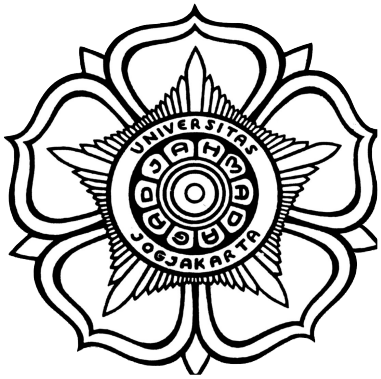
LAPORAN PROYEK AKHIR

PENERAPAN DAN ANALISIS HTTP/2 *SERVER-SENT EVENTS* DAN WEBSOCKET UNTUK *WEB APPLICATION* PADA SISTEM RUMAH PINTAR

ANALYSIS AND IMPLEMENTATION HTTP/2 SERVER-SENT EVENTS AND WEBSOCKET FOR WEB APPLICATION ON SMART HOME SYSTEM



Diajukan oleh :

MUHAMMAD RUSMINTO HADIYONO

15/386767/SV/10153

PROGRAM SARJANA TERAPAN TEKNOLOGI REKAYASA INTERNET

DEPARTEMEN TEKNIK ELEKTRO DAN INFORMATIKA

SEKOLAH VOKASI

UNIVERSITAS GADJAH MADA

YOGYAKARTA

2019

USULAN TUGAS AKHIR YANG DIAJUKAN KEPADA
PROGRAM SARJANA TERAPAN TEKNOLOGI REKAYASA INTERNET
SEKOLAH VOKASI UNIVERSITAS GADJAH MADA

1. JUDUL TUGAS AKHIR	:	Penerapan dan Analisis HTTP/2 *Server-Sent Events* dan WebSocket untuk W*eb Application* pada Sistem Rumah Pintar *Analysis and Implementation HTTP/2 Server-Sent Events and WebSocket for Web Application on Smart Home System*
2. PENYUSUN	:	Muhammad Rusminto Hadiyono
3. DOSEN PEMBIMBING I		
a. Nama Lengkap	:	Muhammad Arrofiq, S.T., M.T., Ph.D.
b. NIP	:	197311271999031001
4. TEMPAT PENELITIAN	:	Ruang Layanan Internet Teknologi Rekayasa Internet UGM.

Yogyakarta, 5 April 2019

Disetujui Oleh :

Dosen Pembimbing	Penyusun
Muhammad Arrofiq, S.T., M.T., Ph.D.	Muhammad Rusminto Hadiyono
NIP. 197311271999031001	NIM. 15/386767/SV/10153

Mengetahui,
Ketua Program Studi Teknologi Jaringan

Muhammad Arrofiq, S.T., M.T., Ph.D.
NIP. 197311271999031001

INTISARI

USULAN TUGAS AKHIR

PENERAPAN DAN ANALISIS HTTP/2 *SERVER-SENT EVENTS* DAN WEBSOCKET UNTUK *WEB APPLICATION* PADA SISTEM RUMAH PINTAR

Akses internet yang cepat dan mudah didapatkan dapat mempermudah masyarakat untuk menerapkan teknologi *Internet of Things.* Bentuk penerapan yang sederhana dari *Internet of Things* yang bisa dimanfaatkan masyarakat adalah rumah pintar. Salah satu hal yang sering menjadi kendala dalam penerapan rumah pintar adalah cara mereka agar bisa terhubung dengan perangkat apapun di rumahnya melalui internet. Untuk terhubung dengan internet, setidaknya pengguna harus memiliki atau menyewa sebuah server terlebih dahulu atau menggunakan melalui perantara pihak ketiga. Selain itu, pengguna harus tahu protokol mana yang sesuai dengan dengan kondisi rumah beserta penggunaan peralatan elektronik di dalamnya. Protokol yang dapat digunakan oleh *Internet of Things* ada banyak dan tentunya setiap protokol memiliki karakteristik yang berbeda. WebSocket dan SSE adalah contoh beberapa protokol atau teknologi yang sering dimanfaatkan untuk *Internet of Things* untuk menghubungkan pengguna dengan perangkat yang tersedia*,* namun yang manakah dari keduanya yang sesuai untuk digunakan di rumah pintar masih memerlukan pembahasan lebih lanjut.

Maka dari itu pada tugas akhir ini dilakukan perbandingan teknologi SSE dengan Webscoket pada sistem pengendali rumah pintar berbasis *web*. Parameter yang diujikan adalah *response time* dan presentase penggunaan CPU.

Kata Kunci : *Internet of Things*, WebSocket, *Server-Sent Events*, MQTT, *smarthome.*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

1.1 Latar Belakang

Dalam menghadapi era Industri 4, Indonesia sudah memiliki banyak perkembangan di bidang teknologi dibandingkan era sebelumnya terutama di dalam pemanfaatan internet. Hal ini terlihat dari data statistik pengguna internet yang diterbitkan oleh Asosiasi Penyelenggara Jasa Internet Indonesia, pada tahun 2016 telah terdapat 132,7 juta pengguna sedangkan pada tahun 2017 sudah terdapat 143,26 juta pengguna dari total populasi penduduk Indonesia sebanyak 262 juta orang (APJII, 2017). Pesatnya pertumbuhan pengguna internet disebabkan oleh semakin cepatnya proses pengiriman data melalui internet serta semakin mudahnya cara untuk mendapatkan akses internet. Hal ini pula yang mendorong semakin beragamnya perangkat yang mampu terhubung dan saling terintegrasi atau lebih dikenal dengan istilah *Internet of Things*.

Konsep yang paling penting dari *Internet of Things* adalah mengintegrasikan semua hal yang ada di dunia nyata ke dalam dunia *digital* (Kwan et al. 2016). Salah satu penerapan dari *Internet of Things* adalah pengendalian rumah pintar melalui *web browser*. Hal yang perlu diperhatikan dalam pembuatan rumah pintar adalah kecepatan transaksi informasi, yang mana sebaiknnya memiliki nilai *response time* serendah mungkin (Saito and Menga 2015). Untuk mendapatkan nilai *response time* yang rendah, dibutuhkanlah infrastruktur jaringan yang bagus serta proses pengiriman data yang cepat dan tepat. Walaupun demikian, penerapan rumah pintar seringkali menggunakan infrastruktur jaringan seadanya serta menggunakan dana seminimal mungkin.

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*. Dari beberapa pilihan tersebut, Polling memiliki metode yang berbeda dengan lainnya dimana *web browser* haruslah secara aktif meminta data ke *server*. Di dalam penerapan *Polling*, *client* akan meminta data ke *server* secara terus-menerus dengan jeda pengiriman yang telah ditentukan. Selain itu terdapat pula, *Long-Polling* yang bekerja dengan cara mengirimkan permintaan ke *server* dan *server* akan menjawab hanya ketika data baru tersedia. Teknologi *Polling* dan *Long-Polling* kurang cocok digunakan dalam rumah pintar karena keduanya membutuhkan *bandwith* dan *response time* yang besar, berbeda halnya dengan *WebSocket* ataupun *Server-Sent Events* (Souders 2009).

WebSocket serta *Server-Sent Events* memungkinkan *web browser* menerima data dari *server* tanpa perlu *request* data baru setiap ada data baru, sehingga data yang telah tersedia di *server* akan dapat langsung dikirimkan ke *web browser*. Dengan berkurangnya waktu yang dibutuhkan untuk mengirimkan data, keduanya mampu untuk lebih *real-time* daripada metode *Polling* maupun *Long-Polling*.

Di sisi lain, proses pengiriman data pada mikrokontroler juga memiliki pengaruh pula terhadap nilai *response time* dari *web browser*. Maka dari itu, dibutuhkanlah metode pengiriman yang cepat dan tepat untuk *Machine-to-Machine*. Dari beberapa protokol *Machine-to-Machine*, protokol MQTT yang berarsitektur *publish-broker-subscribe* memiliki rata-rata *response time* paling rendah (Kayal and Perros 2017). Dengan menggabungkan arsitektur *publish-broker-subscribe* yang dimiliki oleh MQTT dan

kelebihan yang dimiliki *WebSocket* ataupun *Server-Sent Events*, proses pengiriman data dari mikrokontroler menuju *web browser* akan lebih cepat.

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. Salah satu faktor yang berpengaruh terhadap performa *server* adalah proses pengolahan data, termasuk metode pengiriman data.

Baik *WebSocket* maupun *Server-Sent Events* masih memerlukan penelitian lebih jauh untuk mencari teknologi mana yang memiliki nilai *response time* paling rendah serta menggunakan *resource* di *server* terendah pula ketika diterapkan di rumah pintar, dengan alasan itulah penelitian ini dilakukan.

1.2 Rumusan Masalah

Rumusan permasalahan pada penelitian ini adalah bagaimana cara menerapkan HTTP/2 *Server-Sent Events* ataupun WebSocket pada sistem rumah pintar berbasis website serta untuk mengetahui hasil perbandingan response time serta presentase penggunaan CPU pada HTTP/1.1 SSE, HTTPS SSE, HTTP/2 SSE dan WebSocket.

1.3 Batasan Masalah

Beberapa batasan masalah yang akan dilakukan selama proses pelitian proyek akhir adalah sebagai berikut :

1. Software yang dijalankan untuk penelitian ini adalah Mosquitto versi 1.4.8
2. Tidak membahas keamanan pada jaringan maupun perangkat

1.4 Tujuan Penelitian

Tujuan dari penelitian dalam proyek akhir ini adalah mengimplementasikan serta membandingkan metode pengiriman data melalui HTTP/1.1 SSE, HTTPS SSE, HTTP/2 SSE serta WebSocket dari rumah pintar menuju *browser* pengguna beserta sebaliknya menggunakan parameter *response time* serta presentase penggunaan CPU.

1.5 Manfaat Penelitian

Manfaat yang dapat diambil dari penelitian dan pengerjaan proyek akhir ini adalah sebagai berikut :

1. Memberi informasi mengenai implementasi HTTP/2 *Server-Sent Events*, serta *WebSocket* dari *server API* menuju *web browser*.
2. Memberi informasi mengenai cara mudah mengawasi dan mengubah status perangkat dari jarak jauh.
3. Hasil perbandingan *response time* dari ketika *web browser* meminta sampai mendapatkan data dari *server API* dengan metode yang berbeda (HTTP/2 *Server-Sent Events* dan *WebSocket*)

1.6 Sistematika Penulisan

Untuk menggambarkan secara menyeluruh mengenai masalah yang akan dibahas dalam laporan proyek akhir ini, maka dibuat sistematika penulisan yang terbagi dalam lima bab sebagai berikut :

BAB I, PENDAHULUAN, memuat latar belakang masalah, perumusan masalah, batasan masalah, tujuan penulisan, kegunaan penulisan dan sistematika penulisan.

BAB II, TINJAUAN PUSTAKA, berupa uraian sistematis tentang informasi yang relevan dan mutakhir yang terkait dengan lingkup materi penelitian atau teknologi yang akan diterapkan. Uraian dalam tinjauan pustaka ini selanjutnya menjadi dasar teori yang digunakan oleh penulis dalam melaksanakan penelitian dan menyajikan argumentasi dalam pembahasan hasil penelitian.

BAB III, BAHAN DAN METODE PENELITIAN, memuat bahan, peralatan, tahapan penelitian, dan rancangan sistem serta analisis data yang ada pada penelitian ini.

BAB IV, ANALISIS HASIL DAN PEMBAHASAN, memuat semua temuan ilmiah yang diperoleh sebagai data hasil penelitian, atau hasil unjuk kerja prorotipe yang dibuat. Pada bagian ini peneliti menyusun secara sistematis disertai argumentasi yang rasional tentang hasil unjuk kerja yang diperoleh dari hasil penelitian.

BAB V, PENUTUP, memuat kesimpulan serta saran dari penelitian yang dilakukakan pada proyek akhir ini.

BAB II

TINJAUAN PUSTAKA

2.1 Mikrokontroler

Microprocessor telah banyak membantu pekerjaan manusia, baik dalam hal pengendalian suatu peralatan maupun pemantauan. *Microprocessor* dapat diibaratkan sebagai otak dari suatu komputer, yang berisi *Control Unit* (CU) dan *Aritmetic and Logic Unit* (ALU). Dalam penggunaannya, *microprocessor* yang dikenal sebagai *single-chip computer* memerlukan *chip* tambahan untuk dapat bekerja. Sebagai hasilnya, banyak *chip* yang perlu disatukan dan membutuhkan daya yang semakin besar dalam penggunaannya. Di samping itu dengan banyaknya *chip* yang digunakan, biaya yang diperlukan dalam pembuatan pun juga akan semakin mahal dan pemecahan masalah akan semakin sulit dikarenakan rumitnya sambungan antar *chip* (Ibrahim 2014).

Untuk menyelesaikan permasalahan tersebut dibuatlah mikrokontroler, yang dibuat dengan merangkai *microprocessor* bersama dengan *chip-chip* tertentu menjadi sebuah *chip* sehingga mempermudah dalam penggunaan maupun perawatannya. Sebuah mikrokontroler terdiri dari sebuah atau beberapa *microprocessor*, *memory*, ADC (*Analog-to-Digital Controller*), DAC (*Digital-to-Analog Controller*), *Parallel I/O interface*, *Serial I/O interface*, serta penghitung waktu. Dalam penerapannya, mikrokontroler telah dapat ditemukan pada berbagai peralatan elektronik yang telah digunakan sehari-hari seperti mesin cuci, *printer*, *keyboard* maupun beberapa komponen mesin mobil (Udayashankara and S Mallikarjunaswamy 2009).

Saat ini mikrokontroler telah mampu terhubung dengan jaringan internet. Dengan terhubungnya mikrokontroler dengan internet, suatu mikrokontroler mampu mengirim maupun menerima data melalui jaringan menuju sistem lain yang mampu mengelola data-data tersebut untuk ditampilkan ke berbagai antarmuka maupun disimpan. Konsep inilah yang dinamakan *Internet of Things.*

2.2 Konsep *Internet of Things*

Internet of Things merupakan penamaan atas suatu sistem yang menghubungkan suatu atau beberapa *smart object* dengan *smart object* lainnya ataupun dengan sistem informasi lainnya melewati jaringan IP (*Internet Protocol*) (Cirani et al. 2018). Setiap *smart object* terdiri dari *microprocessor*, perangkat komunikasi, sensor atau aktuator serta sumber energi listrik. *Microprocessor* berguna untuk memberikan kemampuan komputasi pada *smart object*. Perangkat komunikasi memungkinkan *smart object* untuk berkomunikasi dengan *smart object* lainnya ataupun sistem lainnya. Sensor atau aktuator menghubungkan *smart object* dengan lingkungannya, memungkinkan mereka untuk mengukur besaran fisis tertentu sampai halnya mengendalikan obyek tertentu. Sumber energi listrik dibutuhkan untuk menjalankan perangkat elektronik pada *smart object*, yang mana dapat berupa baterai ataupun dari sumber energi listrik lainnya (Vasseur et al. 2010). *Internet of Things* dapat diterapkan pada berbagai skenario seperti rumah pintar, *smart cities* serta pengolahan agrikultur (Cirani et al. 2018).

2.3 Rumah Pintar

Rumah pintar adalah istilah yang digunakan untuk sekumpulan perangkat yang digunakan dalam kehidupan sehari-hari yang saling terhubung dalam suatu jaringan. Perangkat yang terhubung bisa berupa lampu, kunci pintu, gerbang, pintu garasi, *DVD*

player, CCTV, sensor suhu, sensor asap sampai halnya *laptop* ataupun *server*. Dengan menerapkan rumah pintar, status atau informasi yang dimiliki dari setiap perangkat dapat diperoleh ataupun diubah melalui perangkat lainnya (Briere and Hurley 2011).

Dalam penerapannya, terdapat banyak pilihan protokol yang dapat digunakan untuk mengendalikan *smart object* yang berada di dalam rumah menuju *server*. Walaupun demikian, dalam penyusunan rumah pintar dibutuhkan protokol yang memiliki proses pengiriman yang cepat dan tepat. Protokol MQTT adalah salah satu protokol yang sering digunakan untuk keperluan pengiriman data *machine-to-machine*.

2.4 *Message Queuing Telemetry Transport*

Message Queuing Telemetry Transport (MQTT) adalah protokol yang ringan dan bekerja dengan sistem *publish-broker-subscribe*. Protokol MQTT bekerja di atas protokol *Transmission Control Protocol* / *Internet Protocol* (TCP/IP). MQTT sangat cocok digunakan untuk pengiriman data yang bersifat *real-time* (Hillar 2017). Selain itu, dalam penerapannya semakin singkat jeda pengiriman data melalui protokol MQTT semakin kecil nilai *delta time*-nya serta nilai integritas data yang dikirim dan diterima mencapai 100% (Rochman, Primananda, and Nurwasito 2017). Walaupun demikian, protokol MQTT tidak bisa digunakan secara langsung ke *web browser*, sehingga protokol MQTT perlu dibungkus dengan *WebSocket* atau dilewatkan ke *server* lain untuk mengubah protokol MQTT ke HTTP. Di dalam penerapannya, membungkus MQTT dengan *WebSocket* dapat diwujudkan dengan mudah. Namun dalam kasus-kasus tertentu, semisal diperlukannya *Application Programming Interface* (API) untuk proses penyimpanan, pengumpulan, serta pengolahan data maka pengubahan metode pengiriman dari MQTT menuju ke HTTP untuk ditampilkan ke *web browser* diperlukan. Selain HTTP masih banyak protokol yang dapat

digunakan untuk mengirimkan data dari *web API* menuju *web browser*. Protokol – protokol tersebut memiliki berbagai karakteristik yang berbeda, namun berdasarkan format data yang dikirimkan protokol-protokol tersebut dapat dikategorikan menjadi dua, yakni *Plain Text Protocol* dan *Binary Protocol.*

2.5 *Binary Protocol* dan *Plain Text Protocol*

Plain Text Protocol adalah protokol yang dapat dengan mudah dibaca oleh manusia, contohnya adalah HTTP/1.1 dan SMTP. Sedangkan *Binary Protocol* adalah protokol yang ditujukan untuk dibaca langsung oleh mesin dengan tujuan mempercepat proses penafsiran data. Contoh dari protokol yang termasuk kategori *Binary Protocol* adalah HTTP/2 serta *WebSocket* (Rhee and Hyun Yi 2015). Data yang dikirimkan melalui *Binary Protocol* cenderung lebih ringan daripada ketika dikirmkan melalui *Plain Text Protocol*. Hal ini dikarekanan data yang dikirimkan melalui *Binary Protocol* lebih berorientasi ke struktur data dibandingkan *Plain Text Protocol* yang lebih cenderung ke *Text String*. Dengan adanya susunan yang lebih mengutamakan struktur data, dapat memungkinkan pengiriman angka dalam bentuk *integer* bukan sebagai beberapa karakter dilakukan. Tidak hanya *integer* saja, tetapi hal ini juga berlaku untuk berbagai tipe data lainnya yang telah ditetapkan oleh suatu *Binary Protocol*. Dari kedua kategori tersebut, HTTP adalah protokol komunikasi yang termasuk ke dalam keduanya.

2.6 *Hypertext Transfer Protocol*

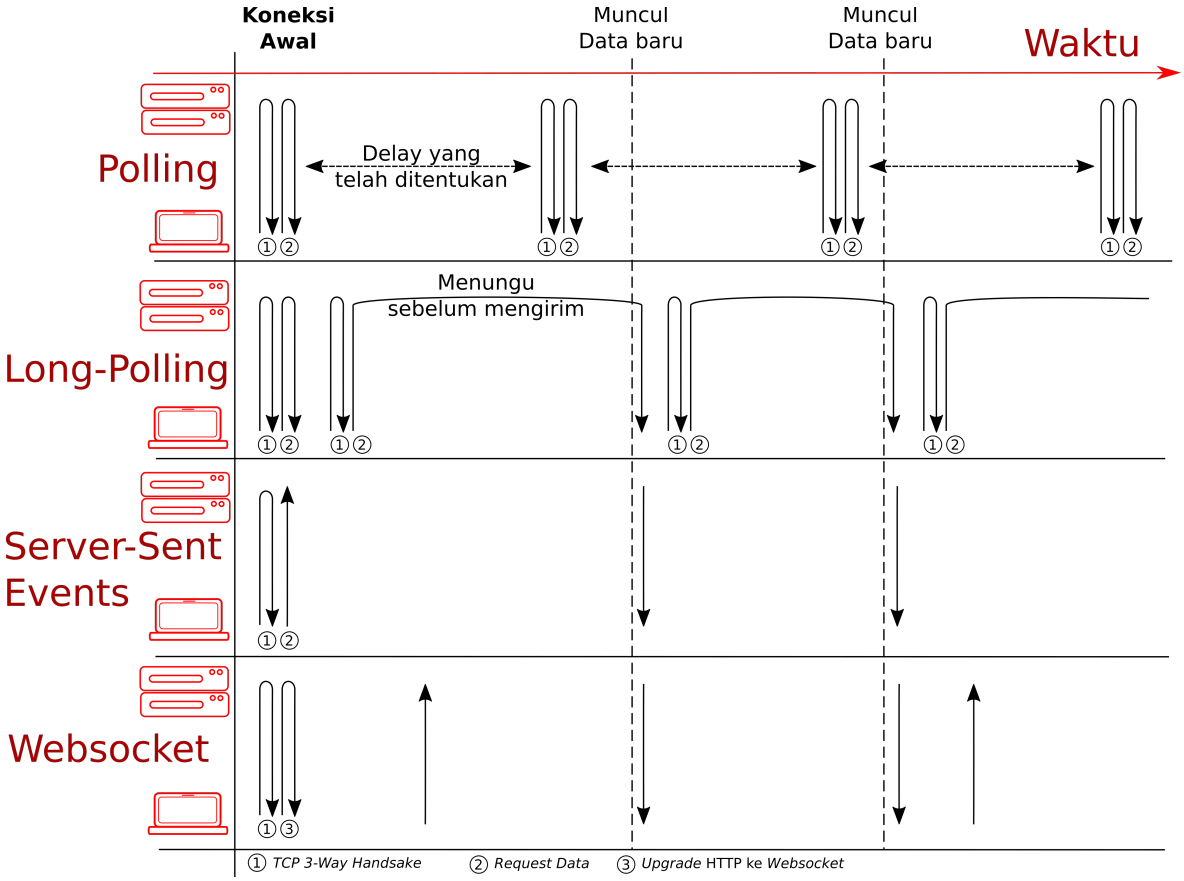
Hypertext Transfer Protocol (HTTP) adalah protokol komunikasi yang berguna untuk komunikasi antara *web browser* dengan *web server* yang telah digunakan oleh *World-Wide Web* sejak 1990. Protokol HTTP berada pada *application-layer* di dalam OSI *Layer.* Dalam perkembangannya, HTTP memiliki beberapa generasi. Diawali dengan HTTP/0.9 yang

berupa protokol sederhana untuk pengiriman *raw data* di *internet*. Setelah itu muncul generasi berikutnya yakni HTTP/1.0 yang memungkinkan pengiriman informasi dalam *format* seperti MIME. Sayangnya, HTTP/1.0 tidak sesuai dengan jaringan yang memakai *proxy*, *caching*, ataupun *virtual host* serta jaringan yang membutuhkan koneksi yang bertahan lama (Leach et al. 1999).

Kemudian muncullah HTTP/1.1, HTTP/1.1 telah memperbaiki masalah yang ada pada HTTP/1.0. Dengan wajib menyertakan *Host header*, memungkinkan HTTP/1.1 untuk melakukan *virtual hosting* ataupun melayani beberapa pengguna yang berbeda pada sebuah alamat IP. Keunggulan dari HTTP/1.1 daripada HTTP/1.0 adalah munculnya OPTIONS *method*, *upgrade* pada *header*, pemampatan dengan *transfer-encoding* maupun *pipelining (Ludin and Garza 2017)*. Pada generasi ini pula, muncul WebSocket yang memanfaatkan kemampuan *upgrade* pada *header* HTTP/1.1.

Setelah HTTP/1.1 bertahan cukup lama, muncullah HTTP/2 yang memungkinkan penggunaan jaringan yang lebih efisien dengan melakukan pemampatan *header* menggunakan HPACK . Selain itu, HTTP/2 mampu menangani *Head of Blocking* yakni kejadian ketika data yang diterima dari *server* harus mengantri untuk bisa dimuat pada halaman HTML. Hal ini mengakibatkan beberapa data yang dikirim maupun diterima dapat dilakukan dalam satu waktu tanpa saling menghalangi dengan kata lain HTTP/2 mampu melakukan *multiplexing*. HTTP/2 juga tidak membutuhkan koneksi tambahan untuk memungkinkan koneksi paralel dan hanya cukup menggunakan satu koneksi HTTP/2 (Peon and Ruellan 2015). Sebaga informasi tambahan, saat ini HTTP/2 masih harus menggunakan koneksi yang aman (TLS/SSL).

Dalam kasus pemantauan sensor maupun status aktuator terbaru dari *web browser*, dibutuhkan metode pengambilan data terbaru dari *server* secara terus-menerus. Pengambilan data terbaru secara terus-menerus dapat dilakukan menggunakan metode *Polling*, *Long-Polling*, *Server-Sent Events*, maupun WebSocket. Dari beberapa pilihan tersebut, perlu dipilih metode yang membutuhkan penggunaan memori dan CPU serendah mungkin dan juga tidak memakan banyak *bandwith*. Untuk cara kerja masing-masing metode dapat diamati pada Gambar 2.1.



Gambar 2.1 Berbagai metode pengiriman data ke *web browser*

Metode *Polling* atau yang lebih sering dikenal dengan 'AJAX' bekerja dengan cara *web browser* melakukan permintaan data dalam setiap kurun waktu yang telah ditentukan. Sehingga apabila tersedia data baru di sisi server, maka data tersebut tidak akan langsung

diterima oleh *web browser*. Dengan data yang tidak langsung diterima, dapat mengakibatkan kenaikan nilai *response time* yang dibutuhkan untuk setiap data baru yang diterima. Selain itu, dengan begitu banyaknya permintaan data untuk setiap data baru yang diterima, metode ini dapat memakan *bandwith* lebih banyak dibandingkan metode-metode lainnya.

Untuk mengatasi data yang tidak langsung diterima pada metode *Polling*, muncullah metode *Long Polling*. Metode ini bekerja dengan melakukan permintaan data terlebih dahulu dan dilanjutkan dengan menunggu tersedianya data baru dari sisi *server*. Apabila data baru telah diterima, *web browser* akan melakukan permintaan data lagi sampai mendapatkan data terbaru dari *server*. Walaupun demikian, metode ini masih memakan banyak *bandwith*, apalagi dalam kondisi dimana *server* menyediakan data baru setiap beberapa *miliseconds.* Masalah lain yang timbul ketika menggunakan metode *Long-Polling* adalah ketika terjadi beberapa permintaan data dari satu *web browser* secara serentak, hal ini dapat menyebabkan data yang diterima saling tumpang tindih ataupun terjadinya duplikasi data secara berlebih. Selain itu, *Long-Polling* memperumit arsitektur server ketika digunakan pada beberapa server yang saling berbagi kondisi *session's client* (Abyl 2018). Metode lain yang dapat digunakan selain *Polling* dan *Long Polling* pada protokol HTTP adalah *Server-Sent Events.*

2.7 *Server-Sent Events*

Server-Sent Events memungkinkan *server* untuk mengirimkan data baru ke *web browser* setiap mucul data baru dari sisi *server* (*streaming*). Berbeda dengan *Long-Polling* maupun *Polling,* metode ini tidak perlu melakukan permintaan data untuk setiap data baru yang muncul pada *server* sehingga dapat mengurangi penggunaan *bandwith,* CPU serta

memori berlebih (Örnmyr and Appelqvist 2017). Setiap kali *web browser* mengirimkan permintaan data ke *server*, metode ini mampu membuka koneksi selama *server* tidak menghentikkannya. Selama itu pula, data baru yang tersedia di *server* dapat langsung dikirimkan ke *web browser* dalam bentuk potongan-potongan data (*chunk*).

Kemampuan lain yang dimiliki oleh *Server-Sent Events* adalah kemampuan untuk terhubung kembali apabila terjadi putus koneksi antara *web browser* dengan *server*. Namun karena itu pula, server tidak mampu mengetahui kapan web browser terputus tanpa mencoba mengirimkan data terlebih dahulu. Selain itu, *Server-Sent Events* pada *web browser* yang diterapkan menggunakan bahasa pemrograman Javascript (*EventSource object*) tidak mampu melakukan penambahan *field* pada *headers*, *field* yang digunakan akan selalu sama yakni hanya berisikan "Content-Type: text/event-stream". Tidak mampu mengubah *headers* berdampak pada ketidakmampuan *web browser* menggunakan *Authorization* yang berguna untuk memastikan keamanan pengirim data ataupun fungsi *field headers* lainnya. Selain itu, dalam penerapannya di HTTP/1.1, *Server-Sent Events* hanya terbatas memiliki enam koneksi yang terbuka dalam satu *web browser*. Namun, dengan dikeluarkannya HTTP/2 koneksi yang mampu terbuka di saat bersamaan semakin bertambah. Sebagai catatan tambahan, Internet Explorer tidak mendukung *Server-Sent Events* maupun *WebSocket* (Elman and Lavin 2014).

2.8 *WebSocket*

Protokol *WebSocket* memungkinkan komunikasi *full duplex* antara *web browser* dengan *server* melalui jaringan internet. Dalam penerapannya, protokol *WebSocket* perlu melakukan permintaan *upgrade* koneksi pada protokol HTTP/1.1. Hal ini dilakukan guna berganti jalur dari HTTP ataupun HTTPS menuju protokol *WebSocket*. Apabila *server*

memutuskan untuk *upgrade* koneksi, *server* akan mengirimkan pesan "101 Switching Protocols", namun jika *server* menolak maka permintaan akan diabaikan. Setelah protokol berpindah ke protokol *WebSocket*, *handsake* pembuka akan dilakukan. Pada *WebSocket*, *handshake* pembuka berupa membukanya suatu koneksi baru. Setelah terbukanya koneksi baru, *server* maupun *web browser* dapat saling bertukar pesan (MDN 2019).

Untuk menggunakan protokol *WebSocket* pada suatu aplikasi, dibutuhkanlah *WebSocket API*. *WebSocket API* dikembangkan oleh *World Wide Web Consortium* (W3C) sedangkan untuk protokol *WebSocket* dikembangkan oleh *Internet Engineering Task Force* (IETF). Dengan menggunakan *WebSocket API*, tindakan untuk membuka dan menutup koneksi, mengirim pesan, maupun menangkap pesan dari *server* melalui protokol *WebSocket* dapat dilakukan (Wang, Salim, and Moskovits 2013). Salah satu kelebihan protokol *WebSocket* adalah memiliki rata-rata *response time* lebih rendah dalam kondisi pengguna *resource* CPU yang terus naik (Kayal and Perros 2017). Walaupun demikian penggunaan protokol *WebSocket* secara langsung tanpa menggunakan (TLS/SSL) rentan terhalangi oleh *proxy server*. Namun hal ini tidak berlaku pada *Secure WebSocket* maupun HTTP/2 dikarenakan data telah terenkripsi dengan TLS/SSL (Ramli, Jarin, and Suryadi 2018).

Dalam penerapannya baik WebSocket maupun *Server-Sent Events*, metode yang digunakan untuk pengiriman data dapat mempengaruhi nilai *response time.* Penelitian untuk menguji *delay* (*one-way trip*) antara *WebSocket* dengan *Server-Sent Events* juga pernah dilakukan dan diperoleh hasil bahwa adalah rata-rata *delay* dari penggunaan *Server-Sent Events* lebih kecil dibandingkan ketika menggunakan *WebSocket (Muhammad, Yahya, and Basuki 2018)*.

2.9 *Response Time*

Response Time merupakan parameter yang digunakan untuk mengukur waktu yang dibutuhkan oleh *web browser* untuk mendapatkan balasan dari *server*. Nilai *response time* dapat dipengaruhi oleh banyak hal. Beberapa faktor yang mempengaruhi nilai *response time* adalah infrastruktur jaringan, lama pengolahan data maupun *web browser* yang digunakan (Estep 2013). Dalam penelitian ini, menghitung *response time* berarti menghitung lamanya waktu yang dibutuhkan dari saat pengguna mengubah status perangkat dari *web browser* sampai mendapatkan status perangkat terbaru dari mikrokontroler.

Tabel 2.1 Ringkasan tinjauan pustaka

No	Judul Penelitian	Penulis & Tahun	Metode	Tipe Penelitian	Pengiriman Data	Kesimpulan
1	Sistem Kendali Berbasis Mikrokontroler Menggunakan Protokol MQTT Pada Smarthome	(Hudan Abdur , dkk. 2017)	Membuat sistem kendali sensor dan lampu LED dengan protokol MQTT pada Smarthome	Purwarupa	ESP8266 (MQTT)	Semakin singkat jeda pengiriman data melalui protokol MQTT semakin kecil nilai *delta time*-nya serta nilai *integritas* data yang dikirim dan diterima mencapai 100%
2	*A Comparison of IoT Application Layer Protocols Through a Smart Parking Implementation*	(Paridhika Khayal, dkk. 2017)	Membandingkan *response time* untuk protokol MQTT, CoAP, XMPP dan MQTT melalui WebSocket	Simulasi	Ubuntu 14.04 (MQTT, CoAP, XMPP dan MQTT-WS)	Ketika penggunaan *resource server* dinaikkan, *response time* WebSocket relatif tetap
3	*A Real-time Application Framework for Speech Recognition Using HTTP/2 and SSE*	(Kalamullah Ramli, dkk. 2018)	Membandingkan HTTP/2 SSE dan WebSocket dalam penerapan *Speech Recognition* pada ns-3	Simulasi	Ubuntu (HTTP/2 dan WebSocket)	Besar latensi aplikasi pada penggunaan HTTP/2 SSE serta WebSocket relatif sama serta WebSocket lebih

						rentan di-*block* oleh *proxy server* dibandingkan HTTP/2
4	Analisis Perbandingan Kinerja Protokol WebSocket dengan Protokol SSE pada Teknologi *Push Notification*	(Panser Brigade Muhammad, dkk. 2018)	Analisis Perbandingan Kinerja Protokol WebSocket dengan Protokol SSE pada Teknologi *Push Notification*	Purwarupa	Ubuntu – smartphone (WebSocket dan HTTP/1.1)	Rata-rata delay pada protokol SSE lebih kecil dibandingkan dengan WebSocket begitu juga dengan presentase penggunaan CPU
5	*Performance comparison of XHR polling, Long-Polling, Server-Sent Events and WebSockets*	(Oliver Örnmyr, dkk. 2017)	Membandingkan penggunaan memori dan CPU dari 100 perangkat virtual yang terhubung dengan server menggunakan *XHR Polling*, *Long-Polling* , *Server-Sent Events* dan WebSockets	Simulasi	Ubuntu (HTTP/1.1 dan Webscoket)	Penggunaan CPU, memori maupun bandwith pada SSE serta WebSocket lebih kecil dibandingkan dengan XHR *Polling* serta *Long-Polling*
6	*Mobile HTML5: Efficiency and Performance of WebSockets and Server-Sent Events*	(Elliot Estep, 2013)	Membandingkan performa browser ketika menggunakan WebSockets dan *Server-Sent Events* dalam berbagai jaringan *smartphone* (WiFi, 3G dan 4G)	Simulasi	Windows 7 (WebSocket dan HTTP/1.1)	Performa SSE maupun Websocket juga dipengaruhi oleh *web browser* serta konfigurasi jaringan yang digunakan

Penelitian yang dilakukan membandingkan *response time* serta presentase penggunaan CPU antara HTTP/1.1 SSE, HTTPS SSE, HTTP/2 SSE dan WebSocket pada sistem kendali rumah pintar berbasis *web*. *Server* yang digunakan berupa Raspberry Pi 3 serta mampu diakses dari luar jaringan jaringan privat.

3.2 Hipotesis

Berdasarkan kajian dari Tinjauan Pustaka, dapat dibuat hipotesis bahwa HTTP/2 *Server Sent Event* memiliki nilai *response time* terendah sedangkan HTTPS memiliki nilai *response time* tertinggi. Namun dari keempat metode yang digunakan, nilai penggunaan CPU keempatnya relatif sama.

BAB III

METODE PENELITIAN

Pada bab ini dijelaskan mengenai kebutuhan peralatan dan bahan serta perangkat lunak pendukung untuk melakukan pengembangan serta pengujian metode HTTP/1.1 *Server-Sent Events*, HTTP/2 *Server-Sent Events* dan *WebSocket* untuk *website* pada sistem rumah pintar. Selanjutnya dijelaskan juga mengenai metode penelitian dan skenario sistem pengujian.

3.1 Peralatan

Berikut merupakan daftar peralatan yang akan digunakan selama proses penelitian berlangsung :

1. Komputer sebagai *client* dengan spesifikasi:
   - CPU : Intel Celeron 1.50GHz x 4
   - Hardisk : 750 GB
   - RAM : 8 GB
   - OS : Linux Mint 18.2 Cinnamon 64-bit
2. Raspberry Pi 3 Model B sebagai *broker* sekaligus *web server* dan *server API* sebanyak 1 unit dengan spesifikasi:
   - CPU : 1.2 GHz quad-core ARM
   - *Memory* : 1 GB LPDDR2-900 SDRAM
   - USB Port : 4
   - Network : 10/100 Mbps Ethernet, 802.11 n Wireless LAN
3. NodeMCU 1 Unit dengan spesifikasi:

- MCU : Xtensa Dual-Core 32-bit L106
- Wifi : 802.11 b/g/n HT40
- Typical Frequency: 160 MHz
- Tipe ESP : ESP32

4. NodeMCU 1 Unit dengan spesifikasi:
   - MCU : Xtensa Single-Core 32-bit L106
   - Wifi : 802.11 b/g/n HT20
   - Typical Frequency: 80 MHz
   - Tipe ESP : ESP8266
5. *Access Point* sebanyak – 1 unit
6. DHT11 – 2 unit
7. LED Putih – 13 unit
8. Servo SG90 – 2 unit
9. *Dinamo motor* DC 3 volt – 1 unit
10. *Cooling fan* DC 5 volt – 1 unit
11. Kabel Jumper

3.2 Bahan

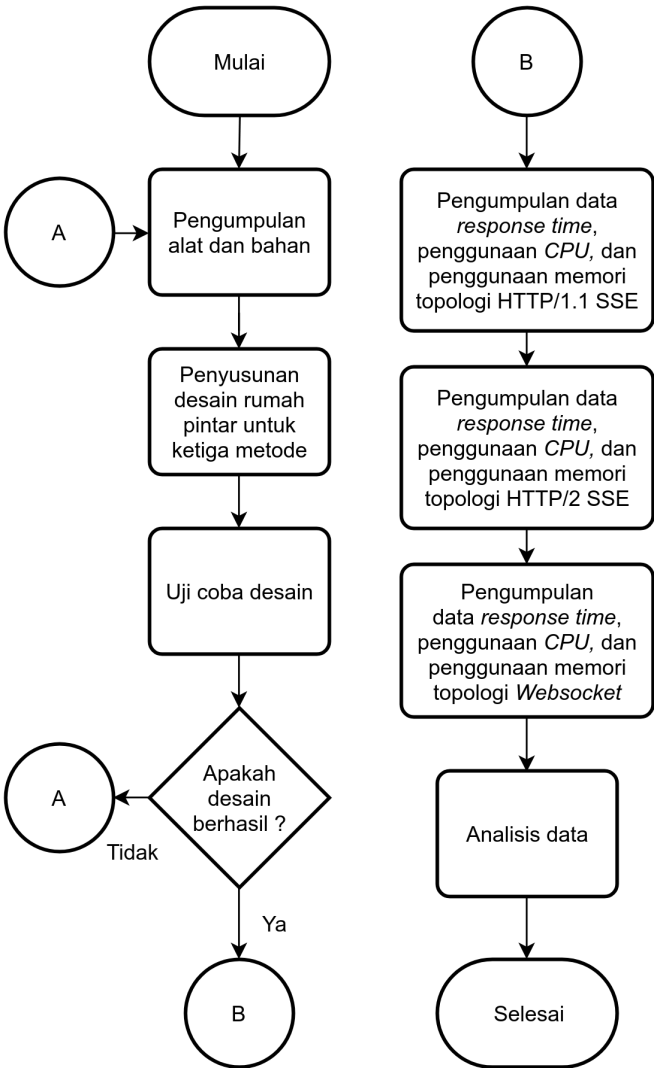
Berikut merupakan daftar perangkat lunak yang akan digunakan selama penelitian ini berlangsung :

1. Mosquitto, sebagai broker untuk protokol MQTT
2. Node.js, sebagai *web server* yang khusus Javascript
3. Vue.js, sebagai javascript framework untuk pembuatan *web application*

4. OpenSSL, untuk pembuatan *public key* dan *private key* pada HTTPS serta HTTP/2
5. Serveo, untuk menjadikan *local server* mampu diakses secara publik
6. Google Chrome, untuk mengakses *website*

3.3 Tahapan Penelitian

Metode penelitian yang dilakukan penulis dalam penelitian ini dapat dilihat pada *flow chart* pada Gambar 3.1.



Gambar 3.1 Diagram alir penelitian

1. Tahap Instalasi.

   Tahap pertama yang dilakukan adalah menginstal semua *software* yang dibutuhkan. Adapun *software* yang di-*install* pada tahap ini adalah Mosquitto, Node.js, Vue.js dan OpenSSL yang akan dipasang pada Raspberry Pi.

2. Tahap Pengembangan Sistem.

   Pada tahap pengembangan sistem, terdapat beberapa fase yang harus dilalui, yakni :

   a. Perancangan Sistem

      Fase perancangan aplikasi memiliki tujuan untuk memberikan gambaran umum mengenai bagaimana proses yang akan berjalan pada sistem yang akan diterapkan.

   b. Pembuatan Sistem

      Fase pembuatan sistem bertujuan untuk mengimplementasikan *WebSocket API* serta *Server-Sent Events API* baik pada sisi *client* maupun *server*. Selain itu, pada fase ini akan diterapkan pula penerapan MQTT di sisi mikrokontroler.

3. Tahap Pengambilan Data

   Di dalam tahap ini akan dilakukan proses pengambilan data untuk metode HTTP/1.1 SSE, HTTPS SSE, HTTP/2 SSE serta Werbsocket yang telah diterapkan di dalam tahap pengembangan sistem. Parameter yang akan digunakan untuk pengambilan data adalah presentase penggunaan CPU serta *response time* untuk setiap metode yang digunakan. Pengujian *response time* dilakukan dengan menghitung waktu antara pengubahan status perangkat yang

dilakukan dari *web browser* sampai dengan mendapatkan balasan status perangkat terbaru dari mikrokontroler.

4. Tahap Analisis.

   Analisis merupakan tahapan paling akhir dalam penelitian ini. Pada tahap ini hasil pengambian data yang telah dilakukan pada tahap sebelumnya akan dikumpulkan, diurai, dibedakan dan dipilah untuk selanjutnya digolongkan dan dikelompokkan berdasarkan kriteria tertentu kemudian dibuat suatu kesimpulan dari hasil yang didapat.

3.4 Instalasi Mosquitto

Instalasi Mosquitto dapat dilakukan dengan cara mengunduh langsung melalui *default repository* Raspberry Pi 3.

```
$ sudo apt update

$ sudo apt install -y mosquitto mosquitto-clients
```

Pada penginstalan di atas, “mosquitto” adalah nama paket *broker* untuk protokol MQTT sedangkan “mosquitto-clients” adalah paket pendukung untuk melakukan percobaan penggunaan protokol MQTT sebagai *client*. Setelah instalasi dilakukan, cek status Mosquitto dengan perintah:

```
$ service mosquitto status
```

Untuk menjalankan Mosquitto secara otomatis setelah Raspberry Pi 3 dihidupkan adalah dengan memasukkan perintah berikut.

```
$ sudo systemctl enable mosquitto.service
```

Setelah Mosquitto aktif, lakukan percobaan menggunakan mosquitto-client. Bukalah terminal baru dan masukkan perintah berikut untuk menjalankan *client* sebagai *subscriber* :

```
$ mosquitto_sub -h localhost -t testing
```

Kemudian buka terminal baru lagi dan masukkan perintah berikut untuk menjalankan *client* sebagai *publisher* :

```
$ mosquitto_pub -h localhost -t testing -m hai
```

Pada kedua perintah di atas, “testing” adalah *topic* yang digunakan untuk oleh protokol MQTT sedangkan “hai” adalah pesan yang dikirimkan oleh *publisher* yang akan diterima oleh *subscriber* dengan *topic* yang sama.

3.5 Instalasi Node.js

Sebelum menginstall Node.js, periksa terlebih dahulu *processor* yang digunakan oleh Raspberry Pi dengan perintah :

```
$ uname -m
```

Sebagai catatan, seluruh Raspberry Pi 3 memiliki processor dengan model ARMv8, sehingga pada penelitian ini diunduh Node.js untuk Linux Binaries (ARM) dengan opsi ARMv8 dari halaman resminya (https://nodejs.org/en/download/). Setelah Node.js berhasil diunduh, masuk pada folder Downloads dan ekstraklah hasil unduhan dengan perintah :

```
$ cd ~/Downloads

$ tar -xzf node-v10.15.3-linux-arm64.tar.xz
```

Kemudian salinlah hasil ekstrak Node.js ke *directory* /usr/local/

```
$ cd node-v10.15.3-linux-arm64
```

```
$ sudo cp -R * /usr/local/
```

Setelah itu, untuk cek apakah instalasi berhasil dengan cara mengecek versi npm dan node yang telah di-*install*,

```
$ npm -v

$ node -v
```

Setelah node dan npm berhasil ter-*install*, *install* modul Vuejs-cli dengan memasukkan perintah:

```
$ npm install -g @vue-cli
```

3.6 Perancangan Topologi dan Model

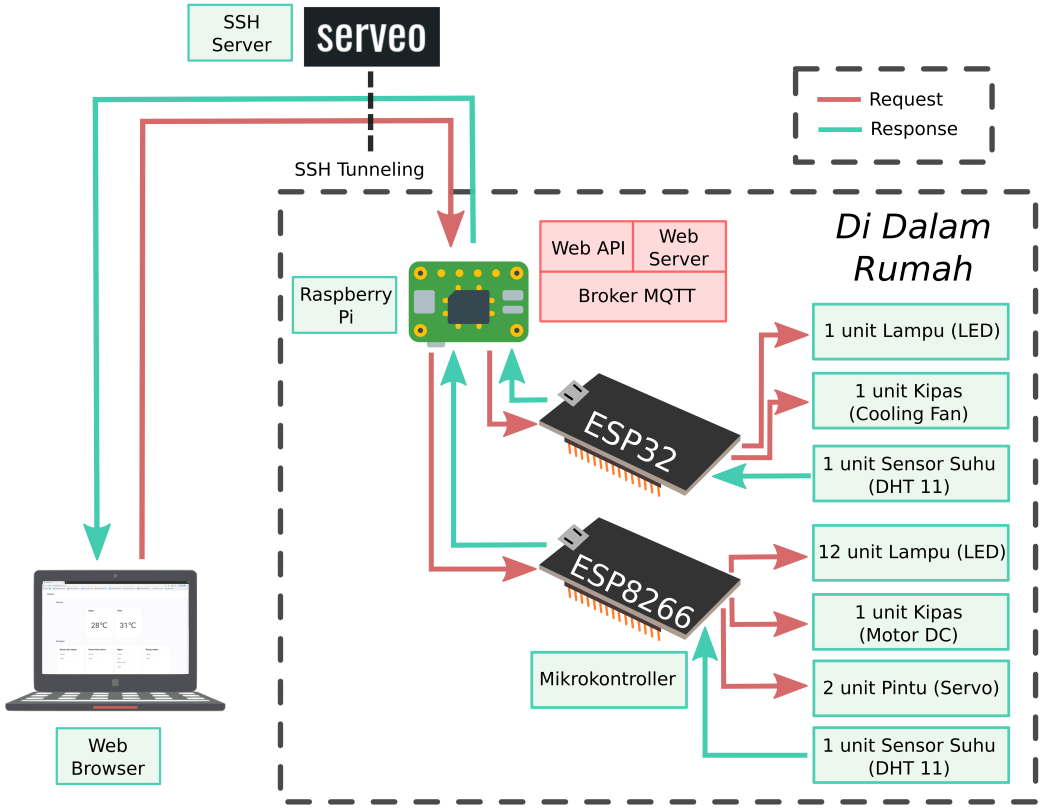
Dalam tahap pengembangan aplikasi, dilakukan dua tahap perancangan, yakni perancangan topologi dan perancangan model.

3.6.1. Perancangan Topologi

Perancangan topologi dibuat dengan memperhatikan gambaran umum proses yang terjadi pada sistem beserta hubungan antar komponen di dalamnya. Perancangan topologi dapat dibagi menjadi perancangan topologi perangkat serta perancangan topologi data. Perancangan topologi perangkat berguna untuk mendeskripsikan alur kerja sistem berdasarkan perangkat yang digunakan maupun dilalui sedangkan perancangan topologi data untuk menggambarkan lalu lintas data yang terjadi pada sistem.

1. Perancangan Topologi Perangkat

Berdasarkan perangkat yang digunakan, terdapat tiga komponen utama yang dibutuhkan supaya sistem mampu berjalan yakni *web browser*, *server* serta mikrokontroler. Untuk lebih lengkapnya dapat diamati pada Gambar 3.2.



Gambar 3.2 Bagan Topologi Perangkat

Web browser merupakan komponen yang berperan sebagai antarmuka pada sistem yang akan dibuat. Data berupa HTML, CSS maupun Javascript yang akan ditampilkan pada *web browser* berasal dari *web server*. Untuk data yang berupa JSON maupun teks seperti halnya suhu berasal dari *web API*.

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".

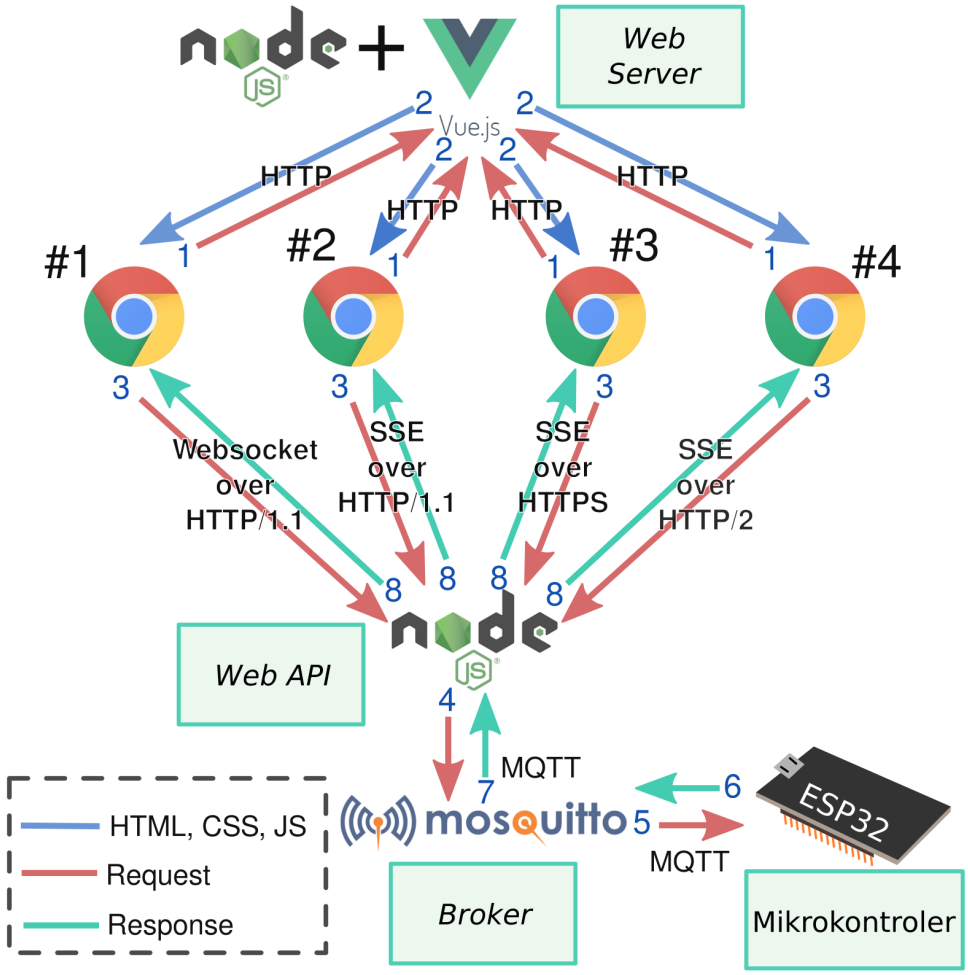
Selanjutnya, terdapat bagian *web API* yang bertugas untuk mengolah data yang berasal maupun menuju MQTT *Broker*. Baik web API maupun web server, keduanya bekerja dengan menggunakan perangkat lunak Node.js. Kemudian data yang telah masuk ke MQTT *Broker* akan diteruskan menuju mikrokontroler .

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, dimana Ngrok membutuhkan biaya tambahan ketika melewatkan data melalui HTTPS, padahal HTTPS diperlukan ketika menggunakan HTTP/2.

Komponen utama terakhir yang dibutuhkan adalah mikrokontroler. Setiap mikrokontroler membutuhkan perangkat yang mampu untuk menghubungkan mikrokontroler dengan jaringan yang akan digunakan, sehingga dalam penelitian ini digunakanlah ESP32 serta ESP8266 sebagai pengirim sekaligus penerima data dari MQTT *Broker*. Selain itu, terdapat pula DHT11 yang digunakan untuk mengukur suhu serta komponen lainnya seperti LED, *motor* maupun kipas yang dapat dikendalikan oleh mikrokontroler.

2. Perancangan Topologi Data

Topologi data berguna untuk menjelaskan alur pengiriman data yang berlangsung selama suatu proses berjalan. Topologi ini juga yang akan digunakan ketika pengujian berlangsung. Untuk lebih detailnya, bagan perancangan topologi data dapat diamati pada Gambar 3.3.



Gambar 3.3 Bagan Topologi Data

Secara keseluruhan terdapat tiga proses utama yang berlangsung selama sistem berjalan yakni pemuatan *web application* di *web browser*, pengiriman perintah dari *web browser* menuju mikrokontroler serta pengiriman data suhu dari mikrokontroler menuju *web browser*. Dari ketiga proses tersebut, pemuatan *web application* merupakan proses yang pertama kali berjalan ketika pengguna berinteraksi dengan sistem.

Proses yang pertama diawali setelah pengguna memasukkan alamat *web server* ke dalam kolom alamat *website* di suatu *web browser* guna meminta pemuatan *web application*. *Web server* yang berjalan menggunakan bantuan

Node.js akan membalas permintaan tersebut dengan mengirimkan dokumen HTML. Dokumen tersebut selanjutnya akan disusun oleh *web browser* untuk ditampilkan. Apabila dalam penyusunan tersebut dibutuhkan dokumen tambahan seperti CSS, Javascript maupun tipe dokumen lainnya, maka web browser akan meminta dokumen tersebut menuju alamat *website* yang telah dicantumkan di dokumen HTML. Alamat *website* tidak harus menggunakan alamat *web server* yang menyediakan *web application*, seringkali beberapa dokumen maupun data dibutuhkan dari alamat lainnya semisal dari *web API.* Hal inilah yang terjadi pada proses pengiriman perintah maupun permintaan data suhu oleh *web browser*.

Dalam penelitian ini, data jumlah perangkat yang digunakan tidaklah tercantum di halaman HTML. Data yang tercantum di halaman tersebut hanyalah data jumlah mikrokontroler serta jumlah sensor yang digunakan. Untuk mendapatkan data jumlah perangkat beserta statusnya perlu dilakukan proses pengiriman perintah dari *web browser*. Hal ini dikarenakan data status perangkat hanya didapatkan setelah perintah diterima oleh mikrokontroler. Untuk menanggulangi hal tersebut dikirimkanlah perintah kosong menuju mikrokontroler setiap kali *web application* dimuat.

Untuk memenuhi kondisi pengujian yang akan dilakukan untuk memperoleh data perbandingan metode pengiriman data, proses pengiriman perintah dapat dilakukan melalui berbagai protokol sesuai dengan URL yang digunakan pada kolom alamat *website.* Protokol yang dicantumkan untuk HTTPS dan HTTP/2 adalah ‘https’, untuk HTTP/1.1 adalah ‘http’ sedangkan untuk WebSocket adalah ‘ws’. Walaupun melalui berbagai metode pengiriman data yang

berbeda, pada dasarnya perintah yang dikirim berisikan data ruangan, perangkat serta status perangkat yang diinginkan oleh pengguna. Selanjutnya, perintah yang telah diterima oleh *web API* akan diteruskan menuju MQTT *Broker* menggunakan susunan data yang lebih sederhana daripada susunan data yang diterima oleh *web API* (JSON). Susunan data ini dibuat supaya memudahkan pengembangan dari sisi mikrokontroler.

Setelah data diterima oleh mikrokontroler, data tersebut akan digunakan untuk menentukan perangkat mana yang statusnya akan diubah. Data status perangkat tersebut kemudian disimpan di mikrokontroler untuk dikimkan kembali setelah status perangkat berhasil diubah sesuai keinginan pengguna. Data yang dikirimkan menuju *web API* tidak hanya data perangkat yang berhasil diubah saja melainkan data status seluruh perangkat yang berada pada mikrokontroler tersebut. Selanjutnya, data tersebut dikirimkan menuju *web API* menggunakan format penyusunan data yang berbeda dengan susunan data yang digunakan untuk menerima data dari *web API*. Susunan dibuat dengan sesederhana mungkin untuk menanggulangi keterbasan jumlah karakter yang mampu dikirimkan dari mikrokontroler. Setelah data tersebut diterima oleh *web API*, data tersebut akan diteruskan menuju *web browser* menggunakan susunan data JSON. Terakhir, data tersebut kemudian diolah oleh web browser sehingga mampu untuk dilihat serta digunakan kembali oleh pengguna.

Berbeda dengan proses pengiriman perintah, proses pengiriman suhu dilakukan dari mikrokontroler menuju *web browser* tanpa membutuhkan permintaan data terlebih dahulu. Hal ini dapat terjadi dikarenakan keempat

metode mampu membuka koneksi dengan rentang waktu yang cukup lama. Proses pembacaan sensor suhu yang dilakukan oleh DHT11 dilakukan setiap beberapa detik. Setelah itu, data suhu kemudian dikirimkan menuju web API untuk diolah dan dikirimkan menuju *web browser*.

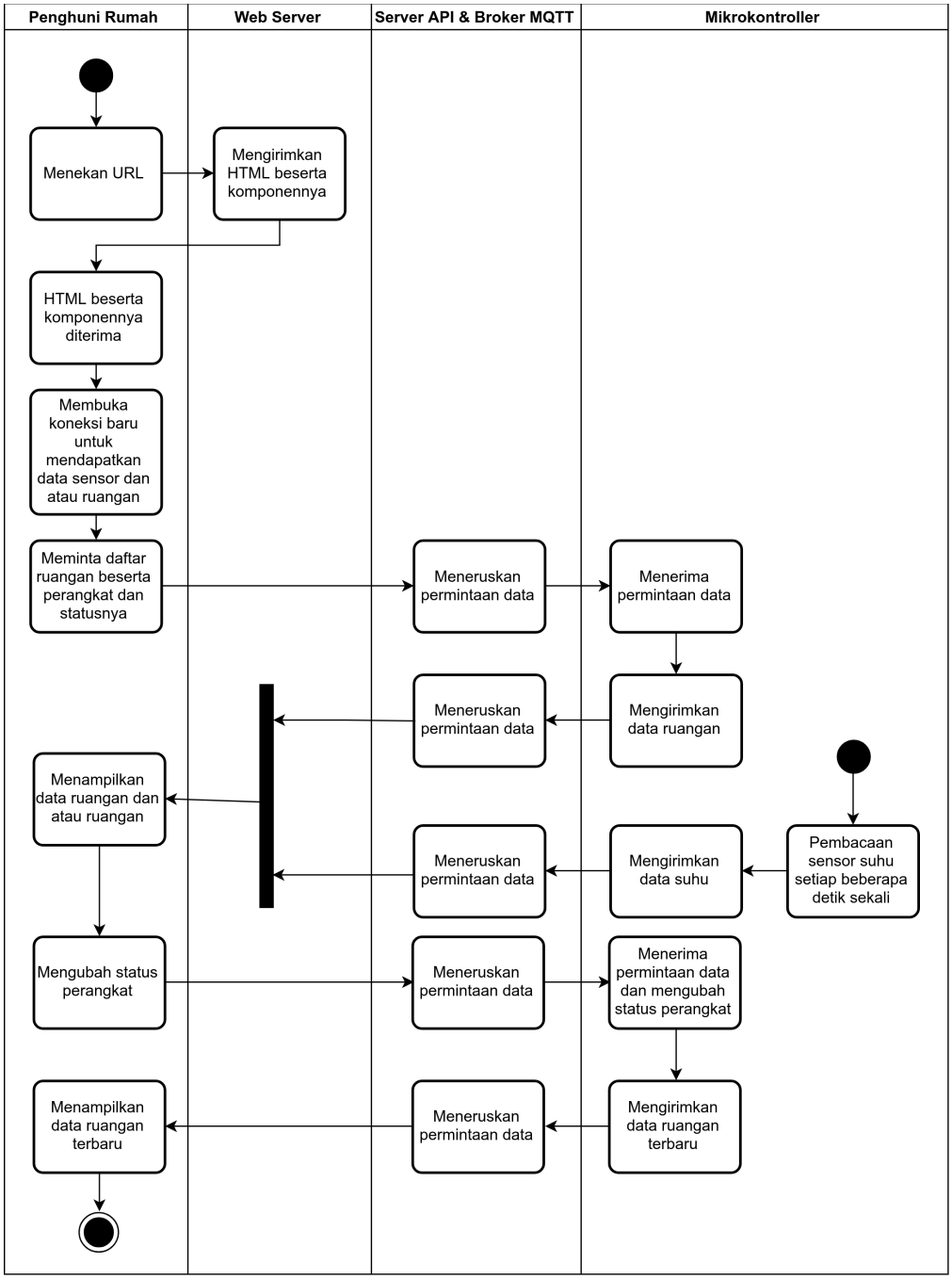
Dari ketiga proses tersebut, web API memiliki peran yang penting untuk mengolah data terutama data yang diterima dari mikrokontroler. Data yang berasal dari mikrokontroler seperti halnya suhu maupun status perlu dikirimkan secara langsung dari web API menuju web browser melalui koneksi TCP yang telah terbuka. Untuk memisahkan kedua data agar tidak saling terpisah digunakanlah *path* tertentu untuk setiap data pada suatu URL.

3.6.2. Perancangan Model

Perancangan model dibuat untuk menggambarkan alur pemakaian sistem oleh pengguna. 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 perancancangan model adalah *use case diagram* dan *activity diagram*.

1. *Activity Diagram*

   Activity Diagram digunakan untuk menggambarkan aliran kerja atau langkah-langkah yang ditempuh selama sistem berjalan. Dalam perancangan *activity diagram*, keseluruhan *activity* yang terdapat di dalam sistem ditampilkan sesuai dengan tiga komponen utama yang terdapat pada aplikasi. Pada Gambar 3.3 terlihat *activity diagram* dari sistem yang akan dibuat.

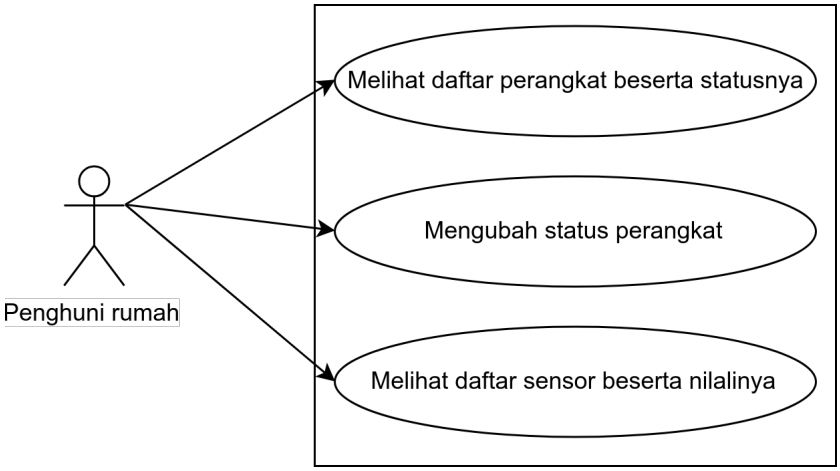


Gambar 3.3 *Activity Diagram* sistem

2. *Use Case Diagram*

Use Case Diagram adalah diagram UML yang ditujukan untuk menggambarkan skenario fungsi dari sistem yang dikembangkan. Tujuan utama

penggunaan *use case diagram* adalah untuk membuat visualisasi dari fungsi yang dibuat dalam suatu sistem. Gambar 3.4 merupakan *use case diagram* dari *web server* yang akan dibuat.



Gambar 3.4 ***Use case diagram web server***

3.7 Pembuatan *Web API* serta *Web Server*

Proses pengiriman data menggunakan metode *WebSocket* berbeda dengan ketika menggunakan *Server-Sent Events* baik di sisi *server* maupun di sisi *web browser*. Hal ini dikarenakan WebSocket mampu untuk melakukan pengiriman data dari dua arah (*bidirectional*), namun tidak dengan *Server-Sent Events*. Selain itu untuk mengirim data menggunakan protokol *WebSocket* di *Javascript* diperlukan *WebSocket API* sedangkan untuk metode *Server-Sent Events* membutuhkan *EventSource API*.

3.7.1 Penerapan Node.js serta Vue.js

Sebelum menerapkan *WebSocket* maupun *Server-Sent Events* di sisi server, perlulah dibuat proyek Node.js baru. Node.js sebagai *Javascript Framework* diperlukan untuk mempermudah pembuatan *Web API* maupun *web server*. Untuk

membuat proyek baru yang akan digunakan sebagai *Web API* dimasukkanlah perintah :

```
$ mkdir webapi && cd webapi

$ touch index.js

$ npm init
```

Setelah itu, masukkan nama beserta *metadata* npm lainnya. *Metadata* npm masih dapat diubah setelah proyek terbentuk dengan cara mengubah lalu konten yang ada pada "package.json".

Setelah *Web API* selesai dibuat, dibuatlah proyek baru yang akan digunakan sebagai *web server* digunakanlah perintah :

```
$ vue create webapp
```

Kemudian pilihlah pilihan "*Manually select features*" dan dilanjutkan dengan memilih fitur Babel dan Router. Babel merupakan *javascript compiler* yang memungkinkan berbagai *web browser* untuk menggunakan ECMAScript2015 keatas, mengubah ekstensi vue maupun jsx sehingga mampu untuk dibaca oleh web browser serta melakukan *polyfill*. Untuk fitur Router berguna untuk memudahkan pemetaan alamat (URL) pada "Vue.js".

3.7.2 Penerapan *WebSocket* dan *Server-Sent Events*

Untuk menggunakan *WebSocket API* di *web browser* dibutuhkanlah *server* yang mampu digunakan sebagai *WebSocket server.* Node.js sebagai *web API* telah mampu untuk dijadikan *WebSocket server* menggunakan *module* (*library*)

tambahan yang bernama ‘WebSocket’. Module ini dapat diunduh melalui npm setelah Web API dibuat dengan memasukkan perintah :

```
$ cd webapi

$ npm install WebSocket
```

Penggunaan *module* ‘WebSocket’ dapat dilihat pada dokumentasinya ataupun pada halaman Lampiran 1. Berbeda dengan di *Web API*, penerapan *WebSocket* di *web server* dapat menggunakan *WebSocket API* yang telah disediakan oleh HTML5, sehingga tidak diperlukan *module* tambahan.

Sama halnya dengan *WebSocket*, dalam penerapan *Server-Sent Events* pada sisi *server* dibutuhkan *module* tambahan yakni ‘express’ dan ‘http2’. Module ‘express’ berguna untuk mempermudah pembuatan *REST API* melalui HTTP/1.1 di dalam Node.js, sedangkan module ‘http2’ berguna untuk pembuatan *REST API* melalui HTTP/2. Untuk penerapan *Server-Sent Events* di dalam *web server* menggunakan *EventSource API* yang telah dimiliki oleh sebagian besar *web browser*.

3.7.3 Penerapan TLS

Tanpa menggunakan TLS, HTTP/2 tidak dapat digunakan. TLS dapat dibuat menggunakan bantuan OpenSSL. Untuk meng-*generate self-signed certificate* di dalam sistem operasi Linux Mint, dimasukkan perintah sebagai berikut :

```
$ openssl req -x509 -newkey rsa:4096 -keyout

  secret.key -out secret.crt -days 365
```

3.8 Metode Pengambilan Data

Tujuan dari penelitian ini dilakukan adalah untuk membandingan beberapa metode pengiriman data di dalam kasus rumah pintar. Metode pengiriman data yang dipilih 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.

3.8.1 *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 pesan balasan diterima dari sisi *web browser*. Untuk proses penandaan waktu digunakanlah *instance* Date yang dimiliki oleh Javascipt, salah satu fungsi *instance* Date adalah mengambil data waktu saat ini dari komputer pengguna dengan format awal ISO 8601.

Proses pengambilan *response time* dilakukan dalam dua kondisi yang berbeda, yakni pada jaringan privat yang sama serta pada jaringan privat yang berbeda. Untuk kedua kondisi tersebut penulis menggunakan jaringan privat indihome untuk mengakses *server* dari *web browser*, hanya saja untuk memenuhi kondisi yang kedua digunakanlah aplikasi pihak ketiga, yakni Serveo. Serveo adalah aplikasi yang berguna untuk melakukan *SSH Tunneling*, yang mana

memungkinkan seseorang untuk mengakses komputer yang berbeda jaringan melalui port tertentu.

3.8.2 Penggunaan CPU

Pengambilan data presentase penggunaan CPU dilakukan dengan menggunakan bantuan aplikasi ‘top’ yang merupakan aplikasi bawaan dari sistem operasi Linux Mint. Perintah yang dimasukkan untuk pengambilan data dengan menggunakan aplikasi ‘top’ adalah sebagai berikut :

```
$ top | grep <pid>
```

Dimana <pid> adalah id dari proses yang berjalan dalam sistem komputer, dalam kasus ini pid yang digunakan adalah pid dari ‘node’. Pemilihan pid ketimbang ‘node’ dikarenakan ketika aplikasi ‘node’ berjalan, terdapat beberapa aplikasi ‘node’ yang muncul di aplikasi ‘top’ sehingga dipilihlah pid dari ‘node’ dengan presentase penggunaan CPU-nya naik maupun turun selama pengujian.

Skenario pengujian presentase penggunaan CPU dilakukan dengan cara mengirimkan perintah setiap detiknya dari beberapa *web browser* menuju mikrokontroler secara bersamaan, dimana 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. Selain menerima balasan dari mikrokontroler, *web browser* juga menerima dua data suhu yang dikirimkan melalui *stream* yang berbeda antar keduanya maupun dengan *stream* yang digunakan untuk menerima pesan dari

mikrokontroler. Untuk jumlah *web browser* yang digunakan akan bervariasi selama pengujian.

Dua w*eb browser* yang sama ketika dibuka pada waktu yang bersamaan akan menggunakan *session*, *history*, maupun *cache* yang sama. Hal ini tidak mengganggu pengujian WebSocket maupun HTTP/2 SSE yang memiliki karakteristik paralel, namun tidak dengan 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
```

BAB IV

HASIL PENELITIAN DAN PEMBAHASAN

Pada bab ini dijelaskan mengenai hasil yang didapatkan setelah dilakukan penelitian sesuai dengan yang telah diterangkan pada bab sebelumnya. Hasil-hasil yang didapatkan berupa sistem kendali rumah pintar berbasis *web* beserta perbandingan response time beserta presentase penggunaan CPU antara keempat metode pengiriman yang telah digunakan.

4.1 Hasil Pengembangan Sistem

Aplikasi antarmuka yang dikembangkan pada penelitian ini merupakan sebuah aplikasi *web-based*. Pengembangan dilakukan dengan bahasa Javascript, begitu juga dengan pengembangan untuk *web API*.

4.1.1 Tampilan Antarmuka

Antarmuka yang digunakan pada sistem rumah pintar berupa sebuah *web server* yang dapat ditampilkan melalui suatu *web browser*. Antarmuka terdiri dari dua bagian, yakni kumpulan data sensor serta kumpulan data status setiap perangkat yang terpasang pada masing-masing mikrokontroler.

4.2 Hasil Pengujian

Terdapat dua parameter yang diujikan untuk membandingkan performa pengiriman data dua arah yang dilakukan oleh HTTP/1.1 SSE, HTTPS SSE, HTTP/2 SSE dan WebSocket yakni response time serta presentase penggunaan CPU.

DAFTAR PUSTAKA

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

Hasibuan, Zainal A. 2007. *Metodologi Penelitian Pada Bidang Ilmu Komputer Dan Teknologi Informasi, Konsep, Metode Teknik, Dan Aplikasi*. 2011.

Hillar, Gastón C. 2017. *MQTT Essentials : A Lightweight IoT Protocol : The Preferred IoT Publish-Subscribe Lightweight Messaging Protocol*. Birmingham: Packt.

Abyl. 2018. "Long Polling - Concepts and Considerations." 2018. https://www.ably.io/concepts/long-polling.

Briere, Danny, and Pat Hurley. 2011. *Smart Homes For Dummies*. New Jersey: John Wiley & Sons.

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

Elman, Julia, and Mark Lavin. 2014. *Lightweight Django: Using REST, WebSockets, and Backbone - Julia Elman, Mark Lavin*. California: O'Reilly Media, Inc.

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

Hillar, Gastón C. 2017. *MQTT Essentials : A Lightweight IoT Protocol : The Preferred IoT Publish-Subscribe Lightweight Messaging Protocol*. Birmingham: Packt.

Ibrahim, Dogan. 2014. "A New Approach for Teaching Microcontroller Courses to Undergraduate Students." *Procedia - Social and Behavioral Sciences* 131 (May): 411–14.

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.

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.

Leach, Paul J., Tim Berners-Lee, Jeffrey C. Mogul, Larry Masinter, Roy T. Fielding, and James Gettys. 1999. “Hypertext Transfer Protocol -- HTTP/1.1.”

Ludin, Stephen, and Javier Garza. 2017. *Learning HTTP/2: A Practical Guide for Beginners - Stephen Ludin, Javier Garza*. Sebastopol: O’Reilly Media, Inc.

MDN. 2019. “Protocol Upgrade Mechanism - HTTP | MDN.” 2019. https://developer.mozilla.org/en- US/docs/Web/HTTP/Protocol_upgrade _mechanism.

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.

Örnmyr, Oliver, and Rasmus Appelqvist. 2017. “Performance Comparison of XHR Polling , Long Polling , Server Sent Events and Websockets.” Blekinge Institute of Technology.

Peon, R., and H. Ruellan. 2015. “HPACK: Header Compression for HTTP/2.” https://doi.org/10.17487/RFC7541.

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.

Rhee, Kyung-Hyune, and Jeong Hyun Yi. 2015. *Information Security Applications: 15th International Workshop, WISA 2014*. Berlin: Springer.

Rochman, Hudan Abdur, Rakhmadhany Primananda, and Heru Nurwasito. 2017. “Sistem Kendali Berbasis Mikrokontroler Menggunakan Protokol MQTT Pada Smarthome.” *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer* 1 (6): 445–55.

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

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

Udayashankara, V, and M S Mallikarjunaswamy. 2009. *Microcontroller*. New Delhi: Tata McGraw-Hill Education.

Vasseur, Jean-Philippe, Adam Dunkels, Jean-Philippe Vasseur, and Adam Dunkels. 2010. "What Are Smart Objects?" *Interconnecting Smart Objects with IP*, January, 3–20.

Wang, Vanessa., Frank. Salim, and Peter. Moskovits. 2013. *The Definitive Guide to HTML5 WebSocket*. Apress.