EMBEDDED CODING VIDEO STREAM DALAM RASPBERRY PI MODEL 2 PADA *DEVICE TO DEVICE COMMUNICATION*

Arisla Choirudin Muzzaki[*)], Aghus Sofwan, and Muhammad Arfan

Departemen Teknik Elektro, Universitas Diponegoro
Jl. Prof. Sudharto, SH, Kampus UNDIP Tembalang, Semarang 50275, Indonesia

[*)]*E-mail: arisla313@gmail.com*

Abstrak

Pemanfaatan Internet of Things (IoT) saat ini sudah semakin meluas. Cakupan pemafaatannya pun semakin beragam, dari hanya pada sektor pendidikan sebagai sarana berkomunikasi hingga pada hal-hal yang berkenaan dengan kebutuhan sehari-hari. Salah satu pemanfaatan IoT yang sering digunakan saat ini adalah layanan video streaming. Saat ini, telah terdapat beberapa cara dan metode konfigurasi untuk membuat sebuah layanan video streaming, salah satunya adalah dengan proses embedded coding. Coding yang ditanamkan ini nantinya yang menjadi pengirim dari proses streaming video. Dengan digunakannya proses embedded coding untuk mendapatkan layanan video streaming, tentunya akan membuat layanan ini dapat diakses dari beberapa software Video Stream Player yang berbeda-beda. Server ini nantinya merupakan program yang dijalankan dalam sebuah mini computer yaitu Raspberry Pi. Dengan adanya Raspberry Pi beserta komponen-komponen pelengkapnya seperti Raspi Camera, akan memperkecil dimensi alat tanpa mengurangi peforma dari sistem yang dijalankan. Penggunaan teknologi video stream dengan menggunakan proses embedded coding diharapkan dapat menjadi sarana baru untuk meningkatkan peforma layanan streaming video dan bisa mengikuti perkembangan di zaman sekarang.

Kata kunci: embedded coding, video stream, raspberry pi

Abstract

Utilization of the Internet of Things (IoT) is increasingly widespread. Usage coverage even more diverse, from just in the education sector as communicating media up to matters relating to daily needs. One of the most commonly used IoT is the streaming video service. Currently, there have been several ways and configuration methods to create a streaming video service, one of method is the embedded coding process. This embedded coding will be the sender of the streaming video. With the use of embedded coding process to get streaming video services, of course will make this service can be accessed from several different Video Stream Player software. This server will be a program that runs in a mini computer that is Raspberry Pi. With the Raspberry Pi and its complement components like Raspi Camera, it will reduce the dimensions of the tool without reducing the performance of the system being run. The use of video stream technology using embedded coding process is expected to be a new tool to improve the performance of video streaming services and can keep up with today's developments.

Keywords: embedded coding, video stream, raspberry pi

1. Pendahuluan

IoT merupakan sebuah konsep dimana suatu objek yang memiliki kemampuan untuk mentransfer data melalui jaringan tanpa memerlukan interaksi manusia ke manusia atau manusia ke komputer. IoT telah berkembang dari konvergensi teknologi nirkabel, *Micro-Electromechanical Systems* (MEMS), dan Internet.

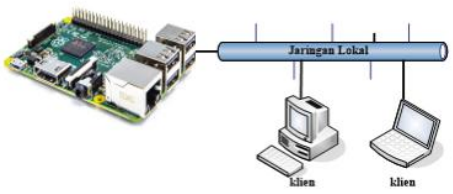
IoT sebagai suatu inovasi teknologi memiliki beberapa contoh, salah satunya adalah *video streaming*. *Video streaming* merupakan sebuah komunikasi yang dilakukan melalui broadcast akses internet untuk menghasilkan sebuah gambar. *Video streaming* bukan hal yang baru bagi kita di tanah air (Indonesia), sejak munculnya 3G (Generasi ke Tiga) pada sebuah telephone seluler, *video streaming* bagaikan jamur bertumbuhan dimana-mana, hingga kepelosok tanah air.[1]

Video streaming sebenarnya sebuah teknologi yang mempermudah kita dalam mendapatkan informasi dalam bentuk tampilan video. *Video streaming* juga sering digunakan sebagai fitur dalam perangkat pemantau jarak jauh, seperti alat pemantau kemacetan jalan, pemantau keamanan rumah, dan lain sebagainya. Layanan *video streaming* ini telah memiliki banyak pengembangan. Mulai

dari pengembangan perangkat kerasnya maupun metodenya, apa perangkat lunak yang digunakan, dan sebagainya. Adapun penggunaan perangkat keras maupun perangkat lunak yang berbeda tentunya memiliki fungsi, kelebihan, dan kekurangan yang berbeda-beda.[2]
Penelitian ini bertujuan untuk membuat layanan *video streaming* pada perangkat Raspberry Pi model 2 dengan proses *embedded coding* yang nantinya *coding* tersebut menjadi sebuah program Video Streaming. Program Video Streaming ini nantinya akan memberi akses kepada perangkat dengan aplikasi *stream player* yang terhubung dengan jaringan Raspberry Pi.[3] Raspberry Pi ini nantinya yang menjadi otak serta pendukung dari berjalannya sistem tersebut. Pada Raspberry Pi ini nantinya juga dilengkapi dengan kamera Raspi sebagai media penangkap video serta *router* sebagai penghubung perangkat Raspberry Pi ke jaringan internet agar hasil dari penangkapan video dapat di lihat di *device* lain. Penggunaan Raspberry Pi sebagai *mini computer* ditujukan untuk mengefisienkan dimensi dari alat yang dibuat. Nantinya alat untuk *streaming video* dengan program Video Streaming ini dapat dimanfaatkan untuk mempermudah pengaksesan layanan *video streaming*[4]

2. Metode

Perancangan sistem ini membutuhkan satu buah Raspberry pi model 2 yang akan digunakan sebagai perangkat dijalankannya *video streaming* yang nantinya juga akan di pasang menggunakan sistem operasi Linux Raspbian.



Gambar 1. Perancangan sistem secara umum

Perangkat Raspberry Pi akan dipasang Camera Raspi dimana kamera ini akan digunakan sebagai penangkap video yang akan dijadikan objek dalam sistem ini. Agar program Video Streaming dapat dijalankan, sebelumnya ada beberapa *library* yang perlu di*install* di dalam OS Raspbian yang digunakan. *Library* tersebut nantinya yang menentukan fungsi-fungsi dalam program Video Streaming bisa dijalankan atau tidak.[5]

Apabila program Video Streaming sudah dijalankan dan HTTP-Server serta Camera Interface sudah berhasil dijalankan, maka video yang ditangkap oleh Camera Raspi sudah bisa ditampilkan di komputer klien melalui jaringan lokal yang dipasang. Adapun cara menampilkan *video streaming* yang dikirimkan ke jaringan lokal tersebut, sesuai dengan langkah-langkah dari *video player* yang digunakan oleh perangkat klien. Selama *video player* tersebut menyediakan layanan streaming, maka klien bisa melakukan pengambilan video yang dikirimkan di jaringan lokal.

2.1. Analisis Kebutuhan

Analisis kebutuhan sistem ini ditujukan untuk menguraikan kebutuhan-kebutuhan yang harus disediakan oleh sistem agar dapat memenuhi kebutuhan pengguna dan sesuai dengan tujuan penelitian yaitu menerapkan program Video Streaming pada perangkat Raspberry pi bersistem operasi Linux Raspbian

2.1.1. Kebutuhan Fungsional

Kebutuhan fungsional merupakan gambaran mengenai fungsi-fungsi yang dapat dilakukan oleh sistem ini. Kebutuhan fungsional sistem meliputi:

a. Sistem yang dirancang memberikan hak akses kepada pengguna yang terhubung ke jaringan lokal.
b. Adanya fasilitas untuk mengubah kamera dalam *mode sleep* bila tidak ada klien yang terhubung ke program Video Streaming.
c. Adanya fasilitas untuk bisa mendapatkan layanan *video streaming* langsung saat Raspberry Pi dihidupkan.

2.1.2. Kebutuhan Non Fungsional

Kebutuhan non-fungsional ini melingkupi beberapa kebutuhan yang mendukung kebutuhan fungsional, rumusan kebutuhan non-fungsional meliputi:

a. Kebutuhan Operasional
   - Program dapat dijalankan di perangkat Raspberry Pi.
   - Program hanya dapat diakses oleh perangkat lain yang terhubung di jaringan yang sama dengan jaringan yang digunakan Raspberry Pi.
   - *User interface* pada aplikasi *video stream player*, sesuai dengan aplikasi yang digunakan oleh pengguna.

b. Performansi Sistem
   Sistem yang dibangun merupakan program yang ditanamkan dalam perangkat Raspberry Pi yang berjalan pada lingkungan tertentu. Terdapat beberapa keterbatasan yang ditemui pada perangkat yang dijalankan pada lingkungan berbatas tersebut, meskipun perangkat serta sistem operasi yang digunakan sudah memadai yaitu Raspberry Pi. Oleh karena itu perlu diperhatikan guna menjadi acuan dalam pengembangan sistem, diantaranya:
   - Sumber daya bergantung pada listrik DC PLN tanpa ada backup baterai. Dengan kata lain, apabila *supply* listrik terputus, alat akan langsung mati. Apabila perangkat Raspberry Pi mati, program juga akan langsung menghentikan proses pengiriman video ke jaringan.

- Tampilan aplikasi antarmuka serta langkah pengambilan video dari program Video Streaming, sepenuhnya bergantung pada aplikasi yang digunakan. Ini membuat tidak adanya pedoman secara universal apabila terjadi kesalahan/error pada video stream player.

Dari keterbatasan pada sistem yang dikembangkan tersebut, maka diusulkan beberapa alternatif untuk menunjang performa sistem dengan keterbatasan yang ada, diantaranya:

- Merancang sebuah sistem yang bisa dengan otomatis menjalankan program saat perangkat Raspberry Pi kembali menyala setelah mati.
- Memilih aplikasi *video stream player* dengan antarmuka yang sederhana namun tetap menarik dan mudah digunakan oleh pengguna.

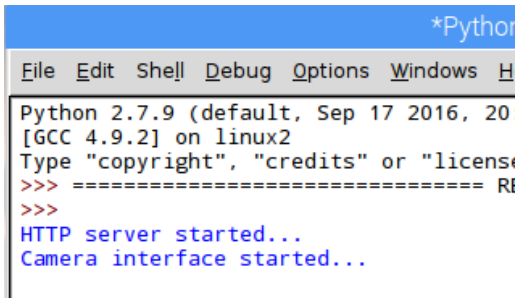
3. Hasil dan Analisa

3.1. Pengujian *Running Coding* Program Video Streaming

Pengujian ini akan dibagi dalam 2 jenis pengujian, yaitu pengujian tanpa *autostart* dan dengan *autostart.* Pengujian tanpa *autostart* adalah dengan menjalankan program secara *manual*. Cara menjalankan program secara *manual* adalah dengan membuka program yang sudah dibuat dengan IDLE Python 2, lalu *run* program tersebut dengan fungsi *run* yang terdapat pada Python 2.

Pengujian dengan fungsi *autostart* berarti kita tidak perlu membuka dan menjalankan program secara *manual* seperti pada pengujian sebelumnya. Perintah untuk membuka program Video Streaming yang sudah penulis letakkan pada file *custom.sh* sebagai file *autostart* yang digunakan akan langsung dieksekusi oleh sistem sesaat setelah sistem berjalan.

Apabila program berhasil dijalankan, akan memunculkan tulisan *HTTP Server started* dan *camera interface started* pada *shell* IDLE Python 2.



Gambar 2. Tampilan program saat berhasil dijalankan

Berikut ini adalah perbandingan dan analisa terhadap seluruh hasil pengujian dari 1 perangkat klien dan 2 perangkat klien dengan *stream player* yang berbeda..

Perbandingan pertama merupakan perbandingan penggunaan memori *virtual* Raspberry Pi pada 1 perangkat dan 2 perangkat.

Tabel 1 Hasil pengujian penggunaan memori *virtual*

	Penggunaan memori *virtual*
Tanpa fungsi *autostart*	
Media Player Classic	154636 KiB
VLC Player	136204 KiB
MX Player	154636 KiB
Dengan fungsi *autostart*	
Media Player Classic	118848 KiB
VLC Player	118848 KiB
MX Player	128064 KiB
2 Perangkat	
Media Player Classic VLC Player	137280 KiB
Media Player Classic MX Player	137280 KiB
VLC Player MX Player	137280 KiB

Tabel 1 menunjukkan perbedaan penggunaan memori pada masing-masing pengujian. Pada pengujian dengan 1 perangkat, VLC Player menunjukkan konsumsi memori *virtual* yang lebih sedikit dibandingkan dengan Media Player Classic dan MX Player. Pada pengujian 2 *stream player* secara bersamaan menunjukkan penggunaan memori *virtual* yang sama. Hal ini dikarenakan adanya perpaduan antara *client buffer* dari masing-masing *stream player* yang membuat penggunaan memori *virtual* menjadi sama pada ketiga kombinasi *stream player* yang berbeda.
Selanjutnya adalah perbandingan data hasil pengujian waktu *delay* perangkat dalam mengakses program Video Streaming. Pada Tabel 4.5 ini merupakan hasil rekap dari data hasil pengujian waktu *delay*

Tabel 2. Perbandingan hasil pengujian waktu *delay*

	Waktu *delay*		
	Pengujian I	Pengujian II	Pengujian III
Tanpa fungsi *autostart*			
Media Player Classic		7.88 detik	
VLC Player		1.78 detik	
MX Player		3.28 detik	
Dengan fungsi *autostart*			
Media Player Classic		18.98 detik	
VLC Player		2.03 detik	
MX Player		7.35 detik	
2 Perangkat			
Media Player Classic	00.67 detik	00.86 detik	01.40 detik
VLC Player	01.03 detik	00.86 detik	01.52 detik
Media Player Classic	00.90 detik	00.59 detik	00.90 detik
MX Player	00.90 detik	00.59 detik	02.15 detik
VLC Player	01.20 detik	01.21 detik	01.13 detik
MX Player	01.00 detik	01.21 detik	00.98 detik

Pada Tabel 4.5 menunjukkan bahwa peforma yang berbeda dari aplikasi *video stream player* yang digunakan. Dari segi rerataan waktu *delay*, Media Player Classic paling lama dibandingkan kedua aplikasi yang lain, bahkan pada pengujian dengan fungsi *autostart*, rerataan waktu *delay* untuk Media Player Classic cukup lama (18.98 detik)

dibandingkan dengan kedua perangkat yang lain yang rata-rata dibawah 10 detik.
Apabila menggunakan 2 perangkat yang berbeda dengan aplikasi *video stream player* yang berbeda pula, waktu *delay* yang muncul berbeda antara satu dengan lain. Pada tabel nampak bahwa tidak adanya keteraturan perubahan data waktu *delay* pada pengujian I, II maupun III di ketiga kombinasi aplikasi yang digunakan. Hal ini menunjukkan bahwa dengan menggunakan 2 perangkat yang berbeda & aplikasi *video stream player* yang berbeda akan membuat waktu *delay* yang tidak beraturan di kedua perangkat tersebut.

3.2. Pengujian *Multiclient*

Pengujian *multiclient* ini bertujuan untuk mengetahui seberapa banyak perangkat yang dapat mengakses layanan ini secara bersamaan dengan hasil layanan yang optimal. Pada pengujian ini pula akan dianalisa seberapa banyak penggunaan *virtual memory* Raspberry Pi yang digunakan saat ada beberapa pengguna yang menggunakan perangkat secara bersamaan

Tabel 3. Hasil pengiriman data pengujian *multliclient*

	Hasil Pengiriman Data
2 Perangkat	Lancar
4 Perangkat	Lancar
6 Perangkat	Urutan pengiriman data ke klien tidak teratur Semua data tersampaikan ke pihak klien
8 Perangkat	Urutan pengiriman data ke klien tidak teratur Adanya saling blocking pada proses transmisi data Beberapa perangkat memiliki waktu *delay* yang lebih lama
10 Perangkat	Urutan pengiriman data ke klien tidak teratur Adanya saling blocking pada proses transmisi data Beberapa perangkat memiliki waktu *delay* yang lebih lama

Pada tabel diatas, dapat disimpulkan bahwa layanan *video streaming* ini dapat melayani hingga 10 perangkat secara bersamaan dengan kelemahan, di antaranya ketidak seimbangan waktu *delay* antar perangkat yang disebabkan oleh ketidak teraturan urutan data yang dikirim ke tiap-tiap perangkat pengguna. Penggunaan fungsi *multiclient* yang paling efektif tanpa adanya gangguan yang berarti itu berada pada penggunaan 4 perangkat pengguna secara bersamaan.

Tabel 4. Perbandingan hasil penggunaan *virtual memory* Raspberry Pi pada pengujian *multliclient*

	Penggunaan *virtual memory* Raspberry Pi
2 Perangkat	137280 KiB
4 Perangkat	161804 KiB
6 Perangkat	178188 KiB
8 Perangkat	220428 KiB
10 Perangkat	220428 KiB

Pada tabel di atas dapat disimpulkan bahwa semakin bertambahnya perangkat yang mengakses layanan *video streaming* ini, maka akan semakin besar pula penggunaan *virtual memory* pada Raspberry Pi.

3.3. Pengujian *Framerate*

Framerate merupakan total *frame* yang dikirim oleh layanan *video streaming* ini per detik. Pada pengujian *framerate* ini, akan dilakukan perbandingan hasil uji coba modifikasi *framerate* pada layanan *video streaming*. Pada pengujian kali ini, akan digunakan variasi *framerate* 1, 4, 6, 8 dan 10 fps. Pemilihan variasi *framerate* ini ditujukan untuk bisa menentukan *framerate* mana yang lebih efektif dan mencukupi untuk diterapkan pada layanan video streaming ini.

Tabel 5. Hasil layanan *video streaming* dari sisi pengguna pada pengujian *framerate*

	Hasil Layanan Video Streaming
***Framerate* 1 fps**	Hasil *video streaming* tidak muncul
***Framerate* 4 fps**	Tidak *smooth*
***Framerate* 6 fps**	Cukup *smooth*
***Framerate* 8 fps**	Baik (*smooth)*
***Framerate* 10 fps**	Baik (*smooth)*

Pada tabel diatas, dapat disimpulkan bahwa layanan *video streaming* ini dapat memberikan hasil yang baik, minimal dengan *framerate* 8 fps. Pada batas minimal ini, layanan sudah bisa memberikan hasil yang baik tanpa memperlihatkan gangguan layanan yang berarti.

Tabel 6. Hasil penggunaan *virtual memory* Raspberry Pi pada pengujian *framerate*

	Penggunaan *virtual memory* Raspberry Pi
***Framerate* 1 fps**	132108 KiB
***Framerate* 4 fps**	135180 KiB
***Framerate* 6 fps**	135180 KiB
***Framerate* 8 fps**	135180 KiB
***Framerate* 10 fps**	135180 KiB

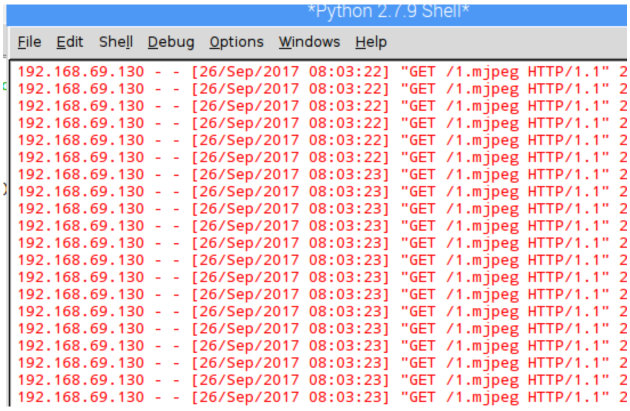
Penggunaan *virtual memory* Raspberry Pi dengan variasi *framerate* akan berbeda apabila layanan tersebut tidak memunculkan hasil sebagaimana mestinya. Pada pengujian dengan pengaturan *framerate* 1 fps menggunakan *virtual memory* Raspberry Pi yang berbeda dibanding pengujian yang lain karena hanya pada pengujian tersebut dimana hasil pengujiannya tidak menampilkan hasil pada sisi klien.

3.4. Pengujian Performa Sistem

Pada pengujian ini, akan menggunakan 2 jenis pengujian yaitu ketahanan layanan pada lingkungan dengan koneksi tidak stabil dan ketahanan layanan pada penggunaan dengan waktu yang lama. Kedua parameter tersebut dipilih karena 2 kejadian tersebut yang mungkin terjadi pada penerapan layanan ini di lingkungan nyata.

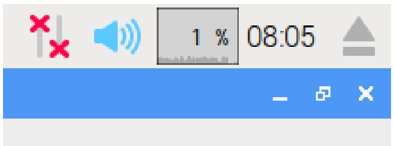
3.4.1. Ketahanan Layanan Pada Lingkungan Dengan Koneksi Tidak Stabil

Pada pengujian ketahanan layanan dengan diberikan simulasi gangguan berupa koneksi yang tidak stabil. Pada penelitian ini, akan diujikan bagaimana pengaruh pada layanan apabila terjadi gangguan koneksi dan apa yang terjadi apabila koneksi tersebut terhubung kembali.



Gambar 3. Hasil pengiriman data layanan *video streaming*

Saat program sedang berjalan dan layanan mengirimkan *frame* gambar ke klien, lakukan pemutusan jaringan internet yang berfungsi untuk memberikan gangguan koneksi. Pada Raspberry Pi, pemutusan jaringan internet dapat dilihat pada tampilan Raspberry Pi yang menunjukkan perangkat tidak terhubung ke jaringan.



Gambar 4. Indikator Raspberry Pi saat tidak terhubung ke jaringan

Pada Raspberry Pi, pemutusan jaringan internet dapat dilihat pada tampilan Raspberry Pi yang menunjukkan perangkat tidak terhubung ke jaringan. Indikator dari Raspberry Pi saat tidak terhubung ke jaringan

Gambar 5. Hasil pengiriman data saat terjadi gangguan koneksi

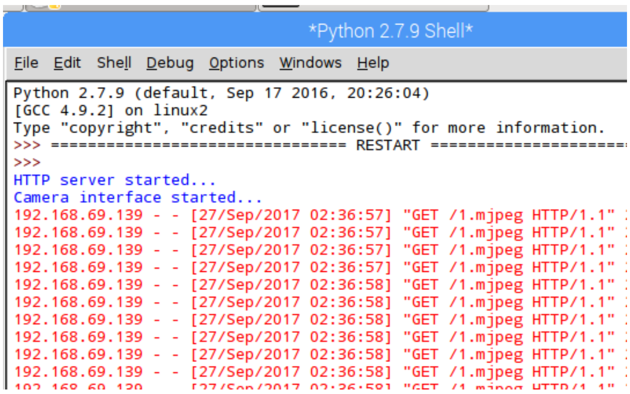
Saat terjadi gangguan koneksi pada perangkat, program Video Streaming akan langsung berhenti mengirimkan data *frame* gambar ke perangkat klien. Pada pengujian kali ini, pengiriman data *frame* berhenti pada tanggal 26 September 2017 jam 08:10:17.

Gambar 6. Hasil pengiriman data saat koneksi dapat terhubung lagi

Dapat dilihat bahwa program Video Streaming yang dibuat ini dapat melanjutkan pengiriman data *frame* yang sebelumnya berhenti saat terjadi gangguan koneksi. Hal ini ditunjukkan pada dilanjutkannya pengiriman data yang sebelumnya berhenti pada jam 08:10:17 dilanjutkan pada saat koneksi kembali terhubung pada jam 08:12:10. Pada pengujian ini dapat disimpulkan bahwa program Video Streaming yang dibuat memiliki ketahanan layanan pada saat terjadi gangguan koneksi pada alat yang digunakan

3.4.2. Ketahanan Layanan Pada Penggunaan Dengan Waktu Lama

Pengujian kali ini akan menguji ketahanan layanan saat melayani klien lebih dari 8 jam secara kontinyu. Pemilihan rentang waktu 8 jam ini berdasarkan pada jam kerja di Indonesia karena layanan ini nantinya akan dikembangkan di lingkungan kampus ataupun kantor. Pada pengujian ini nantinya akan menguji apakah layanan dapat diakses selama lebih dari 8 jam oleh user.



Gambar 7. Hasil program saat pertama kali dijalankan

Dapat dilihat pada log program tersebut bahwa program pertama dijalankan tanggal 27 September 2017 pada jam 02:36:57. Program Video Streaming ini diakses oleh 1 klien dengan IP Address 192.168.69.139

```
192.168.69.139 - - [27/Sep/2017 11:13:27] "GET /1.mjpeg HTTP/1.1"
192.168.69.139 - - [27/Sep/2017 11:13:27] "GET /1.mjpeg HTTP/1.1"
192.168.69.139 - - [27/Sep/2017 11:13:27] "GET /1.mjpeg HTTP/1.1"
192.168.69.139 - - [27/Sep/2017 11:13:27] "GET /1.mjpeg HTTP/1.1"
192.168.69.139 - - [27/Sep/2017 11:13:27] "GET /1.mjpeg HTTP/1.1"
192.168.69.139 - - [27/Sep/2017 11:13:27] "GET /1.mjpeg HTTP/1.1"
192.168.69.139 - - [27/Sep/2017 11:13:28] "GET /1.mjpeg HTTP/1.1"
192.168.69.139 - - [27/Sep/2017 11:13:28] "GET /1.mjpeg HTTP/1.1"
192.168.69.139 - - [27/Sep/2017 11:13:28] "GET /1.mjpeg HTTP/1.1"
192.168.69.139 - - [27/Sep/2017 11:13:28] "GET /1.mjpeg HTTP/1.1"
192.168.69.139 - - [27/Sep/2017 11:13:28] "GET /1.mjpeg HTTP/1.1"
192.168.69.139 - - [27/Sep/2017 11:13:28] "GET /1.mjpeg HTTP/1.1"
192.168.69.139 - - [27/Sep/2017 11:13:28] "GET /1.mjpeg HTTP/1.1"
192.168.69.139 - - [27/Sep/2017 11:13:28] "GET /1.mjpeg HTTP/1.1"
192.168.69.139 - - [27/Sep/2017 11:13:28] "GET /1.mjpeg HTTP/1.1"
192.168.69.139 - - [27/Sep/2017 11:13:28] "GET /1.mjpeg HTTP/1.1"
192.168.69.139 - - [27/Sep/2017 11:13:28] "GET /1.mjpeg HTTP/1.1"
----------------------------------------
Exception happened during processing of request from ('192.168.69
Traceback (most recent call last):
  File "/usr/lib/python2.7/SocketServer.py", line 599, in process
    self.finish_request(request, client_address)
```

Gambar 8. Hasil program saat klien tidak mengakses layanan

Dapat dilihat bahwa klien tidak menggunakan layanan lagi pada tanggal 27 September 2017 jam 11:13:28. Bila dihitung dari awal program dinyalakan (02:36:57), berarti program sudah berhasil melakukan layanan *video streaming* ke klien selama hampir 9 jam secara *nonstop*. Selama hampir 9 jam itu pula berdasarkan pemantauan pada sisi klien maupun log program, tidak ada gangguan selain waktu *delay* yang memang muncul pada setiap *stream player* yang digunakan.

4. Kesimpulan

Program Video Streaming dapat dioperasikan normal pada perangkat Raspberry Pi 2 tipe B dengan spesifikasi : CPU 900 Mhz quad-core ARM Cortex-A7, Power 4.0W dengan minimal RAM 1GB. Dengan kamera Raspi Module v1 dengan resolusi 5 MP. Program dapat dijalankan pada sistem autostart, yang berarti akan berjalan secara otomatis sesaat setelah perangkat Raspberry Pi dinyalakan. Pengujian menggunakan 1 perangkat dengan fitur *autostart* membutuhkan memori *virtual* Raspberry Pi lebih sedikit dibandingkan dengan pengujian menggunakan 1 perangkat tanpa *autostart* dan 2 perangkat bersamaan. Dengan menggunakan 2 perangkat yang berbeda & aplikasi *video stream player* yang berbeda akan membuat waktu *delay* yang tidak beraturan di kedua perangkat tersebut.

Penggunaan fungsi *multiclient* pada program Video Streaming yang paling efektif tanpa adanya gangguan yang berarti itu berada pada penggunaan 4 perangkat pengguna secara bersamaan. Layanan *video streaming* ini dapat memberikan hasil yang baik, minimal dengan *framerate* 8 fps. Program Video Streaming yang dibuat ini dapat melanjutkan pengiriman data *frame* yang sebelumnya berhenti saat terjadi gangguan koneksi. Layanan ini memiliki ketahanan yang cukup baik untuk melayani klien secara *nonstop* selama hampir 9 jam tanpa munculnya gangguan tambahan selain waktu *delay*.

Program ini mudah diaplikasikan jika ingin memanfaatkan Raspberry Pi sebagai media video streaming. Karena selain hanya perlu menjalankan sebuah program berbahasa pemrograman python yang sudah terintegrasi baik dengan perangkat Raspberry Pi, juga dalam segi pengguna dimudahkan karena bisa memilih dari aplikasi apa hasil *video streaming* tersebut akan diputar

Referensi

[1]. Arsam, Arfiandy, 2014, “Pembangunan Aplikasi Video Streaming berbasis Android di STV Bandung”, Jurnal Ilmiah Komputer dan Informatika (KOMPUTA)

[2]. Wei, Chongyu, 2014, “Applications of a Streaming Video Server in a Mobile Phone Live Streaming System”, Journal of Software Engineering and Applications

[3]. Shadiq, Helmi Muhammad, 2014, “Perancangan Kamera Pemantau Nirkabel Menggunakan Raspberry Pi Model B”

[4]. Yustini, Aprinal Adila A, 2009, “Video Streaming dengan Videolan Project”, Elektron: Vol. 1 No. 2

[5]. Upton Eben, Halfacree Gareth, 2013, “*Raspberry Pi User Guide*”