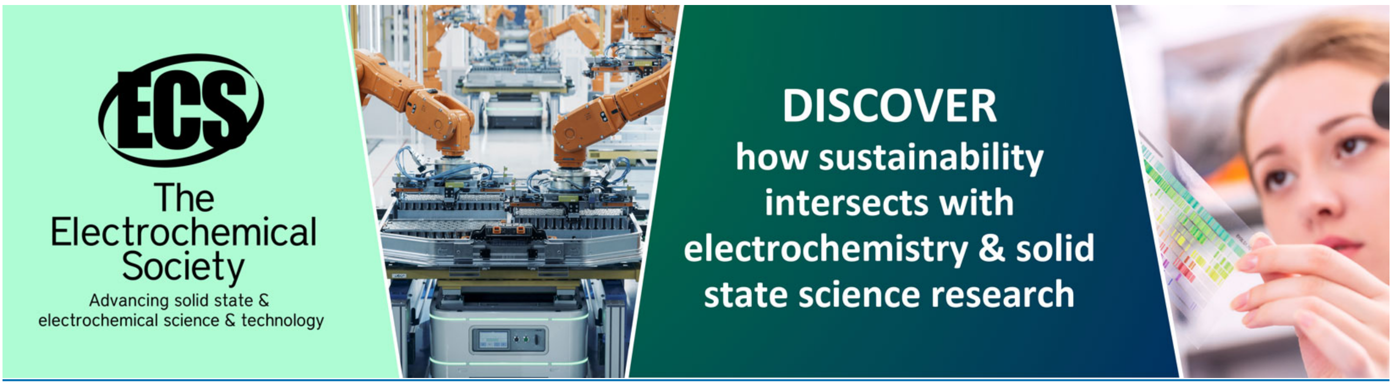
Video streaming technologies using ActiveX and LabVIEW


ECS
The Electrochemical Society
Advancing solid state & electrochemical science & technology
DISCOVER
how sustainability intersects with electrochemistry & solid state science research

Video streaming technologies using ActiveX and LabVIEW

M Panoiu[1] C L Rat[1] and C Panoiu[1]

[1] Politehnica University of Timisoara, Department of Electrical Engineering and Industrial Informatics, 5 Revolution Street, 331128 Hunedoara, Romania

E-mail: manuela.panoiu@fih.upt.ro

Abstract. The goal of this paper is to present the possibilities of remote image processing through data exchange between two programming technologies: LabVIEW and ActiveX. ActiveX refers to the process of controlling one program from another via ActiveX component; where one program acts as the client and the other as the server. LabVIEW can be either client or server. Both programs (client and server) exist independent of each other but are able to share information. The client communicates with the ActiveX objects that the server opens to allow the sharing of information [7]. In the case of video streaming [1] [2], most ActiveX controls can only display the data, being incapable of transforming it into a data type that LabVIEW can process. This becomes problematic when the system is used for remote image processing. The LabVIEW environment itself provides little if any possibilities for video streaming, and the methods it does offer are usually not high performance, but it possesses high performance toolkits and modules specialized in image processing, making it ideal for processing the captured data. Therefore, we chose to use existing software, specialized in video streaming along with LabVIEW and to capture the data provided by them, for further use, within LabVIEW. The software we studied (the ActiveX controls of a series of media players that utilize streaming technology) provide high quality data and a very small transmission delay, ensuring the reliability of the results of the image processing.

1. Introduction

Remote monitoring and processing of real-time images can represents quite a challenge, mainly because the possibilities of performing these tasks are quite limited. Transmitting images of a phenomenon in real-time and making these images available to the environment in which we want to process them is complicated, especially if we desire high quality data in order to ensure the reliability of the results of the processing. Our chosen environment for data processing is NI LabVIEW because it provides high performance toolkits and modules for image processing [5], [6], [8]. Unfortunately, the LabVIEW environment provides little if any possibilities for video streaming, and the methods it does offer are usually not high performance. Hence, we chose for avoiding this inconvenience by using existing software, specialized in video streaming along with LabVIEW and to capture the data provided by it, for further use, within LabVIEW. The aforementioned software is formed of ActiveX components that can be embedded in and controlled from LabVIEW.

But these components seldom provide access methods to the data they render, making that data invisible to the LabVIEW environment and impossible to retrieve. Hence resulting the need for additional programs that can extract data form these components and make it accessible to LabVIEW. This method, although seemingly very intricate, could provide excellent results in time. This paper provides our experimental results regarding the components that can be used and how the data can be recovered.

2. ActiveX Technology

ActiveX is the general name for a set of Microsoft Technologies that allows code reuse and the connecting of individual programs together. Based on COM (Component Object Model) technologies, ActiveX is an extension of a previous technology called OLE (Object Linking and Embedding). Each program does not regenerate components, but rather, reuses components allowing for the combination of applications together.

The most common usage of ActiveX is via ActiveX controls, which are embeddable components that exist inside ActiveX containers. From these containers, the ActiveX controls have their own methods and properties that can be used in order to manipulate the control. LabVIEW constitutes such an ActiveX container and can therefore house ActiveX controls.

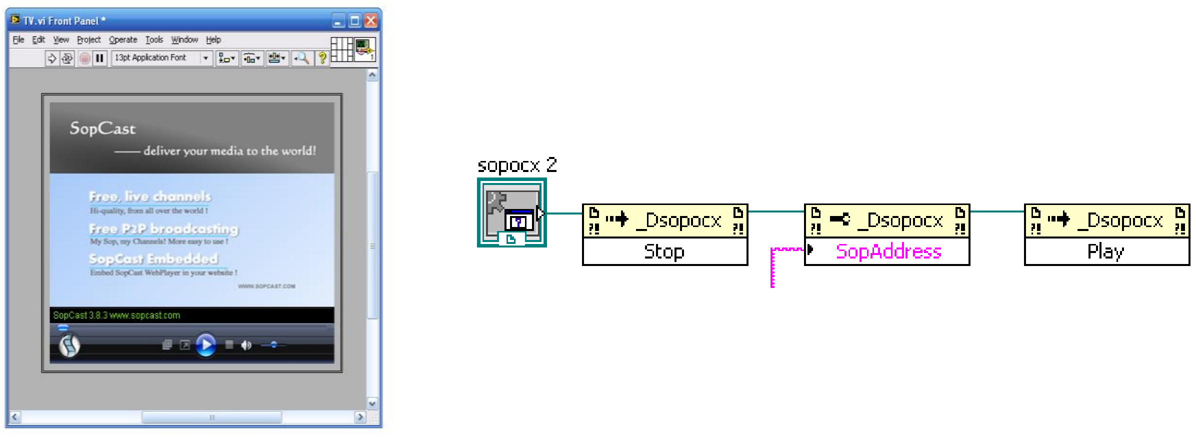
In general, LabVIEW can embed any ActiveX control. Using the control's properties and methods, LabVIEW can programmatically interact with the control.

2.1. SopCast ActiveX

Sopcast is a Streaming Direct Broadcasting System based on P2P, SoP being the abbreviation for Streaming over P2P. Its core communication protocol which is named sop://, or SoP technology.

Due to the fact that it uses P2P technology, channels are easily available and more stable, having a minimal delay in the P2P streaming market and fast buffering (10 – 30 seconds). It is capable of real-time streaming, supporting two main streaming transport protocols (mms, http), as well as streaming media files such as .asf, .wmv, .rm, .rmvb, mp3, .spl. It can also redirect streams towards other players such as Windows Media Player, RealPlayer, VLC, etc.

SopCast has 2 main components a SopCast Client, also named SopPlayer, and a SopCast Server, also named SopServer or SO (Stream Originator). The Sop Player has an ActiveX component that allows it to be embedded into a webpage or any software applications. This feature allowed the author to embed it into a LabVIEW application (Figure 1).



a) **b)**

Figure 1. SopCast embedded in LabVIEW

a) Front Panel featuring the SopCast ActiveX Control

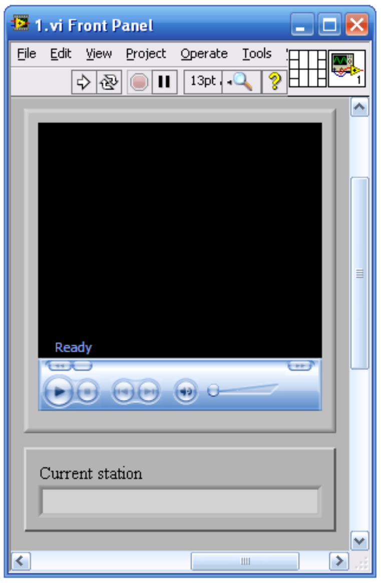
b) Use of methods and properties of the SopCast Control (code for streaming video content from the address given as a String)

Its most important feature, which constitutes the main reason why it was included in this study, is the fact that it allows the users to build their own channels and broadcast it over the Internet, being able to broadcast both video and audio content. It, therefore, constitutes a server and a source of high quality data. It contains security features for the broadcasters and the viewers, the broadcaster having full control on his channels. It supports multiple channels broadcast on the same server (5-10 channels). It also has an extremely low memory footprint and CPU load [9].

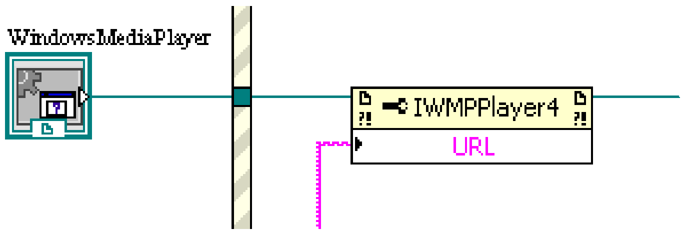
2.2. Windows Media Player ActiveX

The reason why Windows Media Player in included this study is the fact that it is capable of receiving streams through 3 major technologies: the User Datagram Protocol (UDP), the Transmission Control Protocol (TCP), and the Hypertext Transfer Protocol (HTTP). It can also receive a multicast stream. UDP and TCP are basic networking protocols that manage more fundamental tasks such as network connectivity and packet error correction. These are used in combination with the Microsoft Media Server (MMS) and Real Time Streaming Protocol (RTSP) protocols, which manage the high-level exchange of data. Hence, the protocols which are used in managing the high-level data exchange when streaming Windows Media-based content are MMS, RTSP and HTTP [3]. Windows Media-based content consists of a large number of formats: .asf, .asx, .avi, .wav, .wma, .wax, .wm, .wmv, .wvx (Windows Media formats), .ra, .ram, .rm, .rmm (RealAudio and RealVideo), m3u, mp2v, .mpg, .mpeg, .m1v, .mp2, .mp3, .m3u, .mpa, .mpe, mpv2 (MPEG), .mid, midi, .rmi (MIDI), .mov (Apple QuickTime), .aif, .aifc, .aiff (Macintosh AIFF Resource), .ivf (Intel Video File), .avi (Audio Visual Interleave), .swf (Macromedia Flash).

Windows Media Player also has an ActiveX component, making it possible to embed it into LabVIEW (Figure 2).



a)



b)

Figure 2. Windows Media Player embedded in LabVIEW

a) Front Panel featuring the Windows Media Player ActiveX Control

b) Use of methods and properties of the Windows Media Player Control (code for streaming video content from the URL address given as a String)

2.3. *VLC ActiveX*
VLC media player (commonly known as VLC) is a portable, free and open-source, cross-platform media player and streaming media server written by the VideoLAN project. VLC media player supports many audio and video compression methods and file formats, including streaming protocols.

It is able to stream media over computer networks, supporting the following streaming protocols: RTP/UDP, RTSP, RTP/DCCP, Raw UDP, (RTP or raw) Multicast, File, HTTP, and MMSH. It also plays m2t MPEG transport streams, (.TS) files while they are still being digitized from an HDV camera via a FireWire cable, making it possible to monitor the video as it is being played.

Because VLC is a packet-based media player, it plays almost any type of video content, even if it is damaged, incomplete, or unfinished; such as files that are still downloading via a peer-to-peer (P2P) network.

It is also capable of transcoding multimedia files. VLC can transcode or stream audio and video into several formats including: ASF, AVI, FLV, MP4, Ogg, WAV, MPEG-2 (ES, PS, TS, PVA, and MP3), MPJPEG, FLAC, QuickTime File Format, Matroska, WebM, H.263, H.264/MPEG-4 AVC, MJPEG, MPEG-1, MPEG-2, MPEG-4 Part 2, VP5, VP6, VP8, and VP9 [4], [10].

VLC is able to receive streamed data form such sources as SopCast and it can transcode it into an AVI format which can be read with NI Vision components in LabVIEW. This method is not ideal for real-time image processing because of the delay in writing the .avi file and reading it. There are also limits to how large the .avi file can be; making it mandatory to occasionally interrupt the process and create new storage files. Another reason why this method is inconvenient is because it takes up much disk space.

Also, VLC’s ActiveX components are not functional and cannot be embedded into LabVIEW. There is a way of incorporating it into LabVIEW using dynamic libraries, but the method becomes extremely complicated. [11]

2.4. *Other ActiveX Components*
There are a number of other ActiveX Components that, although are not expressly used for video streaming, can be used to render streamed data. We have explored this option because of the possibility that there might not be a stand-alone ActiveX Component that can render a certain data stream. Our purpose is to represent the options available while adapting to the source, but we considered the notion that there might not always be an elegant way to do this. Hence, we present a few universal means that serve the purpose. One of these is the Microsoft Web Browser ActiveX Component (Figure 3). It is a simple browser that can be very useful when it comes to live streaming. Another option would be the ShockwaveFlash ActiveX (Figure 4), which is basically the Shockwave Player. This is a component that is usually required in computers that work with the Internet. Literature indicates that it is capable of streaming data but there is not much information on how this is achieved [12]. This component was mentioned because it represents the best solution for rendering flash videos, which are a convenient option for transferring video data over the internet, being, therefore, extremely widespread.

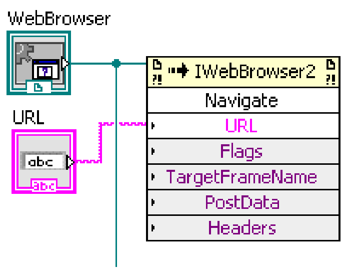


Figure 3. WebBrowser ActiveX Control - code for streaming video content from the URL address given as a String.

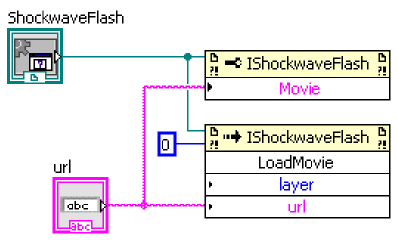


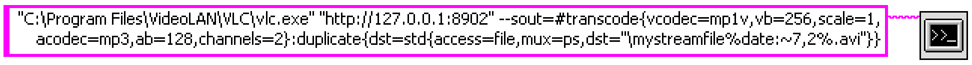
Figure 4. ShockwaveFlash ActiveX Control - code for streaming video content from the URL address given as a String.

3. Transcoding

Although ActiveX components can be embedded into LabVIEW and programmatically controlled by it, the access to them is not complete: they do not have access methods to the data they display and they do not usually have transcode methods that output in a format LabVIEW can recognize, namely AVI. The lack of these things becomes a serious impediment when the data rendered must be extracted for further processing. Therefore, the question of how to make the data available to the environment in which we want to process it comes to mind. We will analyze the options for each of the aforementioned technologies.

The advantage of using SopCast is that it makes broadcasting data simple and efficient, but it does not accept many formats and AVI is not among the ones it accepts. SopCast does not have a transcode method, but it is capable of redirecting steams towards VLC and Windows Media Player.

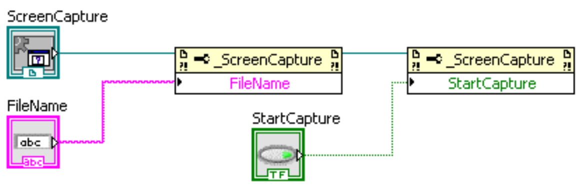
VLC can transcode it to AVI, but Window Media Player can only play the stream. VLC, as a streaming program, has the advantage of being able to recognize a large number of steaming technologies, as well as being able to stream itself, being, in this sense, both a server and a client. Yet it has the great disadvantage of not having functional ActiveX components, making it a challenge to embed it into LabVIEW. There are solutions, but they are considerably intricate [11]. Hence, the control of VLC has to be done either manually or through programmatic command line within LabVIEW (Figure 5).

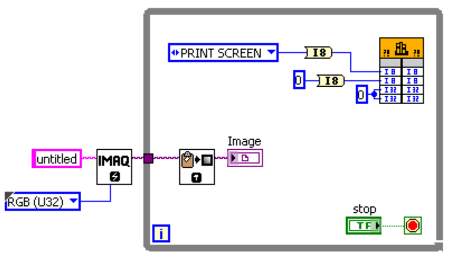


```
"C:\Program Files\VideoLAN\VLC\vlc.exe" "http://127.0.0.1:8902" --sout=#transcode{vcodec=mp1v,vb=256,scale=1,
  acodec=mp3,ab=128,channels=2}:duplicate{dst=std{access=file,mux=ps,dst="\mystreamfile%date:~7,2%.avi"}}
```

Figure 5. LabVIEW code for a programmatic command line used to work with VLC

Windows Media Player has an ActiveX component that can receive a number of stream technologies, but it cannot transcode these into AVI files. Neither can many of the other ActiveX components mentioned in this paper. So, in order to process the data they received in LabVIEW, we resort to a more indirect way of getting the data. This consists of programmatic PrintScreen functions, as can be seen in Figure 6 a) and b). The first one uses an Activex component named ScreenCapture.ocx, the other presses the PrintScreen key programmatically [13]. Both these methods can be used for image acquisition. These methods usually take a picture of the entire screen, but the region of interest can be easily extracted from the obtained image and further processed. This method has the advantage of acquiring the image directly in LabVIEW and being able to process it strait away. Moments of interest can also be saved as .bmp, .jpeg, .png, .tiff images or as .avi videos.





a) b)

Figure 6. Programmatic PrintScreen functions

a) PrintScreen function that uses ScreenCapture.ocx

b) PrintScreen function that presses the PrintScreen key programmatically

4. Conclusions
The goal of this paper is to present means of interaction and data exchange between two programming technologies: LabVIEW and ActiveX.

The LabVIEW environment itself provides little if any possibilities for video streaming, and the methods it does offer are usually not high performance, but it does offer high performance toolkits and modules specialized in image processing. Hence, we chose to use existing software, specialized in video streaming along with LabVIEW and to capture the data provided by them, for further use, within LabVIEW. The software we studied (the ActiveX controls of a series of media players that utilize streaming technology) provide high quality data and a very small transmission delay, ensuring the reliability of the results of the image processing.

References
[1] Ewald H and Page G F 2000 Performing Experiments by Remote Control Using the Internet, *Global Journal of Engineering Education* **4**(3) 287-292
[2] Krehbiel D, Zerger R and Piper J K 2003 A Remote-Access LabVIEW-based Laboratory for Environmental and Ecological Science, International Journal of Engineering Education **19**(3) 495-502
[3] Chen Z J, Lin C and Wei X G 2009 Enabling on-demand Internet Video Streaming Services to Multi-terminal Users in Large Scale, *IEEE Transactions on Consumer Electronics* **55**(4) 1988-1996
[4] Bermudez I, Mellia M and Meo M 2011 Investigating Overlay Topologies and Dynamics of P2P-TV Systems: The Case of SopCast, *IEEE Journal on Selected Areas in Communications* **29**(9) 1863-1871
[5] Kiong T K, Rong W, Htet K K, Narayanan A S and Chung C K 2013 *Adaptive Streaming Framework for Control and Monitoring Applications*, Proceedings of 2013 IEEE International Conference of Mechatronics and Automation, Takamatsu, Japan, August 4-7, pp 207 – 212
[6] Tanwer A and Singh P R 2010 *ReelEffects Of Threshold Of Hard Cut Based Technique For Advertisement Detection In TV Video Streams*, Proceedings of the 2010 IEEE Students' Technology Symposium, IIT Kharagpur, India, April 3-4, pp 211-216
[7] http://www.ni.com/white-paper/2983/en/
[8] http://www.ni.com/vision/
[9] http://www.sopcast.com/
[10] http://en.wikipedia.org/wiki/VLC_media_player
[11] https://decibel.ni.com/content/docs/DOC-24114
[12] http://help.adobe.com/en_US/Director/11.0/help.html?content=25_using_shockwave_01.html
[13] https://decibel.ni.com/content/docs/DOC-23213