ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC BÁCH KHOA



MẠNG MÁY TÍNH CO3033



Đề tài:

**A STREAMING VIDEO SERVER AND CLIENT**



**GVHD** : *Hoàng Nguyễn Minh Đức*

**LỚP**  : L04 (tiết 4-6 thứ 2,6)

**NHÓM**  : Nothing

**SV**  :

1. Phan Lê Tuấn Anh 1811442

2. Lê Trung Hiếu 1812167

3. Lê Thị Hà 1812020

4. Lê Phương Khuê 1811015

TP Hồ Chí Minh, Tháng 11/2020

**MỤC LỤC:**

**I. Requirements Analysis……………………………………………...1**

* + 1. **Functional requirements:…………………………………….1**

**2. Non-functional requirements:……………………………….2**

**II. Function description……………………………………………….2**

1. **Client.py……………………………………………………......2**
2. **RtpPacket.py………………………………………………......5**

**III. Class Diagram…………………………………………………….8**

**IV. A summative evaluation of achieved results…………………….9**

**V. User manual……………………………………………………......9**

**VI. Extend……………………………………………………………13**

**1. Yêu cầu………………………………………………………13**

**2. Kết quả hiện thực và nhận xét…………………………….13**

**3. Giải thích……………………………………………………18**

**DANH MỤC THAM KHẢO………………………………………19**

1. **Requirements Analysis:**
   * 1. **Functional requirements:**

* Phần mềm máy khách (media player, web browser, ...) cần kết nối được và xác định file video trên máy streaming server muốn xem
* Yêu cầu streaming file video đó sẽ được gửi tới streaming server để tìm file video đó.
* Chương trình thực hiện streaming chạy trên máy streaming server sẽ chia file video thành các frame rồi gửi các frame đó tới máy yêu cầu sử dụng các giao thức ràng buộc về thời gian (RTSP, RTP).
* Máy client gửi yêu cầu SETUP tới máy server để bắt đầu thực hiện streaming.
* Một yêu cầu SETUP sẽ chỉ ra cách mà một dòng dữ liệu bắt buộc phải được truyền đi như thế nào.
* Yêu cầu SETUP bắt buộc phải được hoàn thành trước khi một yêu cầu PLAY được gửi từ máy client. Máy server sẽ đáp trả lại bằng các xác nhận các tham số đã được lựa chọn, và điền vào các phần còn thiếu.
* Máy client sẽ gửi yêu cầu PLAY để thực hiện truyền các frame dữ liệu thật sự từ máy server tới máy client, và các frame dữ liệu này sẽ được lưu trong một bộ đệm của máy client, các frame này sẽ được giải mã (decode ). Nếu như luồng dữ liệu có bị tạm dừng (pause) thì luồng dữ liệu này cũng sẽ được phục hồi ở frame mà nó tạm dừng truyền.
* Trong quá trình streaming video, nếu như người dùng muốn tạm dừng quá trình streaming thì sẽ gửi yêu cầu PAUSE tới máy server, yêu cầu này sẽ làm tạm dừng một hay nhiều luồng dữ liệu đang truyền các frame về máy client. Máy server sẽ tạm dừng gửi các frame dữ liệu tới máy client.
* Trong quá trình streaming video, nếu như người dùng muốn dừng hẳn quá trình streaming thì sẽ gửi yêu cầu TEARDOWN để dừng truyền và kết thúc một phiên giao dịch của giao tức RTSP. Máy server sẽ đáp trả lại thông điệp xác nhận cho yêu cầu TEARDOWN và sẽ dừng gửi các frame tới máy client.

**2. Non-functional requirements:**

* Tốc độ phản hồi các nút bấm ngay lập tức
* Chất lượng hình ảnh ổn định
* Việc streaming phải được thực hiện trên một platform đơn lẻ, nhưng có thể thực hiện thông qua các thiết bị hoặc máy chủ riêng biệt

1. **Function description**

**1. Client.py:**

* createWidgets

*Chức năng:*

Create Setup button, Create Play button, Create Pause button, Create Teardown button, Create a label to display the movie

* setupMovie

*Chức năng*: xử lý setup button

* exitClient

Chức năng: xử lí teardown button

* pauseMovie

**Chức năng**: xử lý pause button

* playMovie

**Chức năng**: xử lý play button

* writeFrame

**Chức năng**: ghi các frame nhận được vào thư mục ảnh tạm, rồi trả về thư mục ảnh đó

* updateMovie

*Chức năng*: cập nhật file ảnh như là các frame của video trong GUI

* connectToServer

*Chức năng*: kết nối đến server và khởi tạo một giao thức RTSP/TCP mới

* sendRtspRequest

Tham số truyền vào: requestCode

*Chức năng*: gửi yêu cầu RTSP đến server

Thiết lập yêu cầu:

**if** requestCode **==** self.SETUP **and** self.state **==** self.INIT:

            threading.Thread(target**=**self.recvRtspReply).start()

Cập nhật RTSP sequence number

self.rtspSeq **+=** 1

Ghi yêu cầu RTSP để được gửi đi

request **=** 'SETUP ' **+** self.fileName **+** ' RTSP/1.0\nCSeq: ' **+** str(

                self.rtspSeq) **+** '\nTransport: RTP/UDP; client\_port= ' **+** str(self.rtpPort)

Theo dõi yêu cầu được gửi

self.requestSent **=** self.SETUP

Play request

**elif** requestCode **==** self.PLAY **and** self.state **==** self.READY:

Cập nhật RTSP sequence number

self.rtspSeq **+=** 1

Ghi yêu cầu RTSP để được gửi đi

request **=** 'PLAY ' **+** self.fileName **+** ' RTSP/1.0\nCSeq: ' **+** str(self.rtspSeq) **+** '\nSession: ' **+** str(self.sessionId)

Theo dõi yêu cầu được gửi

self.requestSent **=** self.PLAY

Tương tự cho pause, teardown request.

* recvRtspReply

*Chức năng*: nhận phản hồi RTSP từ server

* parseRtspReply

Tham số đầu vào: data

*Chức năng*: phân tích phản hồi từ RTSP từ server

Thực hiện chỉ khi sequence number của phản hồi từ server giống với của request

**if** seqNum **==** self.rtspSeq:

            session **=** int(lines[2].split(' ')[1])

Tạo mới session ID của RTSP

**if** self.sessionId **==** 0:

                self.sessionId **=** session

Thực thi chỉ khi session ID giống nhau

**if** self.sessionId **==** session:

Cập nhật trạng thái READY cho RTSP

self.state **=** self.READY

Mở cổng RTP

self.openRtpPort()

Cập nhật trạng thái PLAYING cho RTSP

self.state **=** self.PLAYING

Luồng chạy thoát, một luông mới được tạo ra

self.playEvent.set()

Bật cờ của teardownAcked để đóng socket

self.teardownAcked **=** 1

* openRtpPort

*Chức năng*: mở RTP socket được liên kết đến cổng được chỉ định

Tạo một datagram socket mới để nhận gói RTP từ server

self.rtpSocket **=** socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

Đặt giá trị thời gian kết thúc là 0.5s

self.rtpSocket.settimeout(0.5)

Liên kết socket với địa chỉ sử dụng cổng RTP được cho bởi client

self.rtpSocket.bind(("", self.rtpPort))

* handler

Chức năng: xử lý việc đóng cửa sổ GUI một cách rõ ràng.

**2. RtpPacket.py**

* encode

Tham số đầu vào: version, padding, extension, cc, seqnum, maker, pt, ssrc, payload

*Chức năng*: mã hóa gói RTP với những header field và payload

Điền mảng dữ liệu của header với những mảng dữ liệu của header với RTP

header[0] **=** (header[0] **|** version **<<** 6) **&** **0x**C0;  # 2 bits

        header[0] **=** (header[0] **|** padding **<<** 5);  # 1 bit

        header[0] **=** (header[0] **|** extension **<<** 4);  # 1 bit

        header[0] **=** (header[0] **|** (cc **&** **0x**0F));  # 4 bits

        header[1] **=** (header[1] **|** marker **<<** 7);  # 1 bit

        header[1] **=** (header[1] **|** (pt **&** **0x**7f));  # 7 bits

        header[2] **=** (seqnum **&** **0x**FF00) **>>** 8;  # 16 bits total, this is first 8

        header[3] **=** (seqnum **&** **0x**FF);  # second 8

        header[4] **=** (timestamp **>>** 24);  # 32 bit timestamp

        header[5] **=** (timestamp **>>** 16) **&** **0x**FF;

        header[6] **=** (timestamp **>>** 8) **&** **0x**FF;

        header[7] **=** (timestamp **&** **0x**FF);

        header[8] **=** (ssrc **>>** 24);  # 32 bit ssrc

        header[9] **=** (ssrc **>>** 16) **&** **0x**FF;

        header[10] **=** (ssrc **>>** 8) **&** **0x**FF;

        header[11] **=** ssrc **&** **0x**FF

Thiết đặt RTP’s header và payload

self.header **=** header

Lấy payload từ argument

self.payload **=** payload

* decode

Tham số đầu vào: byteStream

Chức năng: giải mã gói RTP

* version

*Chức năng*: trả về phiên bản RTP

* seqNum

*Chức năng*: trả về sequence number

* timeStamp

*Chức năng*: trả timestamp

* payloadType

*Chức năng*: trả về kiểu payload

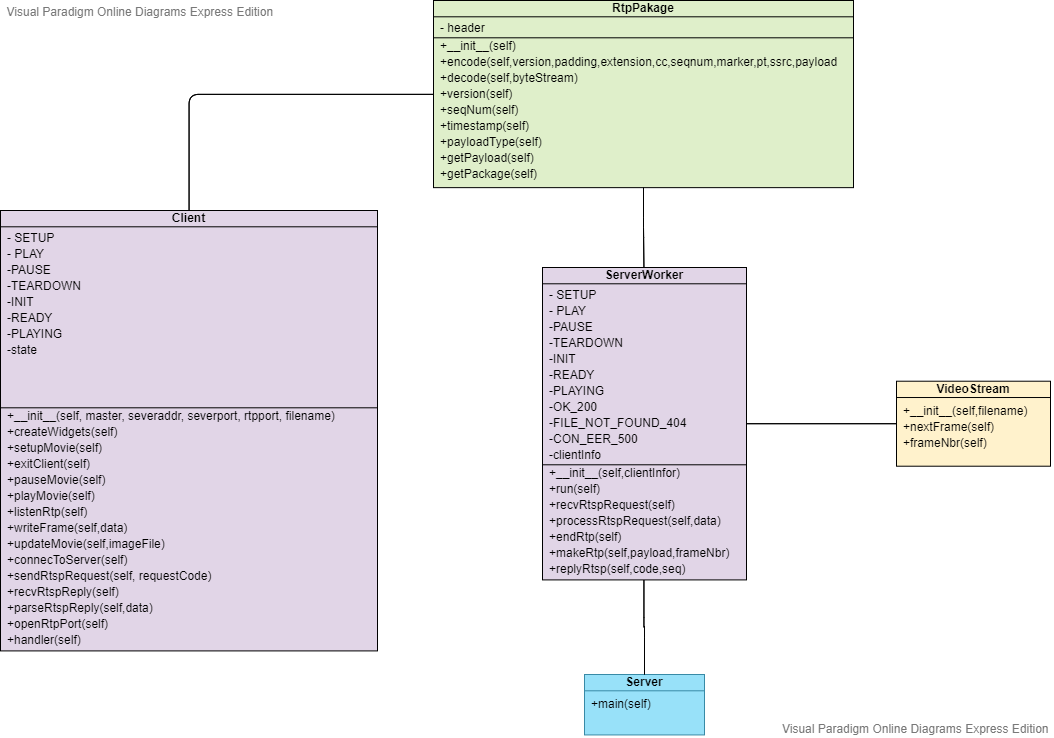
* getPayload

*Chức năng*: trả về payload

* getPacket

*Chức năng*: trả về gói RTP

**III.. Class diagram:**

****

*Hình 1: Class Diagram*

***Mô tả:***

Đây là class diagram cho hiện thực streaming video client và server bằng Real-Time Streaming Protocol (RTSP) và gửi data bằng Real-time Transfer Protocol (RTP). Sơ đồ gồm 5 class: ServerWorker, Client, VideoStream, Server, RtpPakage, giữa các class sử dụng mối liên hệ bằng Association.

**IV. A summative evaluation of achieved results**

- RTP và RTSP được dùng trong truyền tin thời gian thực, giao thức điều khiển truyền thông mạng ở [tầng ứng dụng](https://vi.wikipedia.org/wiki/T%E1%BA%A7ng_%E1%BB%A9ng_d%E1%BB%A5ng) được thiết kế để sử dụng trong các hệ thống giải trí và truyền thông để điều khiển máy chủ chứa các dữ liệu truyền tin đa phương tiện ([streaming media](https://vi.wikipedia.org/wiki/Streaming)).

- Trong bài tập lớn này, chúng em sử dụng giao thức IPv4, TCP cho gói tin RTSP và giao thức UDP cho gói tin dữ liệu RTP (data packets)

- Về phía máy chủ, chúng em mở một socket sử dụng SOCK\_STREAM, đó là một TCP socket. Nhờ có đặc tính tin cậy, TCP socket sẽ được sử dụng để chuyển control requests.

- Chúng em cũng mở một socket khác nhưng nó là SOCK\_DGRAM, cái mà không kết nối. Đó là một UDP socket, chúng em sử dụng socket này để chuyển dữ liệu gói tin của real-time video. Mặc dù đây là một giao thức không tin cậy, nhưng nó có tốc độ nhanh hơn TCP. UDP không đòi hỏi gói tin phải đã được kếu kiểm tra cũng như là resent nếu hư hỏng trước khi xử lí nó.

**V. User manual**

- Đầu tiên chúng ta khởi động server bằng lệnh command line:

**python Server.py <server\_port>**

*Ví dụ :*

python Server.py 1025

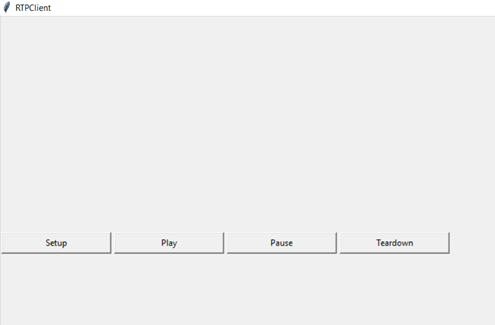
- Sau đó, khởi động client bằng lệnh command line bên dưới.

**python ClientLauncher.py <server\_name> <server\_port> <movie\_file\_name>**

*Ví dụ:*

python ClientLauncher.py 127.0.0.1 1025 5008 movie.Mjpeg

- Sau khi chạy hai lệnh trên chúng ta sẽ thấy xuất hiện cửa sổ như hình bên dưới:

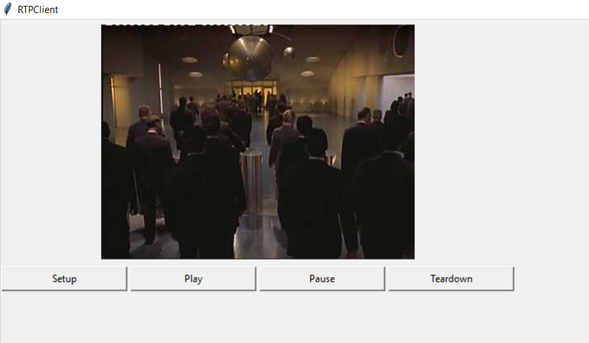


*Hình 2: Giao diện khi chạy xong 2 lệnh command line*

- Click chọn nút “Setup” để tạo kết nối đến server

- Sau đó click nút “Play” để gửi yêu cầu Play movie đến server và phát movie lên màn hình.

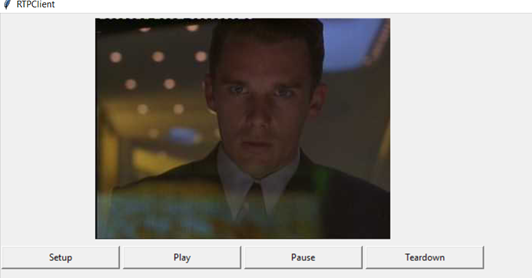
-> Chúng ra sẽ thấy movie được chạy trên màn hình như hình bên dưới:



*Hình 3: Giao diện khi vừa nhấn Play*

- Trong khi movie đang được phát, chúng ta có thể nhấn nút “Pause” để gửi một yêu cầu “Pause” movie đến server và tạm dừng movie lại.

-> Chúng ta sẽ thấy movie được dừng lại như hình bên dưới:

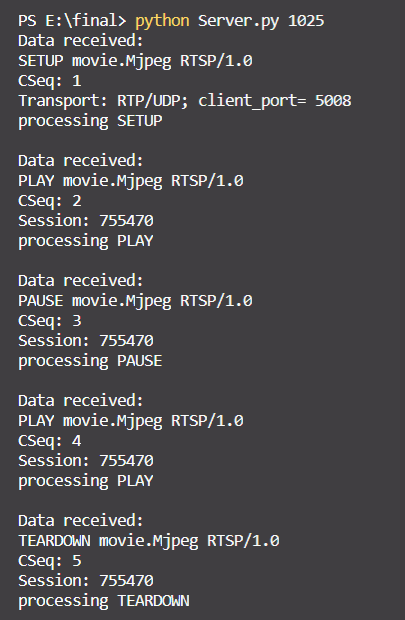


*Hình 4: Giao diện khi vừa nhấn Pause*

- Chúng ta có thể nhấn nút “Play” để tiếp tục xem tiếp movie sau khi nhấn nút “Pause”

- Và cuối cùng khi ta không muốn xem movie nữa, ta nhấn vào nút “Teardown” để stop movie và đóng media player.

- Trong khi media server đang cung cấp kết nối, nó sẽ hiển thị một vài thông tin phản hồi của người dùng trong terminal như hình sau:



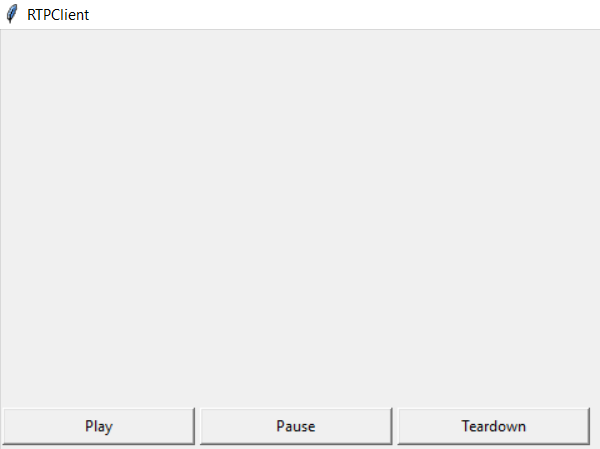
*Hình 5: Thông tin phản hồi cho người dùng*

**VI. Extend:**

**1. Yêu cầu:** The user interface on the RTPClient has 4 buttons for the 4 actions. If you compare this to a standard media player, such as RealPlayer or Windows Media Player, you can see that they have only 3 buttons for the same actions: PLAY, PAUSE, and STOP (roughly corresponding to TEARDOWN). There is no SETUP button available to the user. Given that SETUP is mandatory in an RTSP interaction, how would you implement that in a media player? When does the client send the SETUP? Come up with a solution and implement it. Also, is it appropriate to send TEARDOWN when the user clicks on the STOP button?

**2. Kết quả hiện thực và nhận xét:**

- Sau khi chạy 2 câu lệnh terminal thì chúng ta sẽ thấy xuất hiện cửa sổ như hình bên dưới:

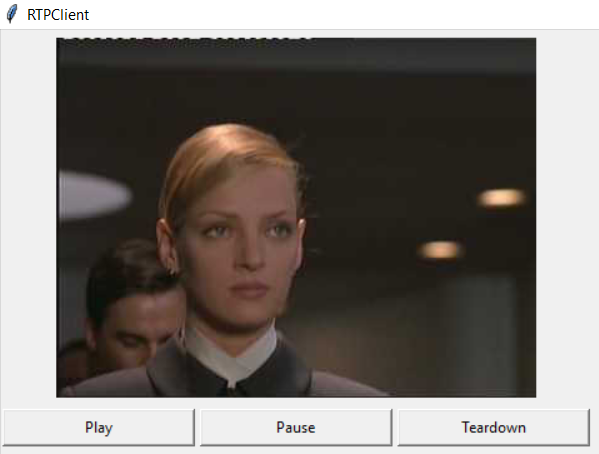
****

*Hình 6: Giao diện khi chạy xong 2 lệnh comamnd line*

- Khác với phần trên chúng ta thấy ở giao diện ban đầu chỉ còn lại 3 nút: Play, Pause và Teardown. Chúng ta không còn sự xuất hiện của nút Setup nữa vì nó đã được tích hợp vào nút Play.

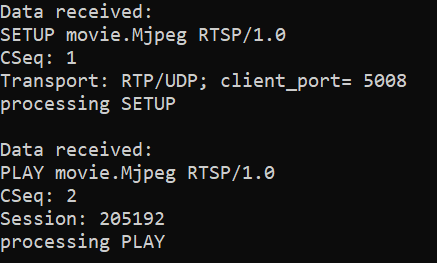
- Khi nhấn Play chúng ta sẽ tạo kết nối đến server đồng thời gửi đến server request Play movie.

- Sau khi nhấn Play ta sẽ thấy movie được phát như hình bên dưới:



*Hình 7: Giao diện sau khi nhấn Play*

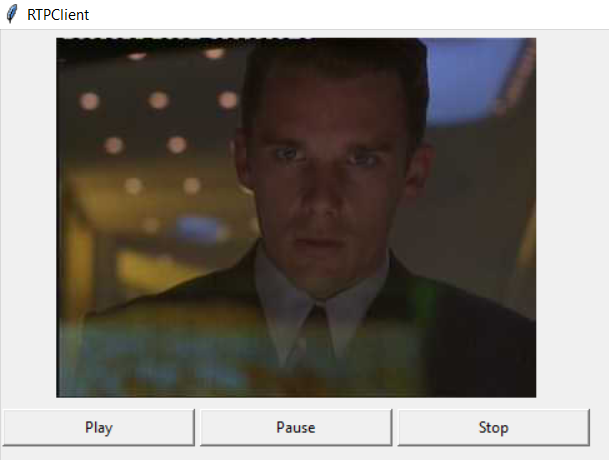
- Trong khi media server đang cung cấp kết nối, nó sẽ hiển thị một vài thông tin phản hồi của người dùng trong terminal như hình sau:



*Hình 8: Phản hồi người dùng khi nhấn nút Play*

Và cũng như chương trình ban đầu, trong khi movie đang được phát, chúng ta có thể nhấn nút “Pause” để gửi một yêu cầu “Pause” movie đến server và tạm dừng movie lại.

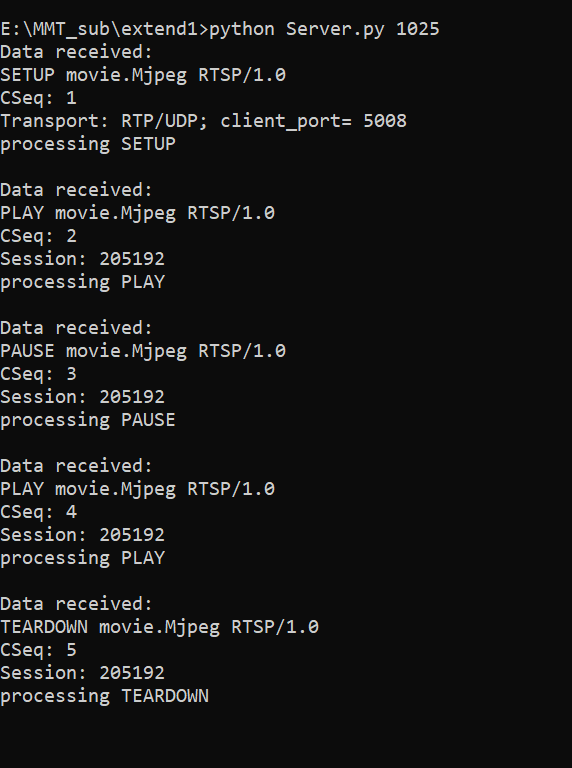
-> Chúng ta sẽ thấy movie được dừng lại như hình bên dưới:



*Hình 9: Giao diện khi nhấn Pause*

- Chúng ta có thể nhấn nút “Play” để tiếp tục xem tiếp movie sau khi nhấn nút “Pause”

- Và cuối cùng khi ta không muốn xem movie nữa, ta nhấn vào nút “Teardown” để stop movie và đóng media player.

- Trong khi media server đang cung cấp kết nối, nó sẽ hiển thị một vài thông tin phản hồi của người dùng trong terminal như hình sau:

*Hình 10:*

*Phản hồi thông tin*

*cho người dùng*

**3. Giải thích:**

Trong các media player không có các nút PLAY mặc dù cũng sử dụng RTSP. Cách hiện thực nút PLAY của media player là thêm trường hợp xét trạng thái nhấn play nếu đang INIT thì gọi request SETUP như nút set up làm, nếu ở READY thì như nút PLAY

**DANH MỤC THAM KHẢO**

[1] <https://realpython.com/python-sockets/>

[2] <https://www.ietf.org/rfc/rfc2326.txt>

[3] <https://stackoverflow.com/questions/478637/teardown-appropriate-in-rtsp>

[4] <https://docs.python.org/3.4/howto/sockets.html>