Raspberry pi as a Video Server

1st Fatma Salih
dept. Computer Engineering
University of Gezira
Wad Madani, Sudan
fsalih@hotmail.co.uk

2nd Mysoon S.A. Omer
dept. Computer Engineering
University of Gezira
Wad Madani, Sudan
misoon.siddig4uofg@gmail.com

***Abstract*— The Use of Raspberry pi microcomputer nowadays is increasing rapidly in various applications and projects that require an acceptable software capability and an affordable one. One of the Raspberry pi applications is to use it as a video server. Real-time video servers are used in monitoring, as well as one-sided educational conferences. In this paper, an embedded LAN live real-time video/audio stream server is designed using the Raspberry pi programming and control capabilities. The video is captured through the Raspberry pi camera module port and is compressed and sent using a special standard that applies HTTP so that it can be received from the network. The audio is also captured using a microphone and sent with the RTP protocol. This system finds somehow solutions for high cost and complex video server systems as well as the mass storage used for such systems. This system uses a video stream of 800 × 400 at 24 frames per second. This paper recommends expanding the project to a WAN network.**

***Keywords*— Raspberry pi, Raspbian, HTTP, RTP, Microcomputer**

I. INTRODUCTION

Video Servers are hardware units that apply processing, storing and delivering video data to clients through a medium. A video server uses a specific technology to deal with video data; these technologies include codecs (compression decoder), sending standards and streaming and real-time protocols [1]. In this paper, a Raspberry pi camera module is used to capture video data, knowing that Raspberry pi modules do not support audio capturing unlike webcams, but the main advantages are that modules generate HD videos and they have super compatibility with the Raspberry pi. To solve the audio hitch this paper demonstrates that the audio data are sent using a separate port after connecting audio input devices to capture audio data. Because the Raspberry pi as a hardware unit does not support sending two different types of data using the same port, each of the video/audio is sent separately. Using Raspberry pi microcomputer lowers the cost, simplifies the process, and diminishes programming needs as well as its easy and enjoyable to learn and use.

II. RELATED WORK

There are several studies that used Raspberry pi for video applications. One of those papers used Raspberry pi to implement a monitoring system using a USB video camera that captures video data and the system processes, compresses and sends the data to a mobile client through a wireless network [2].

Another research managed to describe a low-cost monitoring system with motion detection written python code, the motion detection algorithm was used to decrease their storage usage and save their investment costs. The system was implemented using Raspberry pi which enables live video streaming that can be viewed in any web browser including mobile in real-time [3].

Another paper to previous works is dealing with the delay of the video streaming between Raspberry pi devices using the internet of things (IoT). Their implementation transfers a 6 Mbps H.264 video stream of 1280 × 720 pixels at 25 frames per second [4].

Other studies focussed on video surveillance configuration on live streaming and camera monitoring through Raspberry pi in a way were it can be used in user mobile environments from all over the world not just the local areas as home monitoring systems. This will help rescue sensitive areas or areas which are beyond our reach, by accessing images and videos transmitted with the aid of Internet [5].

III. METHODOLOGY

The system is an embedded system that achieves video/audio data capturing, compressing and sending to other devices connected to the network when they contact the server using the special standard chosen through any application that supports network streaming. The process of this system can be divided into two parts, audio serving and video serving. This division stems from the design of the Raspberry pi that it isn't capable of sending two different inputs through the same port, therefore two separate ports are used, the camera module port and the USB audio input card port. The sending process is implemented using Raspbian terminal and video parameters. Servers' processes are as follows:

A. Audio Server:

To design an audio server there must be a capturing device and the Raspberry pi used in this system does not have an audio input port, so to achieve this part of the system an external audio card or convertor was used to help capture audio data through USB. After the capturing process the audio is then streamed live using RTP (Real-Time Transport Protocol) protocol.

B. Video Server:

The Video serving process also starts when the Raspberry pi camera module capturing takes place. The operation of the video sending uses a different standard than that used in the audio; the main protocol used is HTTP (Hypertext Transfer Protocol), also some of the video parameters were changed to match the user's needs. The video stream used was 800 × 400 at 24 frames per second.

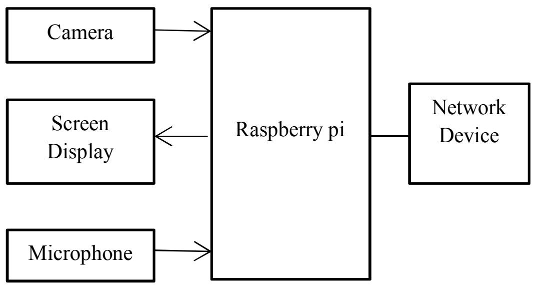


Fig. 1. Block Diagram

The steps followed by this system:

1. The Raspberry pi gets connected to the network. This step is very important because the network plays the main role for clients receiving.

2. Audio input starts functioning.

3. Audio from the Raspberry pi is processed and sent to clients.

4. The connection of the video devices takes place.

5. The camera starts capturing and sends the live video stream.

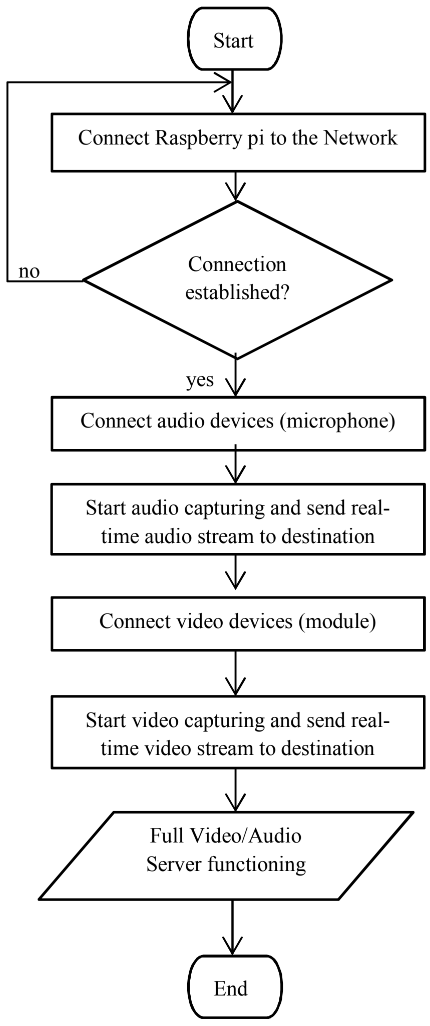

Start
Connect Raspberry pi to the Network
Connection established?
no
yes
Connect audio devices (microphone)
Start audio capturing and send real-time audio stream to destination
Connect video devices (module)
Start video capturing and send real-time video stream to destination
Full Video/Audio Server functioning
End

Fig. 2. Operational Flow Chart

Note: The individual sending of the video/audio produces two separate inputs at the destination. To combine those different types of data, two windows of the network stream application are opened, the audio protocol and port are used in one, and the video parameters are used in the other. Playing the audio at the background window will solve the problem, because it's obvious you don't need to see the audio to be able to hear it, while the video plays in the front. This process results in a full audio/video reception by clients in the network.

This system used software and hardware components as follows:

A. Software used:

1) *Raspbian OS system*:

Raspbian is an operating system based on Debian optimized for the Raspberry pi hardware. An operating system is the set of basic programs and utilities that make the Raspberry pi microcontroller run. Raspbian provides more than a pure OS: it comes with over 35,000 packages; pre-compiled software bundled in a nice format for easy installation on the Raspberry pi [6].

This system used Raspbian stretch, the new Raspbian released 2017.

This Operating system was chosen because it's the very compatible OS with Raspberry pi.

2) *VLC media player:*

VLC media player is a free and open-source, portable media player developed by the VideoLAN project [7]. It also supports network streaming, and supports separate video and audio ports streaming at the same time.

B. *Hardware used:*

1) *Raspberry pi 3 model B*: A microcomputer credit-card sized chip to implement the codes and run the system (server). The selection of this device came from its low cost and high ability of programing and dealing with various hardware and software applications, and also its simplicity.
2) *Screen display*: To deal with the Raspberry pi through it.
3) *Raspberry pi Camera module*: A camera designed to suite Raspberry pi. In this project works it was used as the video input, the Raspberry pi camera module was one of the best possible options because it supports HD video quality and does not need memory consuming packages as its already defined and verified by the Raspberry pi.
4) *Microphone*: That works as the audio input device.
5) *External sound card (convertor)*: The use of this hardware component is because the Raspberry pi has no audio input port so to connect the microphone to a USB port we'll need this converter.
6) *Speaker*: the sound output located at the destination (receiver of the audio and video package) to direct the sound (audio signal).
7) *USB power junction*: Power supply for the Raspberry pi.
8) *Mouse*: To deal with the Raspberry pi desktop.
9) Keyboard: Used to write the commands.

IV. IMPLEMENTATION AND RESULTS

A. *Implementation:*

After installing Raspbian OS into the SD card install the card into the Raspberry pi board and update and upgrade the system, enable the camera and install all software required then reboot the system and follow the steps bellow.

First: Initialize the Raspberry pi and connect it to its display, mouse and keyboard.

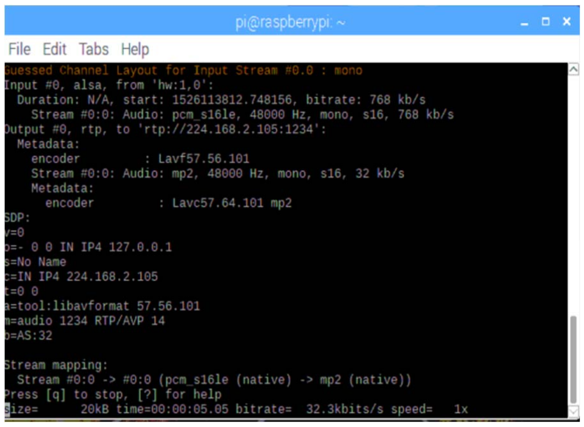
Second: Connect the Raspberry pi to the Network.

Third: Connect all audio devices.

Fourth: write the audio code in terminal.

Fifth: Audio input devices start capturing as soon as the code runs.

The figure below shows the code running and displays some parameters involved in audio sending such as the size and the time counted from when the capturing started.

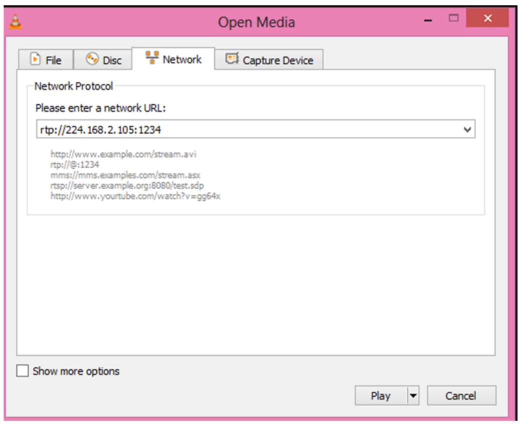


```
uessed Channel Layout for Input Stream #0.0 : mono
Input #0, alsa, from 'hw:1,0':
  Duration: N/A, start: 1526113812.748156, bitrate: 768 kb/s
    Stream #0:0: Audio: pcm_s16le, 48000 Hz, mono, s16, 768 kb/s
Output #0, rtp, to 'rtp://224.168.2.105:1234':
  Metadata:
    encoder         : Lavf57.56.101
    Stream #0:0: Audio: mp2, 48000 Hz, mono, s16, 32 kb/s
    Metadata:
      encoder         : Lavc57.64.101 mp2
SDP:
v=0
o=- 0 0 IN IP4 127.0.0.1
s=No Name
c=IN IP4 224.168.2.105
t=0 0
a=tool:libavformat 57.56.101
m=audio 1234 RTP/AVP 14
b=AS:32

Stream mapping:
  Stream #0:0 -> #0:0 (pcm_s16le (native) -> mp2 (native))
Press [q] to stop, [?] for help
size=      20kB time=00:00:05.05 bitrate=  32.3kbits/s speed=   1x
```

Fig. 3. Audio code Functioning

Sixth: Start VLC media player in the destination device and open Network stream.

Seventh: Insert given IP address and port.



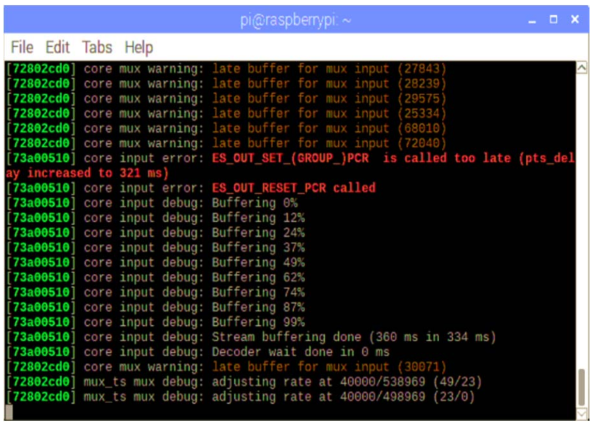
```
rtp://224.168.2.105:1234

http://www.example.com/stream.avi
rtp://@:1234
mms://mms.examples.com/stream.asx
rtsp://server.example.org:8080/test.sdp
http://www.yourtube.com/watch?v=gg64x
```

Fig. 4. IP address and port

Eighth: Connect the video components to the Raspberry pi.

Ninth: write the video code into the terminal.

Similar to the audio, the video code starts functioning, and shows the source's camera view. The figure below shows the screen that displays the video capturing command while running.



```
[72802cd0] core mux warning: late buffer for mux input (27843)
[72802cd0] core mux warning: late buffer for mux input (28239)
[72802cd0] core mux warning: late buffer for mux input (29575)
[72802cd0] core mux warning: late buffer for mux input (25334)
[72802cd0] core mux warning: late buffer for mux input (68010)
[72802cd0] core mux warning: late buffer for mux input (72040)
[73a00510] core input error: ES_OUT_SET_(GROUP_)PCR  is called too late (pts_del
ay increased to 321 ms)
[73a00510] core input error: ES_OUT_RESET_PCR called
[73a00510] core input debug: Buffering 0%
[73a00510] core input debug: Buffering 12%
[73a00510] core input debug: Buffering 24%
[73a00510] core input debug: Buffering 37%
[73a00510] core input debug: Buffering 49%
[73a00510] core input debug: Buffering 62%
[73a00510] core input debug: Buffering 74%
[73a00510] core input debug: Buffering 87%
[73a00510] core input debug: Buffering 99%
[73a00510] core input debug: Stream buffering done (360 ms in 334 ms)
[73a00510] core input debug: Decoder wait done in 0 ms
[72802cd0] core mux warning: late buffer for mux input (30071)
[72802cd0] mux_ts mux debug: adjusting rate at 40000/538969 (49/23)
[72802cd0] mux_ts mux debug: adjusting rate at 40000/498969 (23/0)
```

Fig. 5. Video code Functioning

Tenth: open VLC media player in destination, open network stream and install the IP and port.

Eleventh: video server is now also working.

The two windows have to be opened at the same time so both can work simultaneously.

B. Results:

The system achieved a video and audio server with a very simple method using Raspberry pi. The system used a 16GB memory card and only nearly 8GB was used including the operating system and all software requirements. And the overall cost was approximately $100 divided as follows:

- Raspberry pi board: $25
- Raspberry pi Camera module: $17
- Others: $58 (including screen keyboard and mouse).

The Output of the system fulfilled approximately 80% of the objectives as it was clearly a low cost low storage system, but 30% to 40% of whole video/audio server objectives due to lack of synchronization and delay followed by quality.

V. RECOMMENDATION

Researchers recommend working on the audio-video delay and synchronization, and implementing and expanding this video/audio serving technique to a wide area network sending and receiving (over WAN).

REFERENCES

[1] Video Server, techopedia.com available at http://www.techopedia.com/definition/23872/video-server. Access date 29/5/2018.

[2] Ms. A. Deepa, Ms. R. Dharani, Ms. S. Kalaivani and Ms. P. Manju Parkavi, “Live video streaming system using Raspberry pi with cloud server”. IJAICT, 2016.

[3] Huu-Quoc Nguyen, Ton Thi Kim Loan, Bui Dinh Mao and Eui-Nam Huh, “Low cost real-time system monitoring using Raspberry pi”. IEEE, 2015.

[4] Ulf Jennehag, Stefan Forsstorm and Federico V. Fiordigigli, “Low delay video streaming on the Internet of Things using Raspberry Pi”. Electronics, 2016.

[5] Rhythm Haji, Arjun Trivedi, Hitarth Mehta and Prof. A.B.Upadhyay, “Implementation of Web-Surveillance using Raspberry pi”. IJERT, 2014.

[6] http://Raspbian.org/

[7] http://www.videolan.org/vlc