Design and implementation of an Embedded Linux based application for realtime Digital Video Recording subsystem

Fema Merin Jacob*
Sajan P. Philip*
Dinesh T#
*Assistant Professor, Department of ECE
#UG Scholar
Bannari Amman Institute of Technology, India 638401

***Abstract*—This paper presents the design and implementation of a real time embedded application for recording and storing the video output displayed on the X11 monitor based on embedded Linux using Qt and C++. The hardware platform, Software Platform and the development environment used in this work are IBM's PowerPC 7410, Embedded Linux and Qt and Qt/Embedded respectively. The embedded operating system is MontaVista Linux with X11 built as an abstract layer on kernel providing graphics capability to the system. VGA out of the Argus graphics PMC card is connected to the CRT monitor. Time varying data on monitor is captured and open source *FFmpeg* library is employed to encode captured window frames to mpeg video which is then stored to flash PMC card. The implementation of the entire design environment is based on the X Window System and the Linux operating system and can thus be used on an increasing number of low-cost workstations. Recording of results to a video file enables detailed analysis of the results anywhere/anytime. As part of this work, a study of X window system programming, *FFmpeg* library routines and Qt programming have been undertaken. The experimental testing and results indicated that this system is working stably and reliably. Compatibility, reliability, portability and ease of use of the aforesaid generic application give it an upper hand over the other existing methods.**

Keywords: Qt/Embedded, X11, Argus PMC, Flash PMC, FFmpeg library, Xlib library

I. INTRODUCTION

With the rapid development of embedded chip technology, embedded system is becoming high-powered, multifunctional and all-purpose, which has had a significant impact on changing people's lifestyle and improving quality of life [1]. Many of the applications running on embedded platforms generate time varying visualizations on the display device. On-line and post trial analysis of the data needs to be performed allowing the visualisation of the measurements and comparison to predicted or modelled results [2]. Following the trials, a period of intense data analysis occurs where the measured data from the disparate sources may be modified and compared. When the time varying results are displayed on the monitor connected to embedded board, it needs to be recorded as a video for the required amount of time so that further analysis is possible.

Existing screen capture and recording utilities like Xvidcap available for X window system cannot be ported directly to embedded systems. This paper presents an embedded application to record the time varying data displayed on the X server.

The paper is structured as follows. In section II, the existing methods are discussed. Section III explains the hardware platform while Section IV describes the system software architecture. Section V presents the design and implementation of the Software using the developing platform. Finally, in section VI and VII, results and conclusion are drawn.

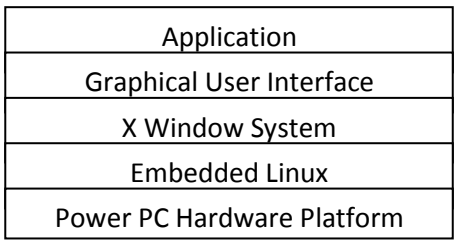

Figure 1. System Architecture

II. EXISTING SYSTEM

Digital Video Recorder is used presently to record the displayed data from an Embedded System. Analog output from the graphics PMC card available in the frame buffer of graphics card is taken, converted to digital data and provided as input to the DVR. Compression and storage is done by DVR which is a black box type implementation. Most of the inbuilt subsystems in DVR have complex interfaces. System is expensive.[10] It also suffers difficulty of maintenance. Analysis is affected because of complexity. The capabilities of an embedded systems hardware parts like graphic cards, flash memory, and software parts like video codec's etc are not explored in the conventional methods. This makes the system more costly and less reliable. There are applications in which portability of the system is important, and the use of the DVR system results in the bulkier implementations. There are chances of security threats when the DVR connected to a network.

III. PROPOSED SYSTEM

In this work a novel method is explained which in cooperate the recording and storing in the embedded system itself. The basic architecture of the design is shown

ISSN: 2278 – 1323

in figure 1. The various components of the proposed system are explained below:

1. Embedded Platform

An embedded platform can have X window system embedded in it for graphic capabilities. The X Window System (X11) is the most commonly used windowing system on UNIX and UNIX-like boxes. X is built as an additional abstraction layer on top of the operating system kernel. It is small and efficient, it runs on a wide range of hardware, it is network-transparent, and it is well documented. It is based on a single client/server relationship where the display server is the program that controls and draws output to the display monitors, tracks client input and updates windows accordingly, while clients are application programs that perform specific tasks. The X Server supports VGA and non-VGA graphic cards, has support for depths 1, 2, 4, 8, 16, and 32, and has built-in support for rendering. X window system architecture is shown in Figure 2.

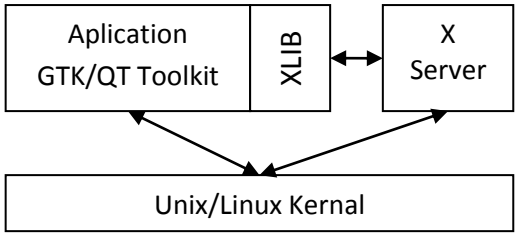

Figure 2. X Windows System Architecture

2. Embedded Language

The development language for the application is Qt [3, 4]. Qt is one of the most used programming languages to develop GUI applications in embedded development platform. Qt/Embedded is a C++ framework for GUI and application development for embedded devices. It runs on a variety of processors, usually with Embedded Linux. Xlib**,** an X Window System protocol client library written in the C programming language functions are used in the coding functions. The X11 library extensions, XComposite extension, XDamage extension and XShared Memory extension library functions are also used.

3. Hardware Platform

The hardware platform is DP-VME-0504 PPC 7410 board. It is the latest powerpc processor (7410 with AltiVec) @ 400 MHz with an inbuilt 2MB L2 cache. The board has up to 32 MB boot flash, 32 MB user flash and up to 128 MB SDRAM with ECC. The board has MontaVista Linux as the embedded operating system and X11 window system built as an abstract layer on embedded linux kernel to provide graphic capabilities for the system. Argus graphics PMC card is attached to the PMC slot on the PPC board. VGA out of the card is connected to the monitor display .Argus graphics PMC card supports an analog (RGB) resolution up to 1920 x 1200.

IV. SYSTEM SOFTWARE ARCHITECTURE

The application is written in C++ using the Qt toolkit as a platform abstraction layer, xlib to interface with the X11 server, the X11 XComposite extension to read the image data from X11 captured window and the libav libraries to encode the data into a video file that can be read with common video players. Application architecture framework is shown in Figure (3).

1. Graphical User Interface

A graphical user interface is designed which provides push buttons for the RECORD and STOP. When the RECORD button is pressed, a message box appears asking the user to click on the window for capture. Thus, the user can select the desired window for screenshot capture and subsequent coding to MPEG video. The duration for capture can be controlled by clicking on the STOP button. It exits the application. The QPushButton widget is used to provide the command button.

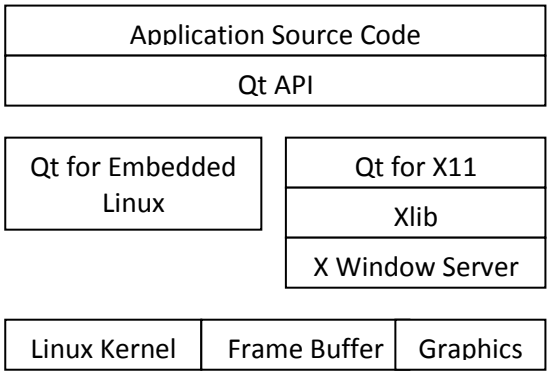

Figure 3. Application Architecture Framework

The push button emits the signal clicked() when it is activated by the mouse. It is required to connect to this signal to perform the button's action. The record() function is connected to the RECORD button's clicked() signal and close() function to the STOP button's clicked() signal.

A. record() function

In record() function, two independent threads are created:doCapture and doEncode. Thus the capture process and hence tracking changes to the window and encoding functionalities are separated by the use of threads. To create a thread, create an object of subclass QThread and reimplement its run() function. For starting the thread, create an object of the thread object and call QThread::start(). QReadWriteLock is used to achieve synchronization between the threads. The QReadWriteLock class provides read-write locking. This type of lock is useful if you want to allow multiple threads to have simultaneous read-only access, but as soon as one thread wants to write to the resource, all other threads must be blocked until the writing is complete.

B. doCapture thread

When the user clicks on desired window for capture,the capture process starts. By actively grabbing control of the mouse pointer and using the mouse Button Press event, the user selected window can be realized. The selected window attributes are stored in

ISSN: 2278 – 1323

XWindowAttributes structure. Using the composite extension of X, the window is redirected to off screen storage and pixmap id is obtained as a reference to the storage space. A shared memory segment is created using shared memory extension of X by the application and server is attached to the segment. Thus, exchange of image data between client and server is done.

During the capture time, changes to the pixmap of the window are to be tracked and image data be updated accordingly. This is done by creating a damage handle for the window which reports a damage notify event when any change in pixmap occurs. Application pulls the events out of the queue one at a time by calling XNextEvent().Each X event has a unique number that identifies that event, with the numbers for the events in the core protocol starting with the base number zero. When the damage notify event is received, application reads the updated image data into the shared memory segment to which the server is attached.

The algorithm used for Window Capture Thread is illustrated below:

Algorithm Window_Capture_Thread(attr,mem_addr)
This function captures the desired window
Pre attr represents desired window attributes
mem_addr represents the memory segment
Post the desired window is saved into a memory segment
1. Get window attributes
2. Direct the attributes to screen storage
3.Creatre a Shared Memory Segment
4. Read the image to the server
5. If (Damage is notified)
1. Update the memory segment
6. End if
End Window_Capture_Thread

C. doEncode thread

Libav libraries which are a part of the video codec library: Ffmpeg are used to encode window frames online to video. To compress the captured window frames into mpeg, libav library routines are used. The window attributes stored in the XWindowAttributes structure are provided as input to libav library routines. Various properties of mpeg video file like frame rate, gop size etc may be specified. The time needed for compression and the size of the final mpeg file depends on these factors also other than the extent of variation in the picture being encoded.

Depending upon the system requirements, the required MPEG codec can be selected. MPEG 1 and MPEG 2 codec uses a frame rate of 25-30 frames per second. MPEG 4 codec makes use of a frame rate of 14 fps. The codec can be selected by providing the appropriate codec id as input to ffmpeg library routines.

Algorithm doEncode thread(av_video)
This function encodes the window into a video
Pre av_video represents the output video
Post the encoded video is stored in av_video
1. Register the video codecs
2. Identify the format
3. Add the video stream
4. Set the parameters
5. Open av_video
6. Write video header
7. Loop (End of file)
1. construct the encode video
2.initialize the video packet
3.write the video frame to av_video
4. write video trailer to av_video
8. End Loop
End doEncode thread

V. IMPLEMENTATION

Qt project is cross compiled to produce the executable for target board. The Qt Everywhere 4.6.3 archive is used for the cross compilation of the application developed on the host system. Qt Everywhere 4.6.3 for Embedded Linux is available for download as an open source. We have to build the Qt Embedded 4.6.3 libraries for Embedded Linux by editing the appropriate qmake.conf file and setting path to the cross compiler powerpc-linux-g++ which must be present on the host system.

The below libraries must be copied from qt-everywhereopensource-src-4.6.3/lib/ on host to /usr/lib on target.

- *libQtCore.so.4*
- *libQtGui.so.4*
- *libQtNetwork.so.4*

This can be done by copying the above libraries to /usr/lib of the target root file system.
Target application is built by means of Qt's qmake tool. We should run the following sequence:

myProjDir$ qmake -project
myProjDir$ qmake
myProjDir$ make

However, this uses the default qmake (for host) and produces an executable that is unsuitable for the target (PPC) architecture. The correct qmake is located in qt-everywhereopensource-src-4.6.3/bin/, so to create a target executable we should run the following sequence:

myProjDir$~qt-everywhere-opensource-src-4.6.3/bin/qmake -project
myProjDir$~qt-everywhere-opensource-src-4.6.3/bin/qmake
myProjDir$ make

The application can now be transferred to and executed on target. Also other dependencies for the application like libavcodec.so, libavformat.so, libswscale.so,

ISSN: 2278 – 1323

libXComposite.so and libXDamage.so also must be copied to /usr/lib on target. The root file system can then be mounted on to the PPC target board via target NFS root mount.

To store the mpeg video file in the flash PMC card, mount the device driver of the card to directory /mnt/flash by using the following command:

mount /dev/sda1 /mnt/flash

Now provide the path of the output video file as "/mnt/flash/screendump.mkv". A video file is created at ./screendump.mkv and application start to dump the video data from the window.

The dataflow model of the implementation is illustrated in Figure 4. The time varying data available in the Monitor is converted into MPEG format and stored into a FLASH Memory available in the PowerPC Board. Which can be played from there and used for data analysis and further processing. A Sample screenshot of the implementation is given in Figure 5.

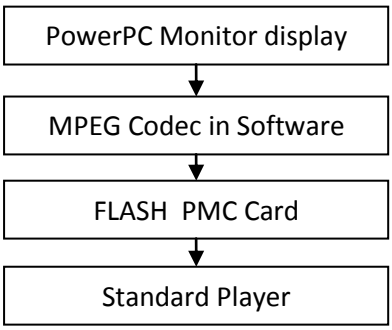

Figure 4. Data FLow Model

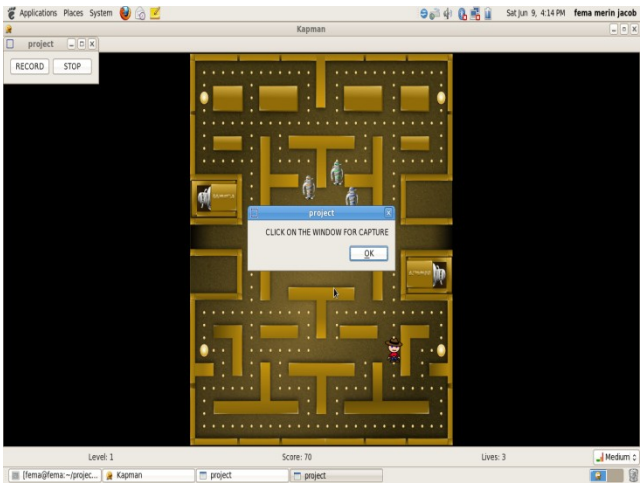

Figure 5. Screenshot of the Implementation

VI. EXPERIMENTAL TESTS AND RESULTS

The application cross compiled and run on DP-VME-0504 PPC 7410 board. The window for capture is selected by clicking on it. A video file is created at ./screendump.mp4 and application start to dump the video data from the window. A new window appears; closing this window stops the capture process and exit application. MPEG 1, MPEG 2 or MPEG 4 codecs can be used to encode the window frames to mpeg video. Depending upon the codec used, the final size of the video file varies. This occurs due to the differences in frame rate and compression ratio. Graph showing the comparison between different mpeg codecs and size of the output video file (in MB) is shown in Figure 6.

The mpeg file recorded will play for the same amount of time for which it was recorded. The mpeg file that has been created has no colour reverses. The captured data and mpeg data has no differences in its colour. There is no flickering in the final mpeg file. There is a smooth flow of frames and hence the final output is as clear and smooth as the input.

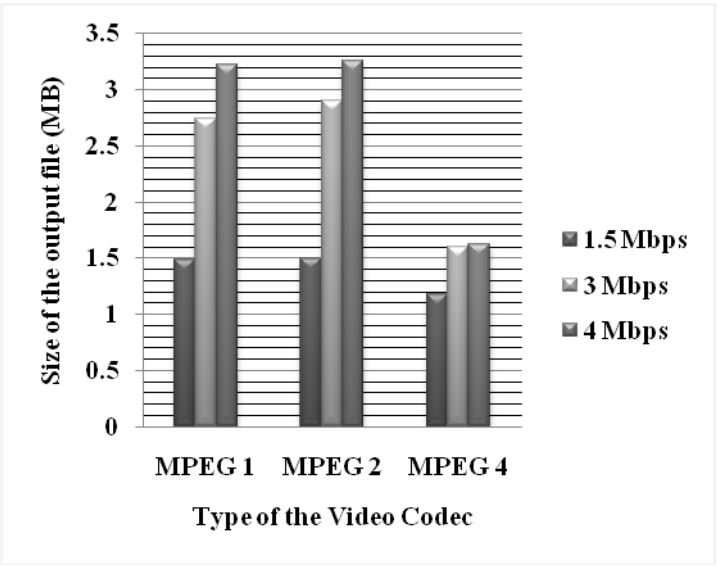

Figure 6. Comparison of Storage Space Utilization for different Video Codecs

VII. CONCLUSION AND FUTURE SCOPE

A design method of recording time varying data like video in an embedded platform is proposed. It is implemented in DP-VME-0504 PowerPC platform. The use of Qt/Embedded greatly increases the reliability of the system and makes the codes to run in cross-platform environments. Open-source *FFmpeg* is used to compress the window frames to mpeg video. Experimental results show that the system fulfils the design requirements in the performance, running speed and stability. The application developed does online recording of a single window on the X11 screen. The video obtained from the operation of the application is stored in the Flash memory attached to the PowerPC Board. This method avoids the complex and costly Digital Video Recorder Method for storing time varying data from the embedded platform. Reliability, compactness, portability etc are achieved through the subsystem design for the video recording of the monitor contents using the same resources available in the

ISSN: 2278 – 1323

hardware and software of the embedded system under consideration. In future, it can be improved, so as to capture the entire contents of frame buffer of graphics PMC card and encode to video. Online encoding of audio along with video can also be incorporated.

REFERENCES

[1] Wang Zheng, Lin Xiaochuan, Zhou Yun-Lian, Ouyang Tianli, "Design and Implementation of Multi-media Player System Based on QT4 & Linux," Journal of Guizhou University (Natural Science Edition), 2009,(1):60-64.
[2] C. Schofield, M. S. Ashley, " Advanced data storage & analysis for oceanographic trials," Electronic Engineering in Oceanography, 19-21 July 1994, Conference Publication No 394.
[3] Trolltech. Qt Reference Documentation [EB/OL]. http://doc.trolltech.com/2.3/index.html.2004.
[4] Trolltech.Qt/Embedded whitepaper[EB/OL]. http://www.trolltech.com/index.html.2006.
[5] Eric F. Johnson and Kevin Reichard,, "*X WindowApplications Programming,"* Tech publications,Singapore, 1989.
[6] http://www.ffmpeg.org
[7] Zhang Qinghui, Li Xudong , "GUI Design of Grain Monitoring and Control System Based on QT," 2010 2nd International Conference on Signal Processing Systems (ICSPS).
[8] Adrian Nye, Xlib Programming Manual, volume1, Q'Reilly and Associates Inc., 1988.
[9] Jie Cao, Lei Yin, Hong Zhao, "Design and Development of Embedded Multimedia Terminal," 2010 Ninth International Symposium on Distributed Computing and Applications to Business, Engineering and Science.
[10] Dalheimer, M.K., Hansen S, "Embedded development with Qt/Embedded.," Dr.Dobb's Journal 2002.vol 27.no.3. pp.48-53.
[11] Zu-jue Chen, Zhi-xiong Zhang, Jian-jiang Zhang, "Design and Implementation of Video Player System Based on Embedded System and Qt/E," VIE 2008, 5th International Conference on July 29 2008-Aug.1,2008:468-472.
[12] Steve Bunch, Ron Hochsprung, Todd Moore, " PowerPC Platform: A System Architecture," 1996 IEEE Proceedings of COMPCON '96.
[13] Yanqing Liu, Guilian Su, "Qt4-based graphical user interface design and implementation process," 2009.
[14] Valerie Quercia, and Tim O'Reilly, "*X* Window *System User's Guide: for XII release 5,"* O'Reilly and Associates Inc., Calif., 1993.
[15] TAN Da-peng, LI Pei-yu, PAN Xiao-hong, LIN Bo-yu, "Lightweight GUI components library development oriented to embedded industry monitoring system based on Qt/E," 2007 International Conference on Convergence Information Technology.
[16] Michael A. Dolan Larry Hare.,"X Window System Servers in Embedded Systems," 1990 IEEE.
[17] Karim Yaghmour, "Building Embedded Linux Systems," O'Reily Media. April 2003.

Fema Merin Jacob received her M.Tech Degree in Embedded Systems and B. Tech Degree in Electronics and Communication Engineering from National Institute of Electronics and Information Technology (NIELIT), Calicut and College of Engineering Chengannur, Kerala respectively. Currently she is working as an Assistant Professor in ECE Department, Bannari Amman Institute of Technology, Sathyamangalam. Her area of interest include Embedded Systems, Image Processing and VLSI Design.

Sajan P. Philip received his M.E Degree in Applied Electronics and B. Tech Degree in Electronics and Communication Engineering from Bannari Amman Institute of Technology, Tamil Nadu and College of Engineering Chengannur, Kerala respectively. Currently he is working as an Assistant Professor in ECE Department, Bannari Amman Institute of Technology, Sathyamangalam. Her area of interest include Embedded Systems, Image Processing and VLSI Design.

Dinesh T is pursuing his B.E Degree in Electronics and Communication Engineering in Bannari Amman Institute of Technology. His area of interest include Embedded Systems and Digital System Design.

ISSN: 2278 – 1323