GStreamer Based VMS

Karan Gandhi[1] S. Anand Kumar[2] Prof. Vijay Patel[3]
[1]M.tech (Embedded System) [3]Associate professor
[1,3]Department of Electronics and Communication Engineering
[1,3]UVPCE, Ganpat University, India [2]eInfochips Pvt. Ltd., Ahmedabad, India

Abstract— VMS (Video Management System) is a system which manages access and controls the IP cameras. VMS software running on a Windows or Unix/Linux server, supplies the basis for video monitoring via GUI. In General people have VMS on their desktops so they have to stick in front of desktop but here we are introducing a system which runs on Android based phones and tablets and even on web browsers so people don't have to stick with their desktops they can walk over the premises (where Wi-Fi is necessary) and see the streaming of any IP camera in their android based phone or tablet so you are under surveillance every time even if the controller of VMS is not on their spot. Even ifthey can stream the particular IP camera on the web browser they just need to run the HTML from the server and give the IP of camera on edit text field in browser.Here in this Paper we have introduced "GStreamer Based VMS", the design is based on GStreamer which is a multimedia framework. The System we have developed is compatible with Windows & Linux desktop (It's a desktop based VMS made with Java), Android phones & Tablets and All the browsers which has enabled Java plugin (It's a Java Applet).

I. INTRODUCTION

Video Management System (VMS) is a system which manages access and controls the IP cameras. VMS software running on a Windows or Unix/Linux server, supplies the basis for video monitoring via GUI.

To view several cameras at the same time, dedicated VMS software is required. You can view up to 4 different live streaming on the same window of GUI.

VMS also displays the list view of IPs of available camera in the same network.

File playback is possible in the mini VMS software GUI. You can view stored video from you computer.

Features such as:

- RTSP live streaming & file playback
- 1/2/4 window live streaming of different IP cameras
- List view of available IP cameras in the network
- Available for windows and Linux machines & Android
- For Android you can create your own database
- Capacity to handle high frame rates and large amounts of data

Here we have developed 3 applications for VMS using GStreamer a media framework. One is a simple Desktop based Application for windows and Linux platform. We can view up to 4 live streaming of different IP cameras with that application and even we can use that for file playback like any media players and the another provision is list of all IP cameras available in the network is listed in the application in tree view. Another one is Java Applet that works with any browsers like internet explorer, Firefox, chrome, opera, safari, etc. We made an html page that will be on the server when you request for that page it will show you the live streaming of any IP camera to that browser. In this particular part we have just 1 live streaming support we can enter IP Address of the camera and username password to the given edit text fieldand it will stream the view of the camera in the browser. And the third one is Android Application for VMS. Now a days the market moves towards Android so for that perspective we made an Android Application for the users. With this Application you can view the live streaming of the IP camera directly on your Android device weather it is a phone or a tablet even we have a database included with the Application so that you can store the IP cameras with appropriate names even we have a facility of recording the video of the camera and take a snap of live streaming and you can view multiple cameras with the application. And we support Axis, Zooter and Cisco cameras.

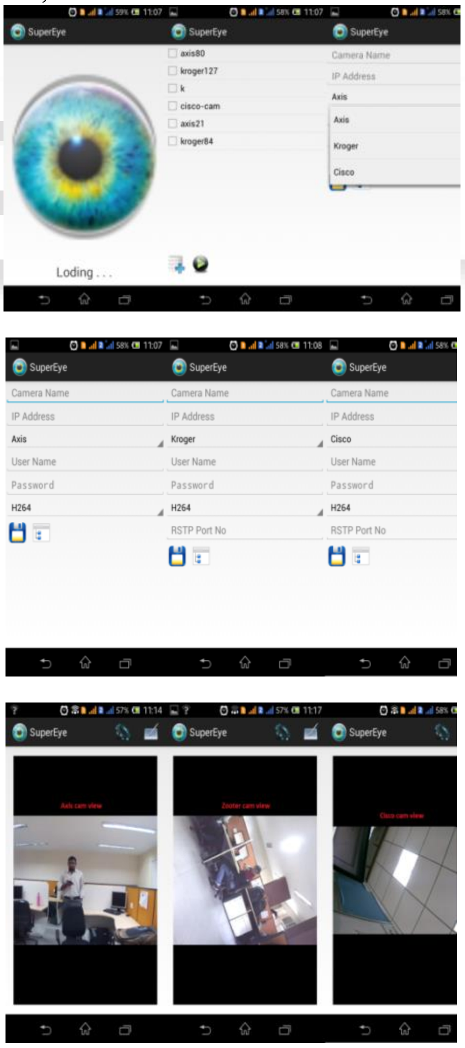


Fig. 1(a),(b),(c): support Axis, Zooter cameras

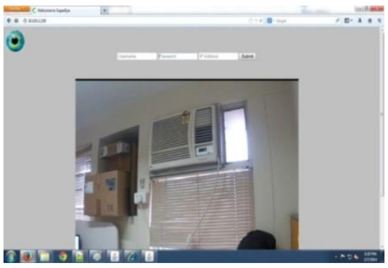


Fig. 2: Cisco cameras

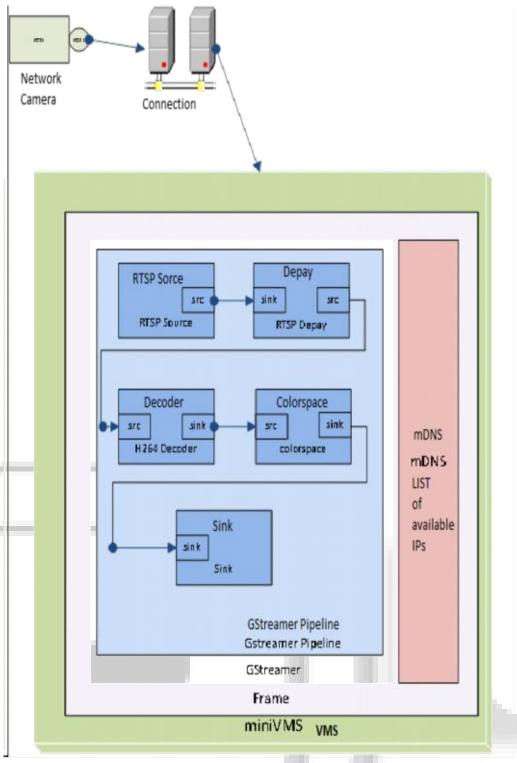
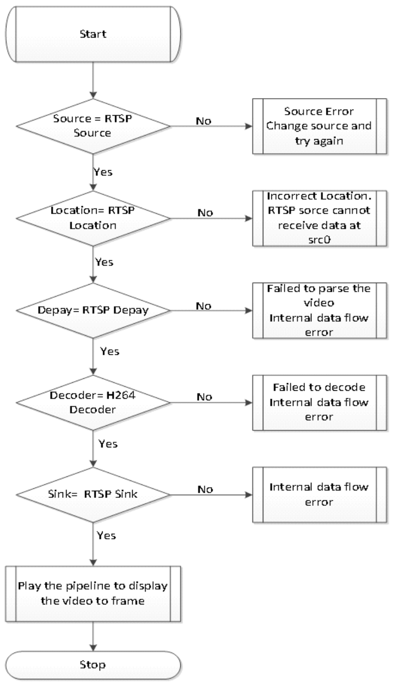


Fig. 3: Component Diagram


Start
Source = RTSP Source
No
Source Error Change source and try again
Yes
Location= RTSP Location
No
Incorrect Location. RTSP sorce cannot receive data at src0
Yes
Depay= RTSP Depay
No
Failed to parse the video Internal data flow error
Yes
Decoder= H264 Decoder
No
Failed to decode Internal data flow error
Yes
Sink= RTSP Sink
No
Internal data flow error
Yes
Play the pipeline to display the video to frame
Stop
Fig. 4: Flow Chart Gstreamer Internal Communication

II. Implementation of Project

A. Short Introduction to GStreamer[1]

It is aPipeline based multimedia framework.

- Cross platform, open source
- Bindings for many languages
- Stable API/ABI
- LGPL
- Runs on Linux, Solaris, *BSD, OSX, Windows.
- X86, PPC, ARM, SPARC...
- Python, C++, .NET, Perl, Ruby...
- Flexible and extensible design
- Plugin-based architecture
- Easy to integrate with other software
- Generic format negotiation mechanisms
- Synchronization and data transport of media
- Flexible communication between app and framework
- Plugins for all important codecs and containers
- Proprietary plugins for patented codecs
- Plugins for different filters
- Hardware support
- Support for many different use cases

B. Java Applet

Java applets are programs that are embedded in other applications, typically in a Web page displayed in a Web browser.First we need Java plug-ins installed and enabled in the browser to run an Applet into a browser. So we had installed and enabled the java plugins in Linux and Windows machine but the one thing that bothering us is security of java we are not authorized to java so we have to minimize the security level. Then create a jar file of the Java code. But we found a problem with that too we can't put a simple jar file at the reference we need to digitally sign the jar file then and then we use that as a reference. A digital signature is a mathematical scheme for demonstrating the authenticity of a digital message or document. A valid digital signature gives a recipient reason to believe that the message was created by a known sender, such that the sender cannot deny having sent the message (authentication and non-repudiation) and that the message was not altered in transit (integrity). The ability to sign and verify files is an important part of the Java platform's security architecture. Security is controlled by the security *policy* that's in force at runtime. You can configure the policy to grant security privileges to applets and to applications. Forexample, you could grant permission to an applet to perform normally forbidden operations such as reading andwriting local files or running localexecutable programs. If you have downloaded some code that's signed by a trusted entity, you can use that fact as a criterion in deciding which security permissions to assign to the code. When you sign a JAR file your public key is placed inside the archive along with an associated certificate so that it's easily available for use by anyone wanting to verify your signature.

1) To summarize digital signing [4]
- The signer signs the JAR file using a private key.
- The corresponding public key is placed in the JAR file, together with its certificate, so that it is available for use by anyone who wants to verify the signature.

2) Creating a Policy for Java Applet[2]

A policy file is an ASCII text file and can be composed via a text editor or the graphical Policy Tool utility.

The Policy Tool saves you typing and eliminates the need for you to know the required syntax of policy files, thus reducing errors.

We used the Policy Tool to create a policy file named example policy, in which you will add a policy entry that grants code from the directory where *.class is stored permission to write in file.

```
/* AUTOMATICALLY GENERATED ON Mon nov 11
17:20:59 UTC 2013*/
/* DO NOT EDIT */
grant {
   permission java.security.AllPermission;
};
<html>
<head>
<title>My Java Applet</title>
</head>
<body>
<applet code="Hello.class" width="200" height="200">
</applet>
</body>
</html>
```

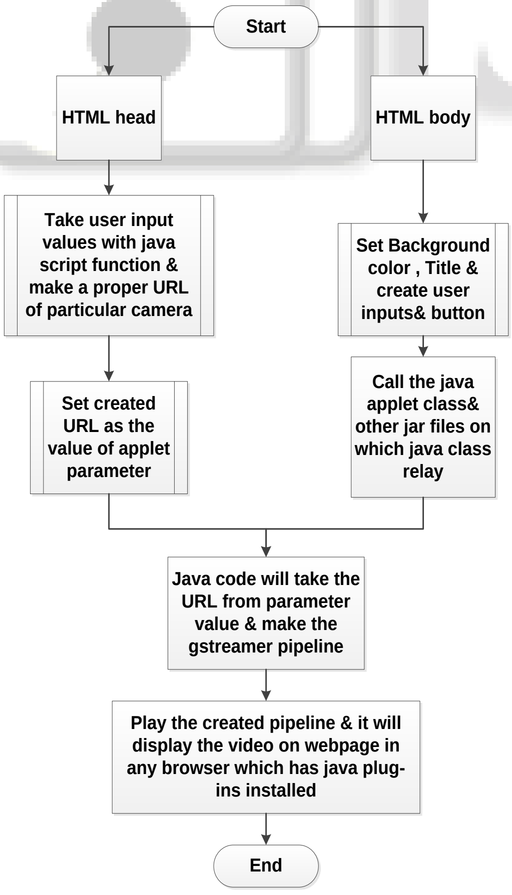

Start
HTML head
HTML body
Take user input values with java script function & make a proper URL of particular camera
Set Background color , Title & create user inputs& button
Set created URL as the value of applet parameter
Call the java applet class& other jar files on which java class relay
Java code will take the URL from parameter value & make the gstreamer pipeline
Play the created pipeline & it will display the video on webpage in any browser which has java plug-ins installed
End

Fig. 5: Flow chart Java Applet

An applet is placed in an HTML document using the <applet> HTML element. The applet tag has three attributes set: code="Hello" specifies the name of the Applet class and width="200" height="200" sets the pixel width and height of the applet. Applets may also be embedded in HTML using either the object or embed element, although support for these elements by Web browsers is inconsistent. However, the applet tag is deprecated, so the object tag is preferred where supported.

The host application, typically a Web browser, instantiates the Hello applet and creates an Applet Context for the applet. Once the applet has initialized itself, it is added to the AWT display hierarchy. The paintComponent() method is called by the AWT event dispatching thread whenever the display needs the applet to draw itself.

C. Android Application

1) Why we used GStreamer on Android that is the first question on any one's mind so let me explain you,We want much more than just playback or capture, We want to write any kind of application from Non-linear.

GStreamer has almost everything we need:
- Supports a very large number of formats.
- Support for more uses cases
- Multimedia backend re-usable across platforms.

2) Problems with using GStreamer on Android
- Plugin-based architecture> too many shared libraries
- GStreamer itself only depends onglib, libxml2, libffi and libz,but plugins pull-in many dependencies
- Android's dynamic linker limits the number of shared Libraries per process.
- Android's dynamic linker has a hard-coded limit on the number
- of .so files (shared libraries and/or plugins) you can load in a single process.
- We have more than 262 shared libraries
- Android's linker is limited to 64, 96 and 128 shared libraries
- Including all plugins we have 262 shared libraries
- The NDK is limited: C library (BIONIC) and otherlibraries like OpenSL.
- Other API's are not even available in C like the
- MediaCodec API

3) How we solved it.Static linking with re-locatable archives.
- A single shared library with everything:
- libgstreamer_android.so
- Integration with ndk-build to link this shared library:
- Complies with the LGPL requirement.
- Allows selecting only the plugins being used.

4) Static plugins and modules of GStreamer

- GStreamer plugins and GIO modules mustbe handled in a different way.
- We are trying to get these changes upstream
- Static plugins need to be registered manually.
- Instead of being loaded manually from path we must explicitly registerthem.

5) Integration with ndk-build[3]

- A set of makefiles that extend nkd-build's core to generate libgstreamer_android.so and link it to the application
- From the point of view of application developers we tried to make things as easy as possible.

Example of Android.mk from the Android NDK samples including GStreamer

```
LOCAL_PATH := $(call my-dir)
include $(CLEAR_VARS)
LOCAL_MODULE := hello-jni
LOCAL_SRC_FILES := hello-jni.c
LOCAL_SHARED_LIBRARIES := gstreamer_android
include $(BUILD_SHARED_LIBRARY)
include $(CLEAR_VARS)
include
$(GSTREAMER_NDK_BUILD_PATH)/plugins.mk
GSTREAMER_SDK_ROOT := /home/cerbero/android_arm
GSTREAMER_PLUGINS =
$(GSTREAMER_PLUGINS_CORE)
$(GSTREAMER_PLUGINS_CODECS)
GSTREAMER_EXTRA_DEPS := json-glib-1.0
include
$(GSTREAMER_NDK_BUILD_PATH)/gstreamer.mk
```

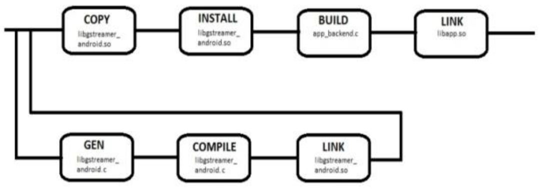


Fig. 4: NDK-Build build Steps gstreamer_android.c:

- Redirects GStreamer logs to logcat and adds an entry point to initialize GStreamer and register static plugins.
- Libraries used from the C backend must be explicetly listed to include the whole archive with --whole-archive, otherwise the linker will not include the object files as no symbol is used by the gstreamer plugins.)

6) Developing applications with the SDK

GStreamer projects can be built using the regular tools

- For Eclipse: using the wizard and
- project→ Android Tools →Add Native Support
- Command line: using the standard Ant build command
- jni/Android.mk must be updated for GStreamer

```
LOCAL_PATH := $(call my-dir)
include $(CLEAR_VARS)
LOCAL_MODULE := hello-jni
LOCAL_SRC_FILES := hello-jni.c
LOCAL_SHARED_LIBRARIES := gstreamer_android
include $(BUILD_SHARED_LIBRARY)
include $(CLEAR_VARS)
include
$(GSTREAMER_NDK_BUILD_PATH)/plugins.mk
GSTREAMER_SDK_ROOT := /home/cerbero/android_arm
GSTREAMER_PLUGINS =
$(GSTREAMER_PLUGINS_CORE)
$(GSTREAMER_PLUGINS_CODECS)
include
$(GSTREAMER_NDK_BUILD_PATH)/gstreamer.mk
```

- Multimedia backend is written in C
- Bind the backend API to use it in the application through JNI
- Bind backend registering dynamic methods with RegisterNatives
- Declare this new methods as dynamic in the Java side

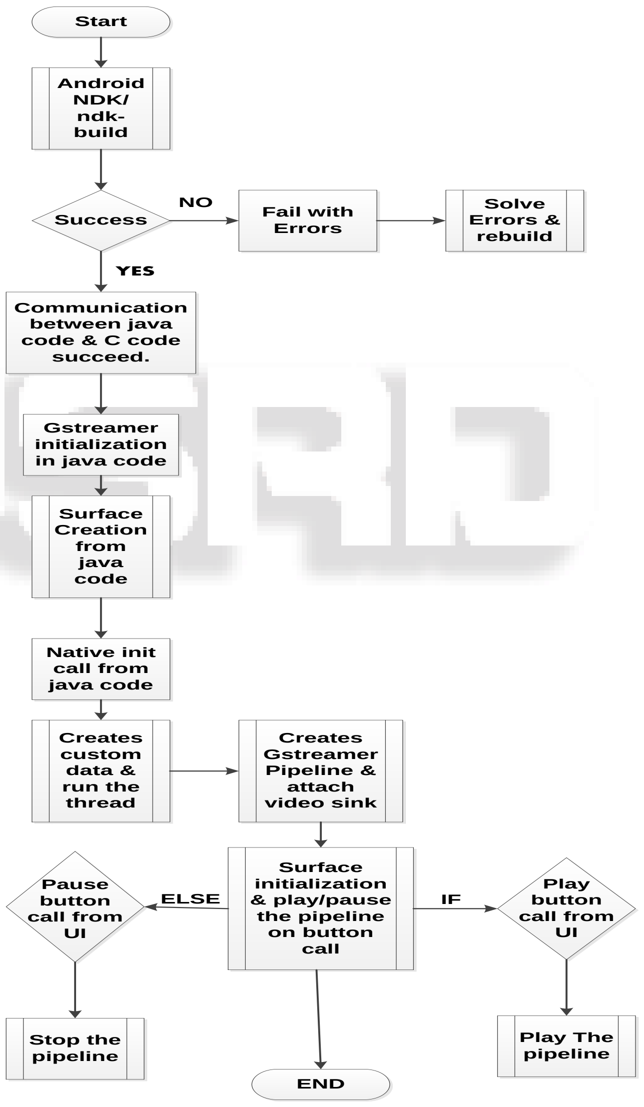

Start
Android NDK/ ndk-build
Success
NO
Fail with Errors
Solve Errors & rebuild
YES
Communication between java code & C code succeed.
Gstreamer initialization in java code
Surface Creation from java code
Native init call from java code
Creates custom data & run the thread
Creates Gstreamer Pipeline & attach video sink
Surface initialization & play/pause the pipeline on button call
ELSE
IF
Pause button call from UI
Play button call from UI
Stop the pipeline
Play The pipeline
END

Fig. 5: Flow Chart GStreamer communication with Android

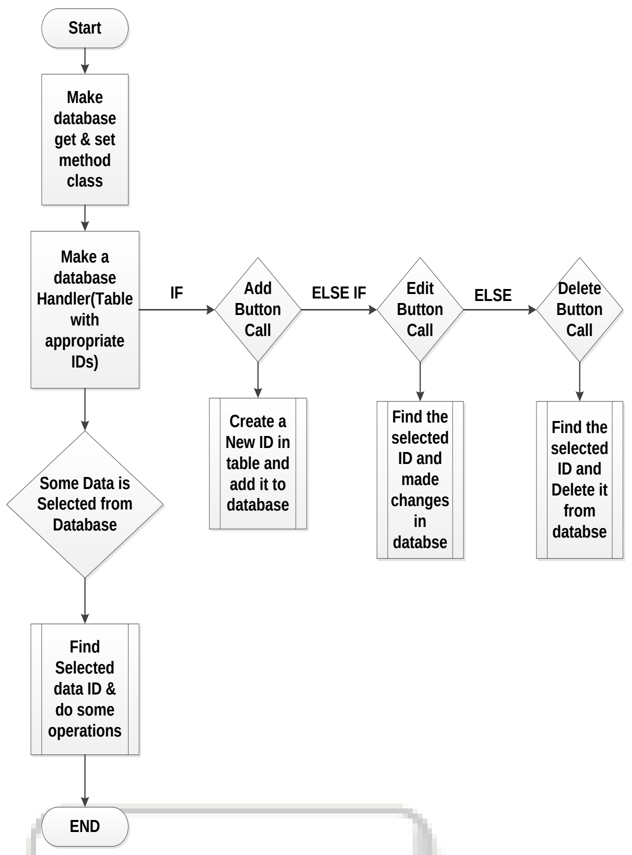

Start
Make database get & set method class
Make a database Handler(Table with appropriate IDs)
IF
Add Button Call
ELSE IF
Edit Button Call
ELSE
Delete Button Call
Create a New ID in table and add it to database
Find the selected ID and made changes in databse
Find the selected ID and Delete it from databse
Some Data is Selected from Database
Find Selected data ID & do some operations
END

Fig. 6: Flow Chart Android Database

```
/* List of implemented native methods */
static JNINativeMethodnative_methods[] = {
{ "nativeInit", "()V", (void *) gst_native_init},
{ "nativeFinalize", "()V", (void *) gst_native_finalize},
{ "nativePlay", "()V", (void *) gst_native_play},
{ "nativePause", "()V", (void *) gst_native_pause},
{ "nativeClassInit", "()Z", (void *) gst_native_class_init}
};
/* Library initializer */
jintJNI_OnLoad(JavaVM *vm, void *reserved) {
JNIEnv *env = NULL;
java_vm = vm;
if ((*vm)->GetEnv(vm, (void**) &env,
JNI_VERSION_1_4) != JNI_OK) {
__android_log_print (ANDROID_LOG_ERROR, "tutorial-
2", "Could not retrieve JNIEnv");
return 0;
}
jclassklass = (*env)->FindClass (env,
"com/gst_sdk_tutorials/tutorial_2/Tutorial2");
(*env)->RegisterNatives (env, klass, native_methods,
G_N_ELEMENTS(native_methods));
pthread_key_create (&current_jni_env,
detach_current_thread);
return JNI_VERSION_1_4;}
```

- Glib's main loop is run in a separate thread
- Use Thread-Local Storage (TLS) for storing the JNI env
- Load libgstreamer_android.so in the application

III. TESTING& TROUBLESHOOTING

A. Unsatisfied Link Error in JNI

- Common reasons why you might encounter "library not found" exceptions:
- The library doesn't exist or isn't accessible to the app. Use adb shell ls -l <path> to check its presence and permissions.
- The library wasn't built with the NDK. This can result in dependencies on functions or libraries that don't exist on the device.
- The library isn't getting loaded. Check the logcat output for messages about library loading.
- The method isn't being found due to a name or signature mismatch.
- Using javah to automatically generate JNI headers may help avoid some problems.

B. FindClass doesn't find my class in JNI

There are a few ways to work around this:

- Do your FindClass lookups once, in JNI_OnLoad, and cache the class references for later use. AnyFindClass calls made as part of executing JNI_OnLoad will use the class loader associated with the function that called System.loadLibrary (this is a special rule, provided to make library initialization more convenient). If your app code is loading the library, FindClass will use the correct class loader.
- Pass an instance of the class into the functions that need it, by declaring your native method to take a Class argument and then passing *.class in.
- Cache a reference to the ClassLoader object somewhere handy, and issue loadClass calls directly. This requires some effort.

C. How do I share raw data with native code

You may find yourself in a situation where you need to access a large buffer of raw data from both managed and native code. Common examples include manipulation of bitmaps or sound samples. There are two basic approaches.

- You can store the data in a byte[]. This allows very fast access from managed code. On the native side, however, you're not guaranteed to be able to access the data without having to copy it.
- The alternative is to store the data in a direct byte buffer. These can be created with java.nio.ByteBuffer.allocateDirect, or the JNI NewDirectByteBuffer function.
  Unlike regular byte buffers, the storage is not allocated on the managed heap, and can always be accessed directly from native. Depending on how direct byte buffer access is implemented, accessing the data from managed code can be very slow.

D. GStreamer compilation on windows[5]

I had tried to compile GStreamer from its source code on windows 7 machine with MinGW cross compilation tool but

I still didn't find any solution for that. I had installed autoconfig, automake, gettext, libtool, pkg-config for windows (needed tools for compile GStreamer) and tried to change autogen.sh &makefile but I failed to compile GStreamer on a windows machine. I had successfully compiled GStreamer on a Linux machine.

IV. CONCLUSION

By doing this project we can conclude that, using GStreamer as a multimedia framework, we can build platform independent application.

V. APPLICATIONS

– Video Surveillance,on cross platform with single design
– Video Lecture viewer
– Handled application on android mobile for parking monitor
– Light weight video conference on 3G network, integrating with SIP protocol
– Baby monitoring app on android mobile.
– Traffic monitoring and handling
– Machine work or Employee monitoring

ACKNOWLEDGEMENT

I would like to express my deep sense of gratitude to all those who helped me in this endeavour. My sincere thanks to eInfochips for providing me a very good platform for training and Project where I got the experience of working in team and watching closely the execution of different project phase. I would like to thank following people who believed in me and motivated me throughout my training period. Mr. S.Anand Kumar (RA): For providing opportunity to work in collaboration as a team, and giving his useful insights and showing me the way whenever some issues and problems came. Prof. V.K. Patel (Internal guide) for providing useful insights on how to approach a project, also providing useful insights on project.

REFERENCES

[1] **"**GStreamer**"** http://gstreamer.freedesktop.org
[2] "Creating a Policy for Java Applet" http://docs.oracle.com/javase/tutorial/security/tour1/step2.html
[3] "Android NDK" https://developer.android.com/tools/sdk/ndk/index.html
[4] "Signing a JAR file" http://docs.oracle.com/javase/tutorial/security/apisign/gensig.html
[5] "GStreamer compilation guide for windows" http://docs.gstreamer.com/display/GstSDK/Building+from+source+using+Cerbero