IJEIS (Indonesian Journal of Electronics and Instrumenations Systems)
Vol.x, No.x, April xxxx, pp. 1~5
ISSN (print): 2088-3714, ISSN (online): 2460-7681
DOI: 10.22146/ijeis.xxxx

Penerapan dan Analisis HTTP/2 Server-Sent Events dan WebSocket untuk Web Application pada Sistem Rumah Pintar

M. Rusminto Hadiyono*[1], Muhammad Arrofiq[2]
[1]Departemen Teknik Elektro dan Informatika, SV UGM, Yogyakarta, Indonesia
[2]Departemen Teknik Elektro dan Informatika, SV UGM, Yogyakarta, Indonesia
e-mail: ***[1]rusminto900@gmail.com,** rofiq@ugm.ac.id

Abstrak

Salah satu hal yang sering menjadi kendala dalam penerapan rumah pintar adalah cara mereka agar bisa terhubung dengan perangkat apapun di dalam rumahnya melalui internet. Pengguna harus tahu metode pengiriman apa yang diperlukan untuk mengirimkan atau menerima data dari web browser ataupun antarmuka lainnya menuju mikrokontroler dengan jeda pengiriman secepat mungkin. Permasalahan tersebut dapat dipecahkan dengan memanfaatkan beberapa teknologi sekaligus, semisal dengan menggabungkan WebSocket atau Server-Sent Events (SSE) dengan MQTT serta Serveo. Di antara WebSocket, HTTP/1.1 SSE, HTTPS SSE, serta HTTP/2 SSE perlu dilakukan pemilihan metode pengiriman yang cocok untuk digunakan pada sistem kendali rumah pintar. Dengan alasan tersebut, pengujian response time dan presentase penggunaan CPU dilakukan kepada keempat metode pengiriman tersebut. Hasil yang didapatkan adalah WebSocket memiliki nilai response time terkecil sedangkan HTTP/2 SSE memiliki nilai presentase penggunaan CPU terkecil dalam pengujian pengiriman dua arah.

***Kata kunci**—WebSocket, Server-Sent Events, MQTT, smart home, HTTP/2*

Abstract

One of the obstacle to create a smart home is how to control their system from internet. The smart home owner have to choose the fastest way to send data from web browser to the microcontroller or vice versa. Many ways can be used to reach that goal, such as combine WebSocket or Server-Sent Events (SSE), MQTT and Serveo. Among those methods (WebSocket, HTTP/1.1 SSE, HTTPS SSE, HTTP/2 SSE), we still need to choose a method that compatible to smart home system condition. To choose a method, we need to examine response time and CPU usage on those methods. The result from examination is WebSocket has the lowest response time value compared to other methods and HTTP/2 SSE has the lowest CPU usage percentage on two-ways transmission.

***Keywords—**WebSocket, Server-Sent Events, MQTT, smart home, HTTP/2*

Received June 1st,2012; Revised June 25th, 2012; Accepted October 10th, 2012

1. PENDAHULUAN

Konsep yang paling penting dari *Internet of Things* adalah mengintegrasikan semua hal yang ada di dunia nyata ke dalam dunia digital dengam melalui jaringan IP [1,2]. Salah satu penerapan dari *Internet of Things* adalah pengendalian rumah pintar melalui *web browser*. Hal yang perlu diperhatikan dalam pembuatan sistem kendali rumah pintar adalah kecepatan transaksi informasi, yang mana sebaiknya memiliki nilai *response time* serendah mungkin [3].

Dalam proses pengiriman data menuju *web browser* dengan rentang waktu sependek mungkin, terdapat berbagai pilihan yang dapat diterapkan pada rumah pintar, seperti halnya *Polling*, *Long-Polling*, *WebSocket* dan *Server-Sent Events*. Teknologi *Polling* dan *Long-Polling* kurang cocok digunakan dalam rumah pintar karena keduanya membutuhkan *bandwith* dan *response time* yang besar, berbeda dengan *WebSocket* ataupun *Server-Sent Events* [4,5,6].

Selain dari kecepatan transaksi data, rumah pintar cenderung dibuat secara sederhana. Karena itu, *server* yang digunakan untuk rumah pintar seringkali memiliki spesifikasi CPU maupun memori yang lebih rendah.Dari keempat metode pengiriman tersebut, WebSocket serta *Server-Sent Events* memiliki presentase penggunaan CPU lebih rendah daripada metode lainnya [7,8]. Selain itu, performa WebSocket serta *Server-Sent Events* juga dipengaruhi oleh *web browser* serta konfigurasi jaringan yang digunakan [9].

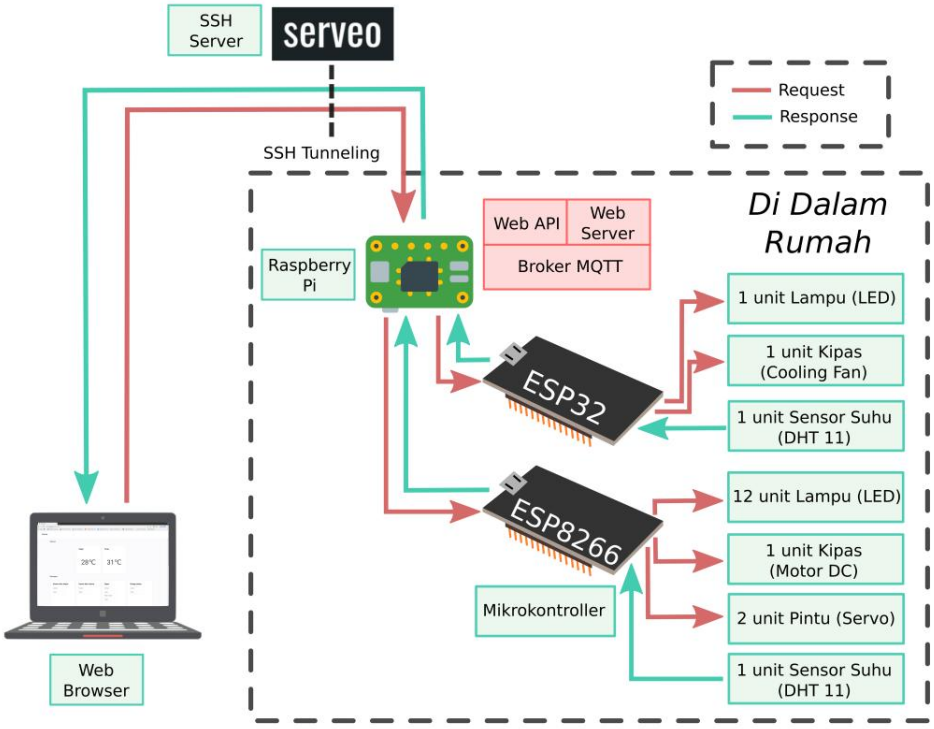
Berbeda dengan WebSocket, *Server-Sent Events* mengirimkan data sepenuhnya melalui protokol HTTP. Protokol HTTP memiliki dua generasi yang umum digunakan yakni HTTP/1.1 serta HTTP/2. HTTP/2 memiliki keunggulan dibandingkan dengan HTTP/1.1 yang mana mampu menangani *Head of Blocking*. Dengan ditanganinya *Head of Blocking* yang sering menjadi penyebab lamanya data masuk ke dalam *web browser* pada HTTP/1.1, HTTP/2 memiliki nilai *response time* lebih rendah dibandingkan dengan HTTP/1.1. Kelebihan lain yang dimiliki oleh HTTP/2 adalah kemampuan untuk melakukan koneksi paralel hanya dengan melalui satu jalur *TCP Connection* [10]. Hal ini sangat menguntungkan terutama apabila digunakan pada *Server-Sent Events*. *Server-Sent Events* yang diterapkan pada HTTP/2 mampu untuk menghapus beberapa batasan yang ada pada HTTP/1.1 seperti halnya jumlah maksimal koneksi terbuka yang hanya berjumlah enam.

Perbedaan antara WebSocket, HTTP/1.1 Server-Sent Events dengan HTTP/2 Server-Sent Events perlu dilakukan penelitian lebih lanjut untuk mengetahui yang mana di antara ketiganya yang sesuai dengan kondisi sistem kendali rumah pintar.

2. METODE PENELITIAN

2.1 Desain Topologi Perangkat

Untuk membangun sistem kendali rumah pintar dibutuhkan beberapa perangkat keras yang mendukung kondisi pengujian. Perangkat keras yang digunakan dapat dikelompokan menjadi tiga bagian utama yakni *web browser*, *server* serta mikrokontroler. Pada sisi *web browser* dibutuhkanlah komputer yang mampu membuka *web browser* yang sering digunakan semisal Google Chrome. Untuk *server* yang digunakan berupa Raspberry Pi 3 Model B yang memiliki memori serta CPU yang cukup untuk menjalankan *server* berskala kecil. Terakhir, pada sisi mikrokontroler digunakan NodeMCU dengan tipe ESP ESP32 dan ESP8266. Untuk lebih lengkapnya dapat diamati pada Gambar 1.



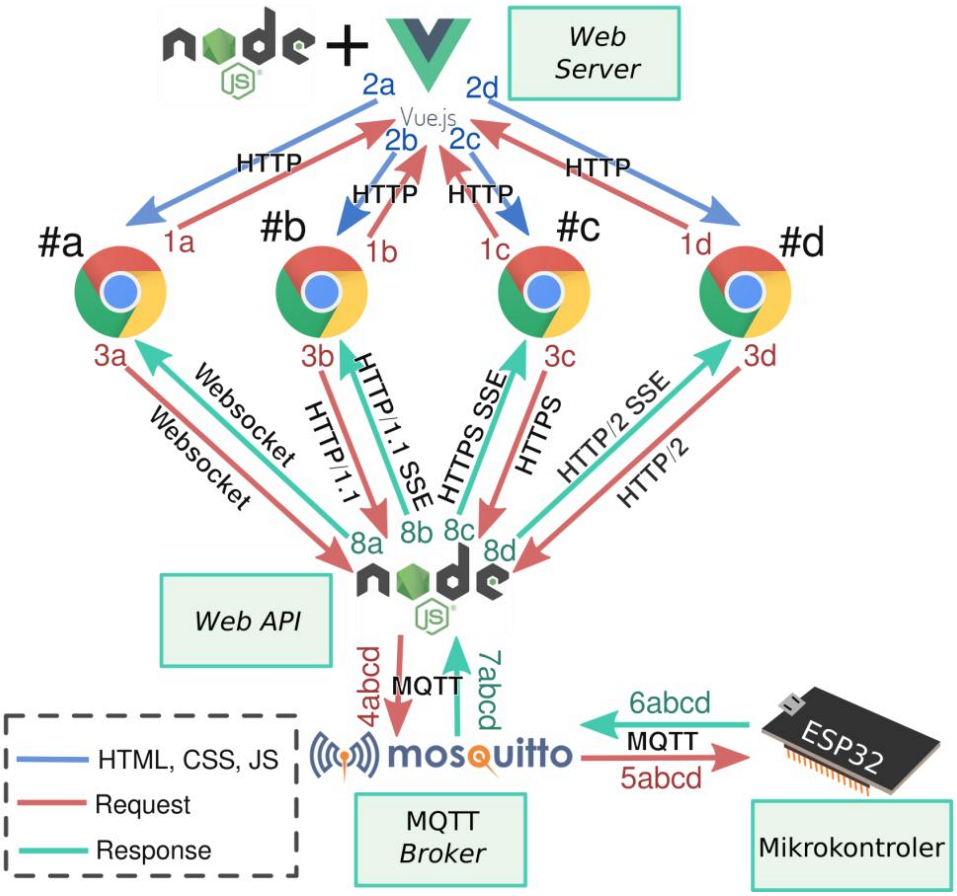
Gambar 1 Bagan Topologi Perangkat

Komponen *server* dapat dibagi menjadi tiga bagian, yakni *web server*, *web API*, serta *MQTT Broker*. *Web server* berperan untuk mengirimkan data *web application* yang berupa HTML, CSS maupun Javascript. *Web application* selanjutnya akan ditampilkan di *web browser* pengguna. Untuk pembuatan *web application* dapat dilakukan dengan berbagai metode. Dalam penelitian ini, *web application* dibuat menggunakan *Javascript Framework* yakni "Vue.js". Selanjutnya, terdapat bagian *web API* yang bertugas untuk mengolah data yang berasal maupun menuju *MQTT Broker*. Baik *web API* maupun *web server*, keduanya bekerja dengan menggunakan perangkat lunak Node.js. Kemudian data yang telah masuk ke *MQTT Broker* akan diteruskan menuju mikrokontroler .

Dalam penerapannya, server tanpa menggunakan Serveo hanya dapat diakses pada jaringan lokal. Serveo adalah server SSH yang digunakan untuk *tunneling* dari suatu komputer sehingga mampu untuk diakses dari luar jaringan. Terdapat pula pihak ketiga yang serupa dengan Serveo, yakni Ngrok. Pemilihan Serveo daripada Ngrok didasarkan pada biaya yang dikeluarkan. Ngrok membutuhkan biaya tambahan ketika melewatkan data melalui HTTPS, padahal HTTPS diperlukan ketika menggunakan HTTP/2.

2.2 Desain Topologi Data

Topologi data berguna untuk menjelaskan alur pengiriman data yang berlangsung selama sistem berjalan. Topologi ini juga yang akan digunakan ketika proses pengambilan data berlangsung. Untuk lebih detailnya, bagan perancangan topologi data dapat diamati pada Gambar 2.

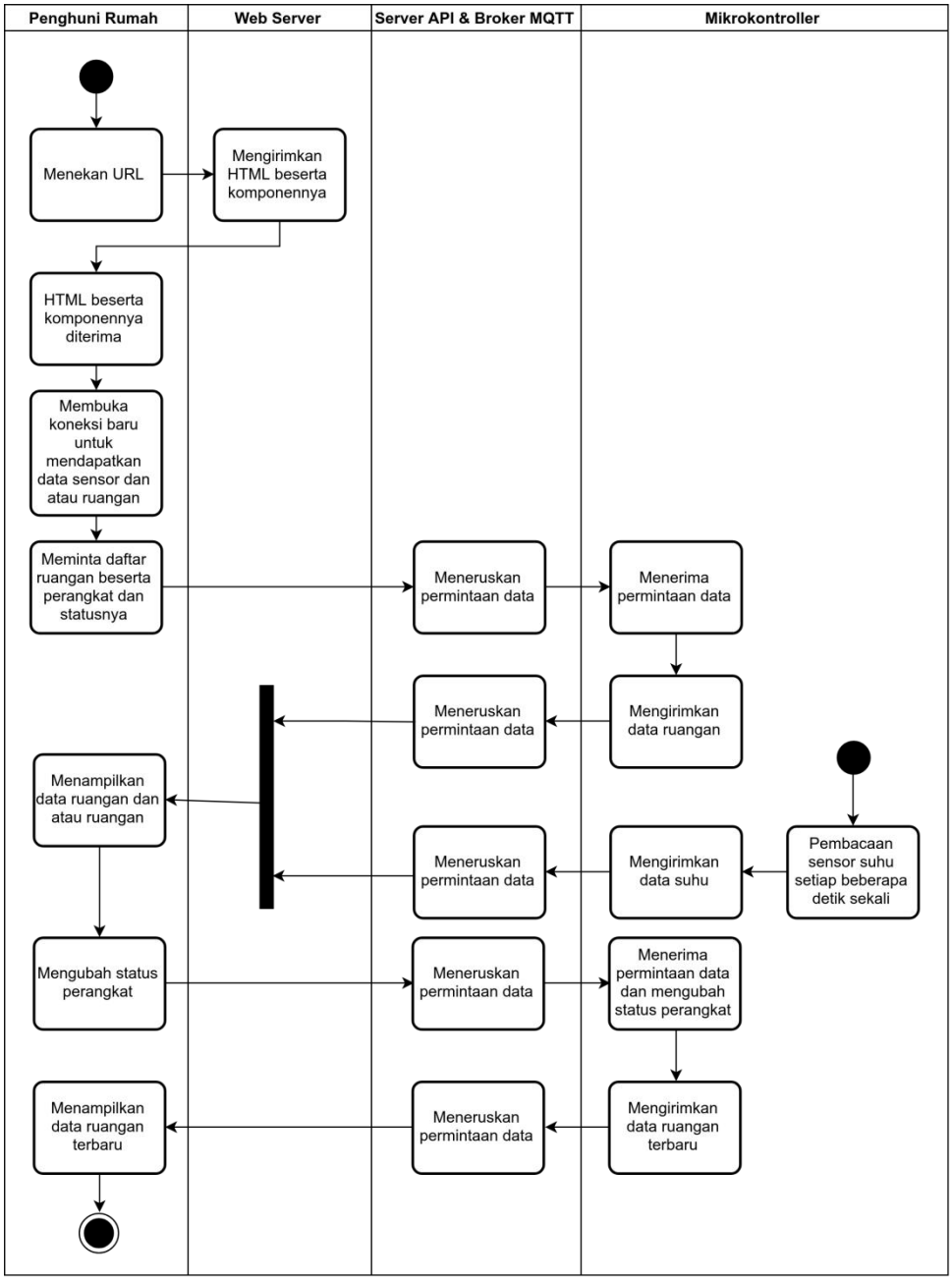


Gambar 2 Bagan Topologi Data

Pada sistem ini digunakan empat metode pengiriman untuk komunikasi antara *web browser* dengan *web API*. Hal ini ditunjukkan oleh Gambar 2 dengan notasi a, b, c dan d yang mewakili setiap skenario pengiriman data. Notasi a mewakili skenario program pengirim melalui WebSocket, notasi b untuk HTTP/1.1 SSE, notasi c untuk HTTPS sedangkan notasi d untuk HTTP/2 SSE. Untuk mempermudah penyebutan keempat skenario digunakanlah notasi abcd.Secara keseluruhan terdapat tiga proses utama yang berlangsung selama sistem berjalan yakni pemuatan *web application* di *web browser* **(Notasi 1a/1b/1c/1d menuju 2a/2b/2c/2d)**, pengiriman perintah dari *web browser* menuju mikrokontroler **(Notasi 3a/3b/3c/3d menuju 8a/8b/8c/8d)** serta pengiriman data suhu dari mikrokontroler menuju *web browser* **(Notasi 6abcd menuju 8a/8b/8c/8d)**.

2.3 Perancangan Model

Perancangan model dibuat untuk menggambarkan alur pemakaian sistem oleh pengguna sistem kendali rumah pintar. Dalam pembuatannya, perancangan model dibuat menggunakan *Unified Modeling Language* (UML) Diagram yang mana memudahkan pengembang maupun orang lain untuk menyampaikan alur kerja suatu sistem yang telah dibuat. UML Diagram yang digunakan dalam perancangan model adalah *activity diagram*. *Activity Diagram* digunakan untuk menggambarkan aliran kerja atau langkah-langkah yang ditempuh selama sistem berjalan. Pada Gambar 3 terlihat activity diagram dari sistem yang akan dibuat.



Gambar 3 Activity Diagram Sistem

2.4 Metode Pengambilan Data

Metode pengiriman data yang dipilih untuk digunakan dalam sistem kendali rumah pintar adalah HTTP/1.1 SSE, HTTPS SSE, HTTP/2 SSE serta WebSocket. Keempat metode dibandingkan berdasarkan nilai *response time* serta presentase penggunaan CPU yang didapatkan setelah proses pengambilan data dilakukan.

2.4.1 Response Time

Pengambilan data *response time* untuk keempat metode dilakukan dengan cara menghitung selisih waktu dari dua puluh perintah yang dikirimkan dari *web browser* sampai mendapatkan balasan dari mikrokontroler. Keduapuluh perintah dikirimkan dengan jeda waktu pengiriman yang bervariasi serta memiliki ukuran paket yang sama. Perhitungan waktu

dilakukan dengan menandai waktu awal pengiriman perintah serta waktu akhir ketika pesan balasan diterima dari sisi *web browser*. Proses pengambilan *response time* dilakukan dalam dua skenario yang berbeda, yakni di dalam jaringan lokal yang sama serta melalui jaringan internet. Untuk kedua skenario tersebut digunakan jaringan Indihome untuk mengakses *web server* maupun *web API* dari *web browser*, namun supaya *web API* dapat diakses dari luar jaringan lokal dibutuhkan aplikasi pihak ketiga, yakni Serveo.

2.4.2 Penggunaan CPU

Pengambilan data presentase penggunaan CPU dilakukan dengan menggunakan bantuan aplikasi 'top'. Skenario pengambilan data presentase penggunaan CPU dilakukan dengan cara mengirimkan perintah setiap detiknya dari beberapa *web browser* menuju mikrokontroler secara bersamaan. Setiap *web browser* diibaratkan sebagai satu pengguna. Setiap perintah yang dikirimkan dari *web browser* akan dibalas oleh mikrokontroler. Perintah yang dikirimkan dari *web browser* memiliki ukuran paket yang sama, begitu juga dengan perintah yang dikirimkan dari mikrokontroler.

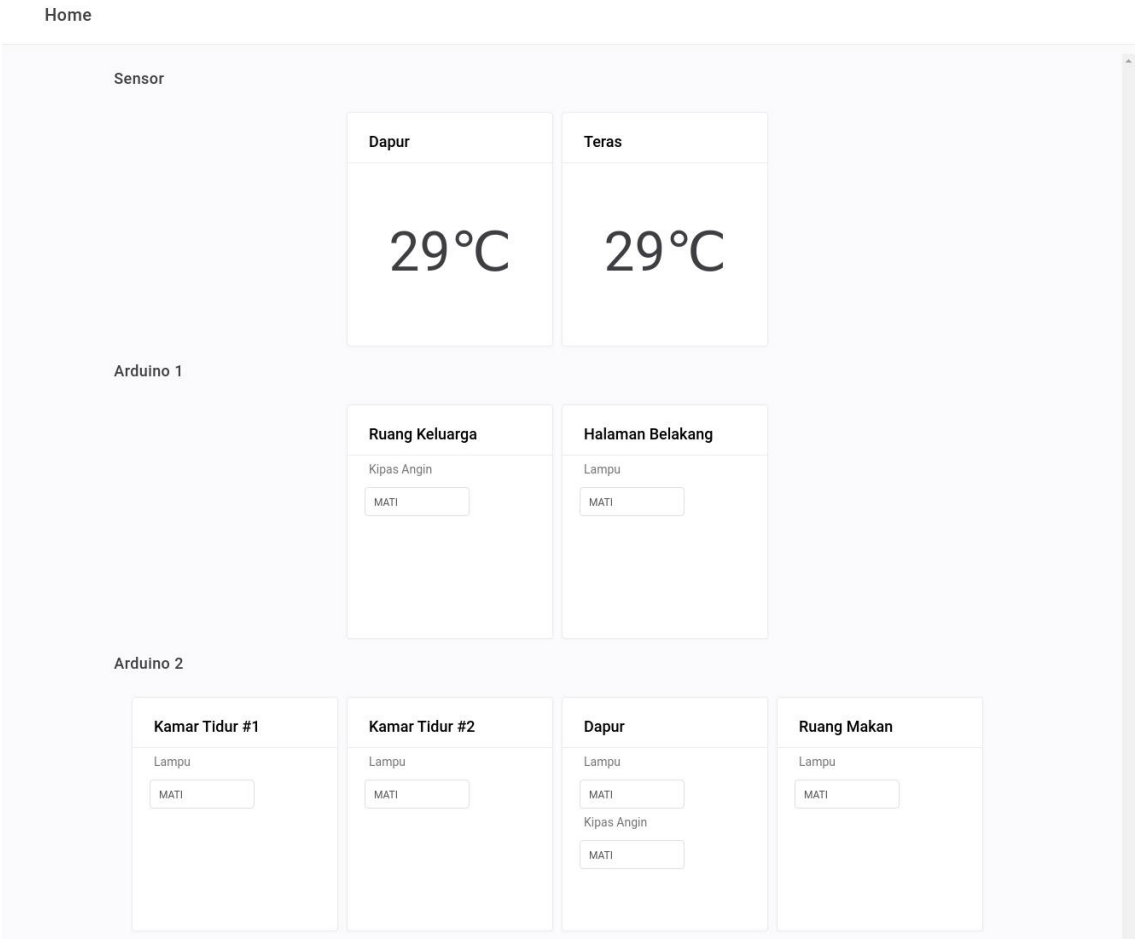
Dua *web browser* yang sama ketika dibuka pada waktu yang bersamaan akan menggunakan *session*, *history*, maupun *cache* yang sama. Hal ini mengganggu proses pengambilan data HTTP/1.1 SSE maupun HTTPS SSE yang hanya mampu membuka enam *TCP Connection*. Untuk mengatasi hal tersebut dibuat *session* yang berbeda untuk setiap membuka *web browser* dengan menjalankan *script* berikut di terminal Linux.

```
#!/bin/bash

RND_DIR="chromee-$RANDOM"

echo $RND_DIR

cd /tmp

mkdir $RND_DIR

google-chrome --user-data-dir=/tmp/$RND_DIR --incognito

rm -R $RND_DIR
```

3. HASIL DAN PEMBAHASAN

3.1 Tampilan Web Application

Berdasarkan perancangan yang telah dikerjakan, tampilan *Web Application* dapat dilihat pada Gambar 4.



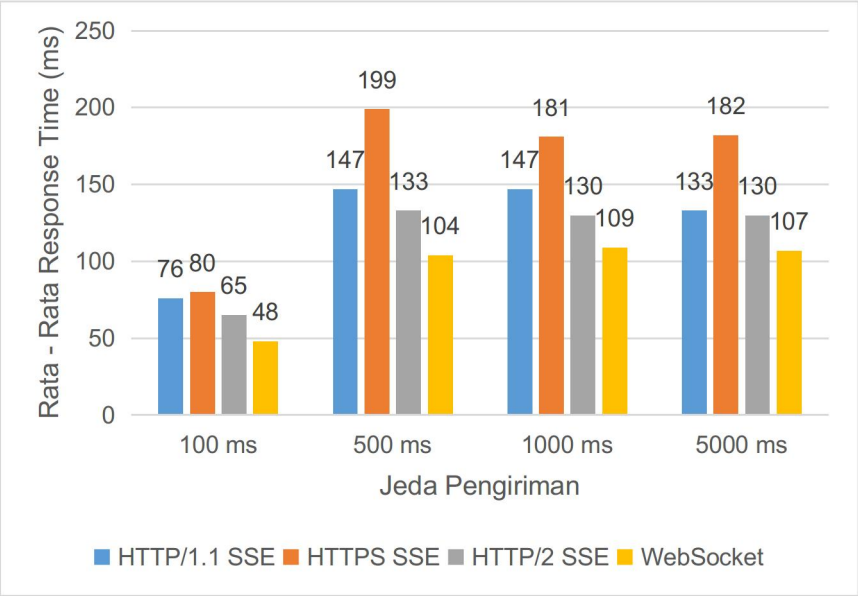
Gambar 4 Tampilan Website

3.1 Hasil Perbandingan Response Time

Response time merupakan lama waktu yang dibutuhkan oleh *web browser* untuk mendapatkan balasan dari mikrokontroler setelah *web browser* melakukan pengiriman perintah menuju mikrokontroler. Hal ini ditunjukkan Gambar 2 dengan notasi dari **3a/3b/3c/3d** menuju **8a/8b/8c/8d**. Data yang didapatkan oleh *web browser* berupa status piranti elektronik yang terhubung dengan mikrokontroler.

3.1.1 Skenario Jaringan Lokal

Pada skenario ini dilakukan pengiriman lima puluh pesan dari *web browser* menuju mikrokontroler dengan kondisi *web browser* berada di dalam jaringan lokal yang sama dengan *web API*. Hasil yang didapatkan dari pengiriman kelimapuluh pesan telah dirangkum pada Gambar 5.

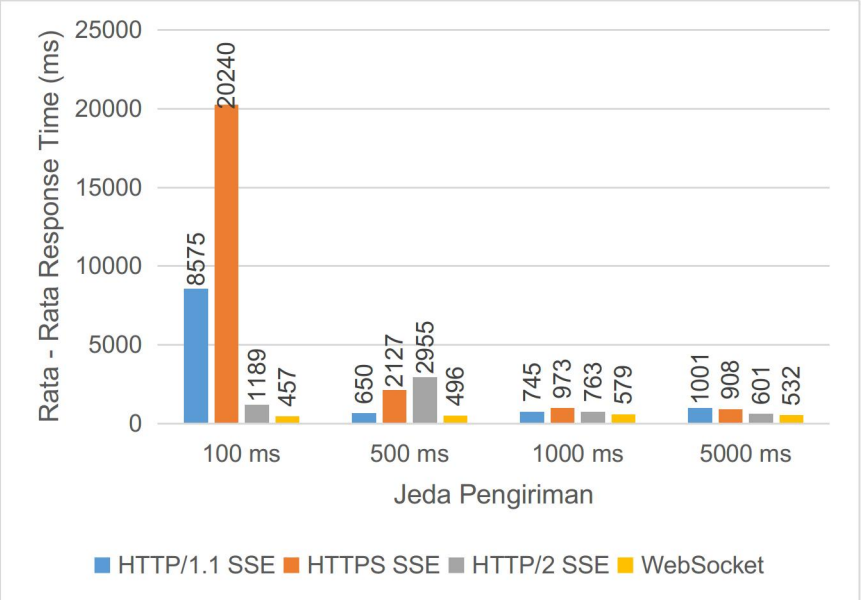


Gambar 5 Grafik rata-rata response time pada skenario jaringan lokal

Berdasarkan hasil yang didapatkan pada percobaan ini dapat dinyatakan bahwa WebSocket memiliki nilai rata - rata *response time* terkecil. Selain itu, nilai rata-rata *response time* dari keempat metode selalu memiliki urutan yang sama. Urutan dapat dimulai dari pemilik rata - rata *response time* terkecil yakni WebSocket, HTTP/2 SSE, HTTP/1.1 SSE sampai dengan HTTPS SSE.

3.1.2 Skenario Jaringan Internet

Proses pengambilan data pada skenario ini menggunakan cara yang sama dengan skenario jaringan lokal, namun dengan kondisi *web browser* terhubung dengan *web API* melalui jaringan internet dengan bantuan *tunneling* oleh Serveo. Hasil yang didapatkan dari percobaan ini telah dirangkum pada Gambar 6.

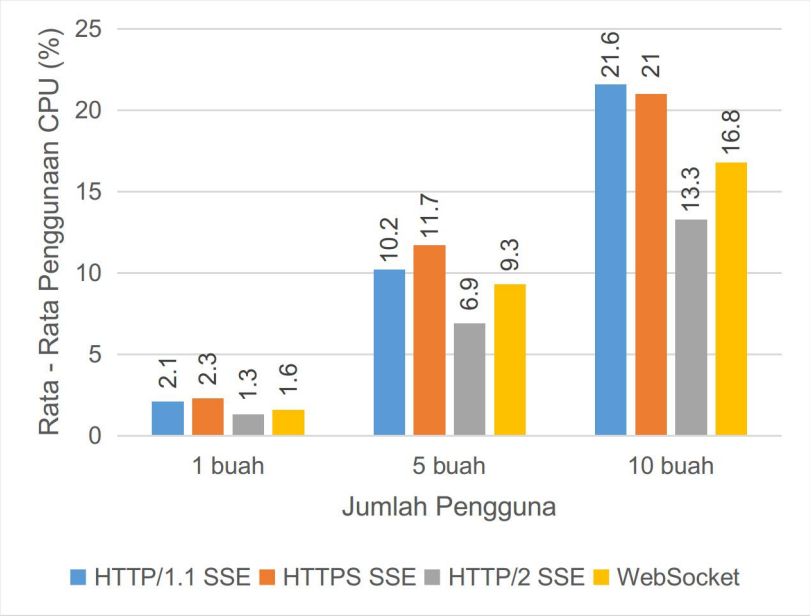


Gambar 6 Grafik rata-rata response time pada skenario jaringan internet

Variasi data yang didapatkan ketika melalui jaringan internet jauh berbeda dibandingkan dengan skenario jaringan lokal, namun untuk hasil yang didapatkan tetaplah sama yakni WebSocket memiliki rata - rata *response time* terkecil dibandingkan dengan ketiga metode lainnya. Hasil lain yang didapatkan dari percobaan ini adalah ketika digunakannya Serveo ada kemungkin hilangnya suatu pesan yang dikirimkan oleh mikrokontroler maupun *web browser* ketika pengiriman dilakukan melalui HTTP/1.1 maupun HTTP/1.1 over TLS.

3.2 Hasil Perbandingan Penggunaan CPU

Proses pengambilan data presentase penggunaan CPU dilakukan dengan mengirimkan perintah menuju mikrokontroler dengan rentang satu detik selama percobaan berlangsung. Selama itu pula dilakukan pengamatan terhadap kenaikan presentase penggunaan CPU oleh Node.js pada perangkat *server* Raspberry Pi. Hasil yang didapatkan dari percobaan ini telah dirangkum pada Gambar 7.



Gambar 7 Grafik rata-rata penggunaan CPU

Berdasarkan hasil yang diperoleh pada percobaan ini dapat dinyatakan bahwa HTTP/2 SSE memiliki nilai rata - rata presentase penggunaan CPU terkecil. Selain itu, nilai rata – rata presentase penggunaan CPU dari keempat metode cenderung memiliki urutan yang sama. Urutan dapat dimulai dari pemilik rata - rata presentase penggunaan CPU terkecil yakni HTTP/2 SSE, WebSocket, HTTP/1.1 SSE sampai dengan HTTPS SSE.

4. KESIMPULAN

Hasil yang didapatkan dari penelitian ini adalah WebSocket memiliki nilai *response time* terkecil diantara keempat metode lainnya sedangkan HTTP/2 Server-Sent Events + HTTP/2 POST Request memiliki nilai presentase penggunaan CPU terendah. Selain itu, terdapat kemungkinan paket yang hilang selama penggunaan Serveo untuk HTTP/1.1 *Server-Sent Events* maupun HTTPS *Server-Sent Events*.

REFERENCES

[1] Kwan, Joel, Yassine Gangat, Denis Payet, and Remy Courdier. 2016. "An Agentified Use of the Internet of Things." In 2016 IEEE International Conference on Internet of Things (IThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), 311–16. IEEE. Available: http://ieeexplore.ieee.org/document/7917104/ [Accessed: 15-Jul-2019]

[2] Cirani, Simone, Gianluigi Ferrari, Marco Picone, and Luca Veltri. 2018. Internet of Things : Architectures, Protocols and Standards. New Jersey: John Wiley & Sons, Inc.

[3] Saito, Nobuo, and David Menga. 2015. *Ecological Design of Smart Home Networks : Technologies, Social Impact and Sustainability*. Cambridge: Woodhead Publishing.

[4] Souders, Steve. 2009. *Even Faster Web Sites: Performance Best Practices for Web Developers*. California: O'Reilly Media.

[5] Kayal, Paridhika, and Harry Perros. 2017. "A Comparison of IoT Application Layer Protocols through a Smart Parking Implementation." *Proceedings of the 2017 20th Conference on Innovations in Clouds, Internet and Networks, ICIN 2017*, no. January: 331–36.

[6] Ramli, Kalamullah, Asril Jarin, and Suryadi Suryadi. 2018. "A Real-Time Application Framework for Web-Based Speech Recognition Using HTTP/2 and SSE." *Indonesian Journal of Electrical Engineering and Computer Science* 12 (3): 1230.

[7] Örnmyr, Oliver, and Rasmus Appelqvist. 2017. "Performance Comparison of XHR Polling , Long Polling , Server Sent Events and Websockets." Blekinge Institute of Technology.

[8] Muhammad, Panser Brigade, Widhi Yahya, and Achmad Basuki. 2018. "Analisis Perbandingan Kinerja Protokol Websocket Dengan Protokol SSE Pada Teknologi Push Notification." *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer (JPTIIK) Universitas Brawijaya* 2 (6): 2235–42.

[9] Estep, Eliot. 2013. "Mobile HTML5: Efficiency and Performance of WebSockets and Server-Sent Events." Aalto University.

[10] Peon, R., and H. Ruellan. 2015. "HPACK: Header Compression for HTTP/2." [Online]. Available: https://doi.org/10.17487/RFC7541. [Accessed: 15-Jul-2019]