developed to eradicate the complete impact of problems due to watching TV by children. But it can be used in some scenarios. This application is tested on the laptop as a remote device and JointSPACE simulator. The same can be developed on an android mobile as remote control application is already available for Philips TV with JointSPACE architecture. This idea can be incorporated with the android remote control application. Similarly can be implemented for IPHONE and Windows based mobile.

5. ACKNOWLEDGEMENT

We thank Dr. A Narendranath Udupa, Philips Research, Bangalore and Mr. Manjunatha Maiya, Sr. Project Manager, MU - BoP, Philips Pvt.Ltd for providing us the JointSPACE platform training and donating a TV for executing projects.

6. REFERENCES

[1].http://foundation.webinos.org/deliverabled026target-platform-requirements-and-ipr/26-nettv-fraunhofer/ (March 2014)

[2]. http://jointspace.sourceforge.net/ (March 2014)

[3]. http://sourceforge.net/projects/jointspace/ (March 2014)

[4].http://sourceforge.net/apps/mediawiki/jointspace/index.php?title=JointSPACE_Simulator (March 2014)

[5]. http://directfb.org/ (March 2014)

[6].” System architecture for virtual world interfacing with TV platform” - Virtual world interfacing with TV platforms Article, http://wg11.sc29.org/mpeg-v/?page_id=298 (March 2014)