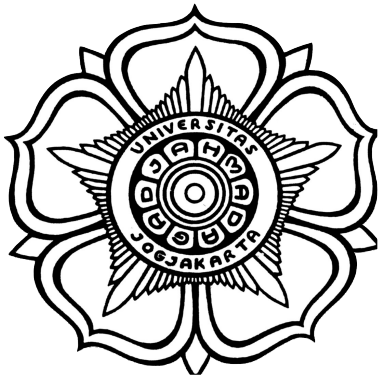
LAPORAN PROYEK AKHIR

ANALISIS DAN PENERAPAN 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

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	:	Analisis dan Penerapan 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*
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

ANALISIS DAN PENERAPAN 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 rumah pintar yang terhubung langsung dengan *web browser* pengguna. Parameter yang diujikan adalah *throughput* dan *latency*.

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 lah 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 mikrokontroller 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 mikrokontroller 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 SSE HTTP/1.1, SSE HTTPS, SSE HTTP/2 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 SSE HTTP/1.1, SSE HTTPS, SSE HTTP/2 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 :

1. BAB I PENDAHULUAN

   Memuat latar belakang masalah, perumusan masalah, batasan masalah, tujuan penulisan, kegunaan penulisan dan sistematika penulisan.

2. BAB II TINJAUAN PUSTAKA

   Tinjauan pustaka menurut 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.

3. BAB III BAHAN DAN METODE PENELITIAN

   Memuat bahan, peralatan, tahapan penelitian, dan rancangan sistem serta analisis data yang ada pada penelitian ini.

4. BAB IV ANALISIS HASIL DAN PEMBAHASAN

   Bagian ini 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.

5. BAB V PENUTUP

   Bagian ini memuat kesimpulan serta saran dari penelitian proyek akhir ini.

BAB II

TINJAUAN PUSTAKA & HIPOTESIS

2.1 Mikrokontroller

Microprocessor telah banyak membantu pekerjaan manusia, baik dalam hal pengendalian suatu peralatan maupun pemantauan. *Microprocessor* dapat diibaratkan sebagai otak dari suatu komputer, yang berisi CU (*Control Unit*) dan ALU (*Aritmetic and Logic Unit*). 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. Disamping 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 mikrokontroller, 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, mikrokontroller 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).

2.2 Konsep *Internet of Things*

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. 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, suatu perangkat dengan perangkat lainnya tidaklah harus terhubung pada *Access Point* yang sama.

Contoh dari penerapan rumah pintar adalah seperti yang telah dilakukan oleh Hudan Abdur dalam penelitiannya mengenai rumah pintar. Pada penelitian tersebut dilakukan pembuatan serta analisis pada sistem kendali berbasis mikrokontroler pada rumah pintar melalui perangkat nirkabel ESP8266 dengan menggunakan protokol MQTT. Penelitian tersebut mengujikan 3 skenario waktu jeda pengiriman data untuk mendapatkan hasil dari Delta time, Rata-rata jumlah paket yang dikirimkan perdetik serta presentase jumlah pengiriman paket perdetik (Rochman, Primananda, and Nurwasito 2017).

2.4 *Message Queuing Telemetry Transport*

Dalam proses pengiriman data dari mikrokontroller menuju server, terdapat banyak pilihan protokol yang dapat digunakan. Walaupun demikian, daam penyusunan rumah pintar dibutuhkan protokol yang memiliki proses pengiriman yang cepat dan tepat.

Dalam jurnal penelitian yang dibuat oleh Paridhika Kayal yang dilakukan perbandingan *response time* untuk protokol MQTT, CoAP, XMPP dan MQTT melalui Websocket. Jurnal ini menunjukkan bahwa protokol MQTT memiliki rata-rata *response time* paling rendah dalam kondisi pengguna *resource* CPU yang terus naik (Kayal and Perros 2017). Dengan adanya hasil tersebut, dipilihlah MQTT untuk digunakan sebagai protokol pengiriman data dari mikrokontroller ke *server*.

MQTT (*Message Queuing Telemetry Transport*) adalah protokol yang ringan dan bekerja dengan sistem *publish-broker-subscribe*. Protokol MQTT bekerja di atas protokol TCP/IP (*Transmission Control Protocol* / *Internet Protocol*). MQTT sangat cocok digunakan untuk pengiriman data yang bersifat *real-time* (Hillar 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.

2.5 *Binary Protocol* dan *Plain Text Protocol*

Protokol dapat dikategorikan menjadi dua, yakni *Plain Text Protocol* dan *Binary 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. Selain itu, data yang dikirimkan melalui *Binary Protocol* cenderung lebih ringan daripada ketika dikirmkan melalui *Plain Text Protocol*. Contoh dari protokol yang termasuk kategor *Binary Protocol* adalah HTTP/2 serta *Websocket* (Rhee and Hyun Yi 2015).

2.6 *Hypertext Transfer Protocol*

HTTP (*Hypertext Transfer Protocol*) telah digunakan oleh *World-Wide Web* sebagai media pengiriman informasi sejak 1990. Diawali dengan HTTP/0.9 yang berupa protokol sederhana untuk pengiriman *raw data* di internet. Dilanjutkan dengan kemunculan HTTP/1.0 yang memungkinkan pengiriman informasi dalam format seperti MIME. Sayangnya, HTTP/1.0 tidak sesuai dengan jaringan yang memakai *proxy*, *caching*, koneksi yang bertahan lama ataupun *virtual host (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)*.

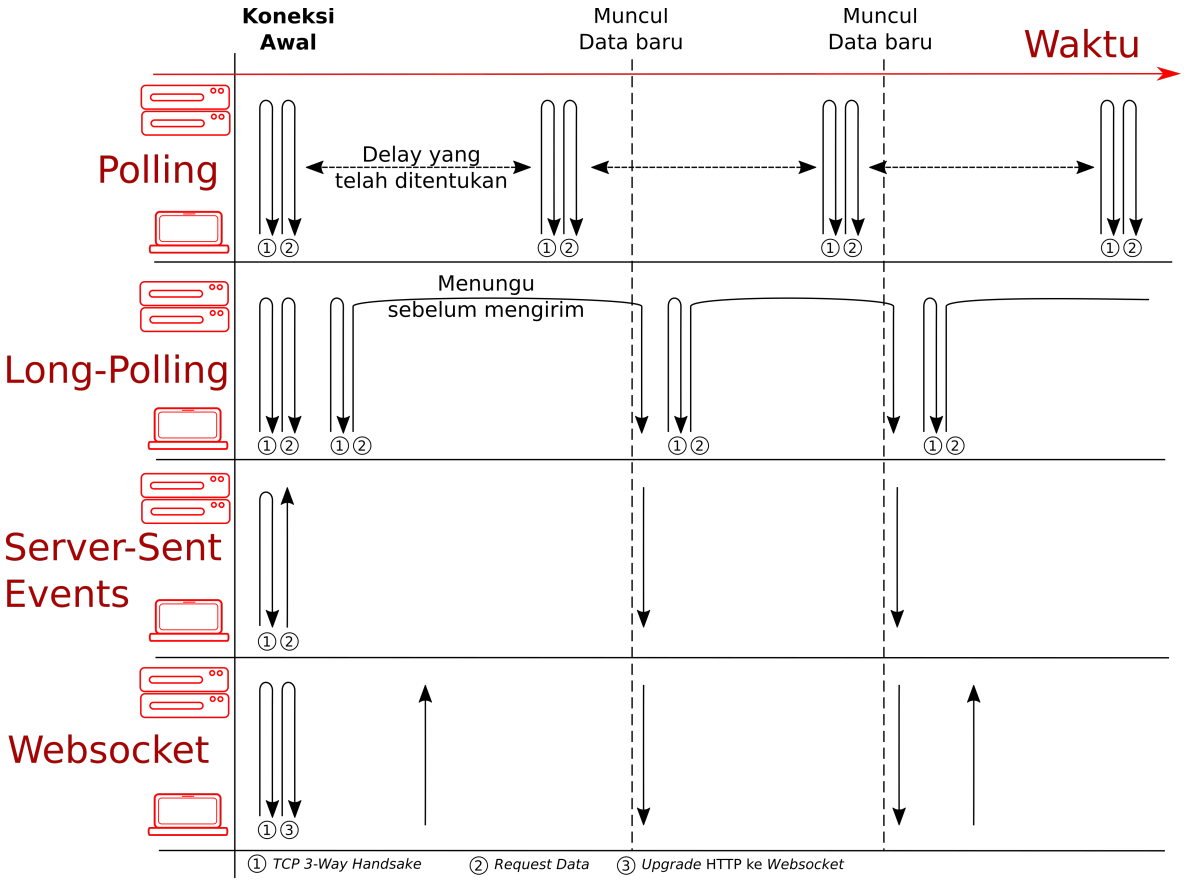
Setelah HTTP/1.1 bertahan cukup lama, muncullah HTTP/2 yang memungkinkan penggunaan jaringan yang lebih efisien dengan melakukan pemampatan *header* menggunakan HPACK . Selain itu, HTTP/2 mampu menangani *Head of Blocking* yakni

kejadian dimana ketika data yang diterima dari *server* harus antri untuk bisa dimuat pada halaman HTML, dengan kata lain HTTP/2 benar-benar *multiplexed* dimana beberapa data yang dikirim maupun diterima dapat dilakukan dalam satu waktu tanpa saling menghalangi. 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*. Untuk membangun rumah pintar, perlu dipilihlah metode yang membutuhkan penggunaan memori dan CPU serendah mungkin dan juga tidak memakan banyak *bandwith*. Hal ini disebabkan perangkat yang akan digunakan sebagai *server* seharusnya memiliki harga yang terjangkau.

Pengujian penggunaan memori dan CPU pada keempat metode tersebut pernah diteliti oleh Rasmus Appelqvist dan Oliver Örnmyr. Pada penelitian tersebut dilakukan pengujian penggunaan memori dan CPU dari 100 perangkat virtual yang terhubung dengan server menggunakan *XHR Polling*, *Long-Polling* , *Server-Sent Events* dan Websockets. Dan didapatkan hasil bahwa dari keempat perangkat tersebut *Server-Sent Events* dan Websocket memiliki nilai penggunaan memori dan CPU terendah serta perbedaan diantara keduanya sangat tipis (Örnmyr and Appelqvist 2017). Untuk cara kerja masing-masing metode dapat diamati pada Gambar 2.1.

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.



Gambar 2.1 Berbagai metode pengiriman data ke *web browser*

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 terjadinnya 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).

2.6 *Server-Sent Events*

Metode lain yang dapat digunakan selain *Polling* dan *Long Polling* adalah *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,* metode ini tidak perlu melakukan permintaan data untuk setiap data baru yang muncul pada *server* sehingga dapat mengurangi penggunaan *bandwith* berlebih. 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 batasan ini pun menghilang. Sebagai catatan tambahan, Internet Explorer tidak mendukung *Server-Sent Events* (Elman and Lavin 2014).

2.7 *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 W3C (*World Wide Web Consortium*) sedangkan untuk protokol *Websocket* dikembangkan oleh IETF (*Internet Engineering Task Force*). 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). Sebagai catatan tambahan, protokol *Websocket* telah mampu digunakan oleh sebagian besar *desktop browsers* yang terkenal kecuali *Internet Explorer (Elman and Lavin 2014)*. Selain itu, penggunaan protokol Websocket secara langsung tanpa menggunakan (TLS/SSL) rentan terhalangi oleh *proxy server*, seperti yang telah diteliti di dalam jurnal "*A Real-time Application Framework for Speech Recognition Using HTTP/2 and SSE*". Dimana dalam penelitian tersebut didapatkan hasil bahwa *Websocket* lebih rentan untuk dihalangi oleh *proxy server* daripada HTTP/1.1 *Server-Sent Events.* Namun hal ini tidak berlaku pada *Secure Websocket* maupun HTTP/2 dikarenakan data telah terenkripsi dengan TLS/SSL (Ramli, Jarin, and Suryadi 2018).

2.8 *Response Time*

Response Time merupakan parameter yang digunakan untuk mengukur waktu yang dibutuhkan oleh *web browser* untuk mendapatkan balasan dari server. Dalam kasus rumah pintar, menghitung *response time* berarti menghitung lamanya waktu yang dibutuhkan dari

saat pengguna mengubah status perangkat dari *web browser* sampai mendapatkan status perangkat terbaru dari mikrokontroller.

Terdapat beberapa faktor yang mampu mempengaruhi nilai *response time,* yakni infrastruktur jaringan dan lama pengolahan data. Ketika dua *web browser* berada pada infrastruktur jaringan yang berbeda, ada kemungkinan untuk mendapatkan hasil *response time* yang berbeda pula. Hal ini pernah diteliti pada penelitian yang berjudul "*Mobile HTML5: Efficiency and Performance of WebSockets and Server-Sent Events*". Penelitian tersebut menguji performa browser ketika menggunakan Websockets dan *Server-Sent Events* dalam berbagai jaringan *smartphone* (WiFi, 3G dan 4G). Hasil dari penelitian tersebut adalah performa konektivitas Websocket dan *Server-Sent Events* tergantung dengan browser dan konfigurasi jaringan yang digunakan (Estep 2013).

Metode yang digunakan untuk pengiriman data juga berpengaruh pada nilai *response time.* Penelitian untuk menguji delay 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 *Webscoket* (Muhammad, Yahya, and Basuki 2018).

Tabel 2.1 Ringkasan tinjauan pustaka

No	Judul Penelitian	Penulis & Tahun	Metode	Tipe Penelitian	Pengiriman Data
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)
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)
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)
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)
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* ,	Simulasi	Ubuntu (HTTP/1.1 dan Webscoket)

			Server-Sent Events dan Websockets		
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)

Untuk melengkapi penelitian yang sebelumnya yang pernah dilakukan, penelitian ini ditujukan untuk membandingkan *response time* serta presentase penggunaan CPU pada SSE HTTP/1.1, SSE HTTPS, SSE HTTP/2 serta Websocket pada rumah pintar. 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 – 1 unit
7. LED Putih – 1 unit
8. Kabel Jumper

3.2 Bahan

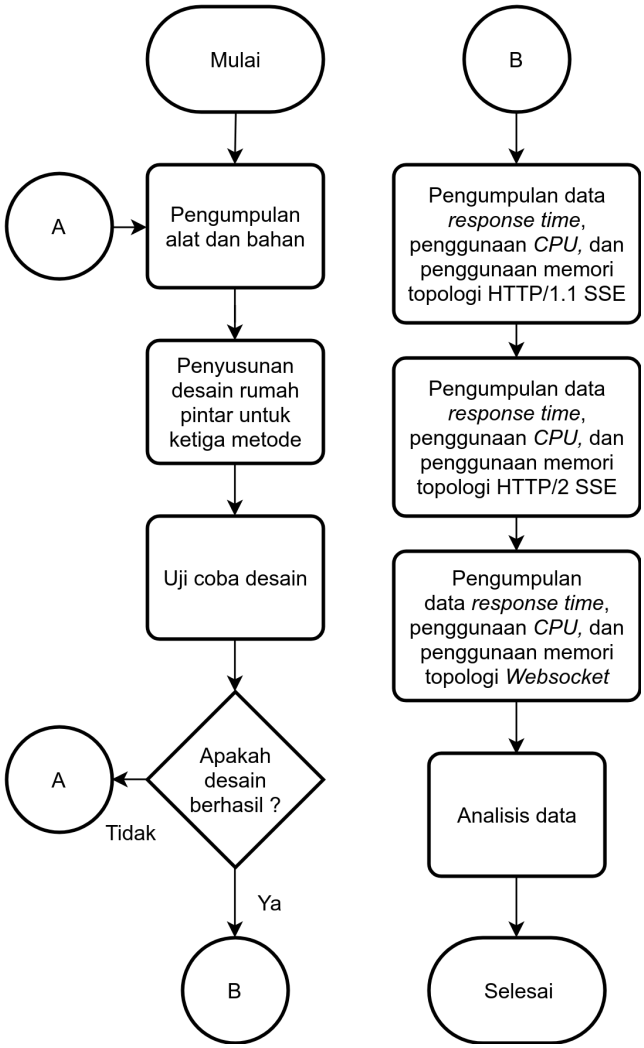
Berikut merupakan daftar perangkat lunak yang akan digunakan selama penelitian ini berlangsung :

1. Mosquitto, sebagai broker untuk protokol MQTT
2. Nodejs, sebagai javascript framework untuk pembuatan *server API*
3. Vuejs, sebagai javascript framework untuk pembuatan *web server*
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* di bawah ini.



Gambar 3.1 Diagram alir penelitian

1. Tahap Instalasi.

   Tahap pertama yang dilakukan adalah menginstal semua software yang dibutuhkan. Adapun software yang diinstall pada tahap ini adalah Mosquitto, Nodejs, Vuejs 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 mikrokontroller.

3. Tahap Pengambilan Data

   Di dalam tahap ini akan dilakukan proses pengambilan data untuk metode SSE HTTP/1.1, SSE HTTPS, SSE HTTP/2 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 mikrokontroller.

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 diunduhlah Node.js untuk Linux Binaries (ARM) dengan opsi ARMv8 dari halaman resminya

(https://nodejs.org/en/download/). Setelah Node.js berhasil diunduh, masuklah pada folder Download 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 diinstal,

```
$ npm -v
$ node -v
```

Setelah node dan npm berhasil terinstal, installah modul Vuejs-cli dengan memasukkan perintah:

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

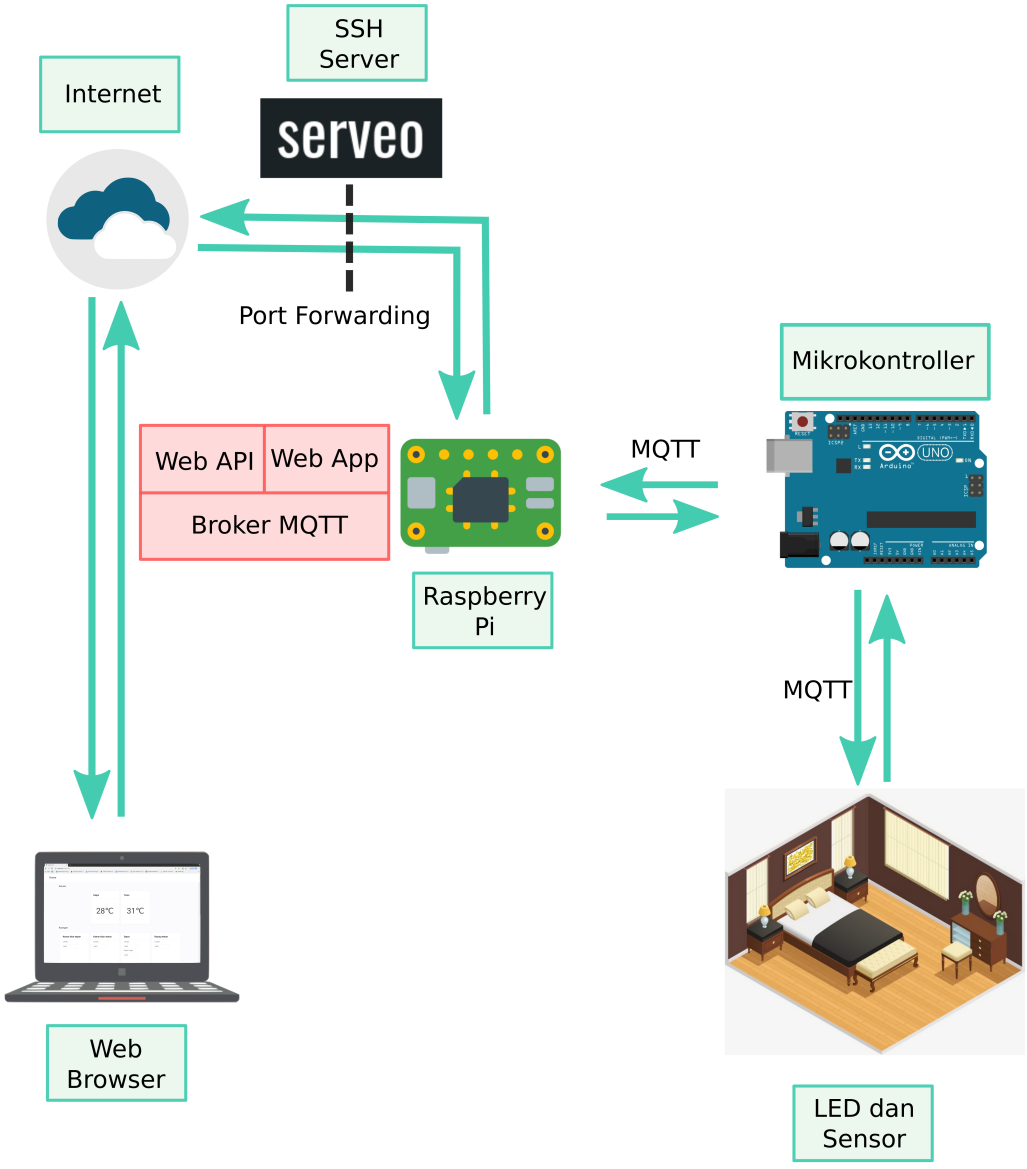
3.6 Perancangan Topologi dan Model

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. Di dalam sistem terdapat tiga komponen utama, yakni *web browser*, *server* serta mikrokontroller. Untuk lebih lengkapnya dapat diamati pada gambar di bawah ini.



Gambar 3.2 Alur kerja sistem

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 application.* Untuk data yang berupa JSON maupun teks seperti halnya suhu berasal dari *web API*.

Komponen server dapat dibagi menjadi tiga bagian, yakni *web application, web API*, serta Broker MQTT. *Web application* berperan untuk mengirimkan data HTML, CSS maupun javascript yang 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 Vuejs. Selanjutnya, terdapat bagian *web API* yang bertugas untuk mengolah data yang berasal maupun menuju Broker MQTT. Data yang telah diterima dari Broker MQTT akan diterima oleh *web browser* melalui salah satu dari keempat metode pengiriman yang telah ditetapkan. Keempat metode pengiriman tersebut adalah Websocket, HTTP/1.1 *Server-Sent Events* serta HTTP/2 *Server-Sent Events*. Web API yang digunakan pada penelitian ini adalah Node.js. Selanjutnya, bagian Broker MQTT yang berperan dalam meneruskan pesan dari *web API* menuju mikrokontroller maupun sebaliknya. Dalam kasus rumah pintar ini, Broker MQTT yang digunakan adalah Mosquitto.

Dalam penerapannya, *server* tanpa menggunakan Serveo hanya dapat diakses pada jaringan lokal. Serveo adalah *server* SSH yang digunakan untuk *port forwarding* dari suatu komputer sehingga mampu untuk diakses dari luar jaringan. Terdapat pula pihak ketiga yang serup 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.

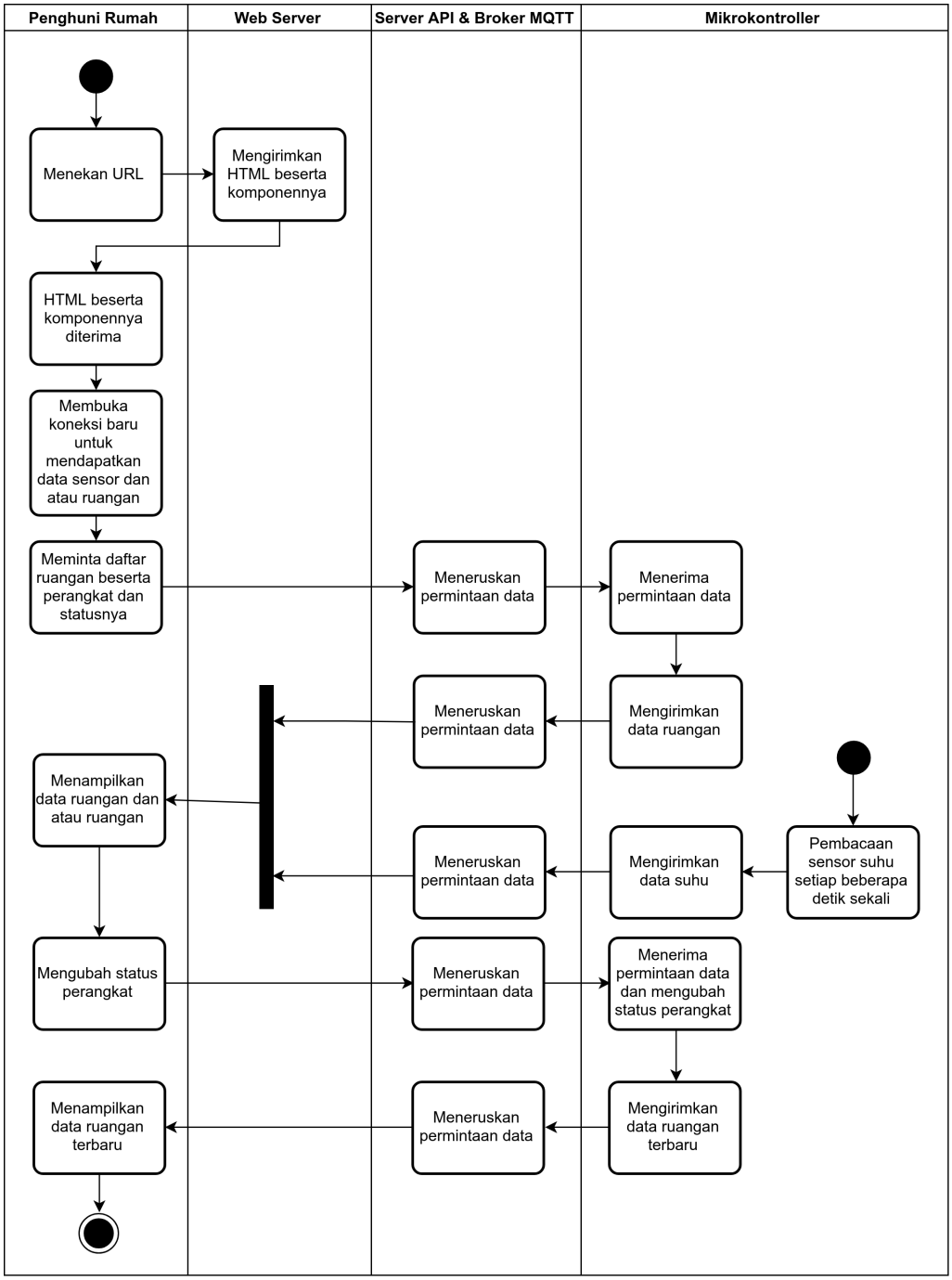
Komponen utama terakhir yang dibutuhkan adalah mikrokontroler. Setiap mikrokontroler membutuhkan perangkat yang mampu untuk menghubungkan mikrokontroler dengan jaringan yang akan digunakan, sehingga dalam penelitian ini digunakanlah ESP32 sebagai pengirim sekaligus penerima data dari Broker MQTT. Selain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

3.6.2 Perancangan Model

Perancangan model dibuat untuk menggambarkan alur pemakaian sistem oleh pengguna. Dalam pembuatannya, perancangan model dibuat menggunakan UML Diagram. 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 di bawah ini terlihat *activity diagram* dari sistem yang akan dibuat.

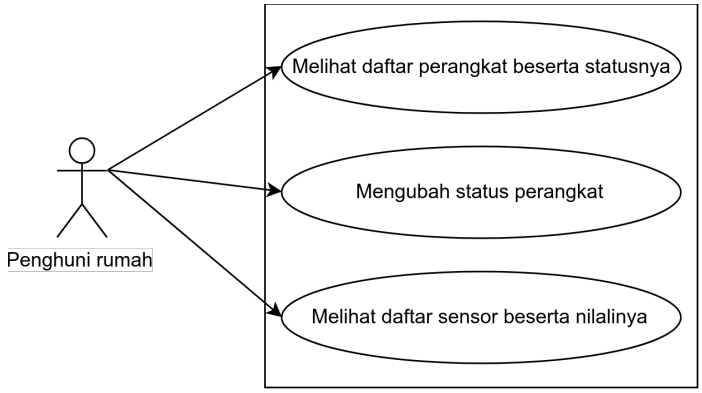


Gambar 3.3 *Activity Diagram* sistem

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 di bawah ini merupakan *use case diagram* dari *web application* yang akan dibuat.



Gambar 3.4 ***Use case diagram web application***

3.7 Pembuatan *Web API* serta *Web Application*

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 Application. 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 Application* 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 Vuejs.

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. Berbeda dengan di *Web API*, penerapan *Websocket* di *Web Application* 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 Application* 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, dimasukkanlah 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 SSE HTTP/1.1, SSE HTTPS, SSE HTTP/2 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 mikrokontroller. 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.

Pemilihan parameter *response time* untuk pembanding dari keempat metode dikarenakan

3.8.2 CPU *Usage*

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 mikrokontroller secara bersamaan, dimana setiap *web browser* diibaratkan sebagai satu pengguna. Setiap perintah yang dikirimkan dari *web browser* akan dibalas oleh mikrokontroller. Perintah yang dikirimkan dari *web browser* memiliki ukuran paket yang sama, begitu juga dengan perintah

yang dikirimkan dari mikrokontroller. Selain menerima balasan dari mikrokontroller, *web browser* juga menerima dua data suhu yang dikirimkan melalui *stream* yang berbeda antar keduanya maupun dengan *stream* yang digunakan untuk menerima pesan dari mikrokontroller. Untuk jumlah *web browser* yang digunakan akan bervariasi selama pengujian.

Dua *Web Browser* yang sama ketika dibuka pada waktu yang bersamaan akan menggunakan *session*, *history*, maupun *cache* yang sama. Hal ini tidak mengganggu pengujian Websocket maupun SSE HTTP/2 yang memiliki karakteristik paralel, namun tidak dengan SSE HTTP/1.1 maupun SSE HTTPS yang hanya mampu membuka enam TCP *Connection*. Untuk mengatasi hal tersebut dibuatlah *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
```

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. https://books.google.co.id/books?id=zWASd07YkVsC&pg=PT543&dq=smart+home&hl=id&sa=X&ved=0ahUKEwifpaWm5qTiAhWJpo8KHTN3Apo4FBDoAQhiMAg#v=onepage&q=smart home&f=false.

Cirani, Simone, Gianluigi Ferrari, Marco Picone, and Luca Veltri. 2018. *Internet of Things : Architectures, Protocols and Standards*. New Jersey: John Wiley & Sons, Inc. https://books.google.co.id/books?id=cg1rDwAAQBAJ&printsec=frontcover&dq=Internet+of+Things: +Architectures, +Protocols+and+Standards&hl=id&sa=X&ved=0ahUKEwjs76mx_pLfAhUP4o8KHWvSBw4Q6AEIKTAA#v=onepage&q&f=false.

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. http://nordsecmob.aalto.fi/en/publications/theses2013/thesis_estep/.

Hillar, Gastón C. 2017. *MQTT Essentials : A Lightweight IoT Protocol : The Preferred IoT Publish-Subscribe Lightweight Messaging Protocol*. Birmingham: Packt. https://books.google.co.id/books?

id=40EwDwAAQBAJ&printsec=frontcover&dq=MQTT+Essentials+-+A+Lightweight+IoT+Protocol&hl=id&sa=X&ved=0ahUKEwiPsKnv-ZLfAhWSe30KHba0DGQQ6AEIKTAA#v=onepage&q=MQTT Essentials - A Lightweight IoT Protocol&f=false.

Ibrahim, Dogan. 2014. "A New Approach for Teaching Microcontroller Courses to Undergraduate Students." *Procedia - Social and Behavioral Sciences* 131 (May): 411–14. https://doi.org/10.1016/J.SBSPRO.2014.04.139.

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. https://doi.org/10.1109/ICIN.2017.7899436.

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. https://doi.org/10.1109/iThings-GreenCom-CPSCom-SmartData.2016.76.

Leach, Paul J., Tim Berners-Lee, Jeffrey C. Mogul, Larry Masinter, Roy T. Fielding, and James Gettys. 1999. "Hypertext Transfer Protocol -- HTTP/1.1." https://tools.ietf.org/html/rfc2616.

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. http://repository.ub.ac.id/3431/.

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. https://doi.org/10.11591/ijeecs.v12.i3.pp1230-1238.

Rhee, Kyung-Hyune, and Jeong Hyun Yi. 2015. *Information Security Applications: 15th International Workshop, WISA 2014*. Berlin: Springer. https://books.google.co.id/books?id=P-pVBgAAQBAJ&pg=PA346&dq=binary+plain+text+protocol&hl=id&sa=X&ved=0ahUKEwja463YgqXiAhVERa0KHSa3CO8Q6AEILDAA#v=onepage&q=binary plain text protocol&f=false.

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. http://j-ptiik.ub.ac.id.

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. https://books.google.co.id/books?id=iru1w4iRx_YC&printsec=frontcover&dq=microcontroller&hl=id&sa=X&ved=0ahUKEwikov6XmKPiAhWJvY8KHSSmAx4Q6AEIKTAA#v=onepage&q=microcontroller&f=false.

Vasseur, Jean-Philippe, Adam Dunkels, Jean-Philippe Vasseur, and Adam Dunkels. 2010. “What Are Smart Objects?” *Interconnecting Smart Objects with IP*, January, 3–20. https://doi.org/10.1016/B978-0-12-375165-2.00001-6.

Wang, Vanessa., Frank. Salim, and Peter. Moskovits. 2013. *The Definitive Guide to HTML5 WebSocket*. Apress. https://books.google.co.id/books?id=YIgdrqKZzYMC&dq=websocket&hl=id&source=gbs_navlinks_s.