Jurnal Politeknik Caltex Riau
Terbit Online pada laman https://jurnal.pcr.ac.id/index.php/jkt/
| e- ISSN : 2460-5255 (Online) | p- ISSN : 2443-4159 (Print) |

Pemanfaatan ReactJS dan Protokol MQTT untuk Visualisasi Sinyal Lampu dan Notifikasi secara Waktu Nyata pada Sistem Pemonitor APILL di Kota Pekanbaru

Ardianto Wibowo[1], M Mahrus Zain[4]

[1] Teknik Informatika, Politeknik Caltex Riau, email: ardie@pcr.ac.id
[2] Sistem Informasi, Politeknik Caltex Riau, email: mahrus@pcr.ac.id

[1] Abstrak

Simpang bersinyal yang ada di kota Pekanbaru mayoritas masih diatur oleh controller yang bersifat analog. Masalah utama yang terjadi dengan kondisi ini adalah terjadinya kemacetan lalu lintas yang lama apabila terjadi kerusakan pada controller APILL.Hal ini diakibatkan informasi yang masuk ke Dinas Perhubungan kota Pekanbaru terkait permasalahan APILL masih mengandalkan laporan dari petugas kepolisian atau masyarakat. Sehingga, Dinas Perhubungan kota Pekanbaru tidak bisa langsung bertindak dengan segera begitu permasalahan terjadi. Terkait dengan permasalahan tersebut, maka telah dikembangkan sebuah sistem terintegrasi yang berfungsi untuk melakukan manajemen dan monitoring kondisi APILL secara real time. Pada penelitian ini difokuskan untuk membahas pemanfaatan ReactJS dan MQTT. ReactJS dimanfaatkan untuk menampilkan data-data sinya lampu APILL serta memberikan notofikasi secara real time ke dalam browser web. Sedangkan MQTT digunakan sebagai media untuk mengelola komunikasi data antara modul deteksi yang dipasang pada controller APILL dilapangan ke server Back-End. Data ini lah yang kemudian ditampilkan ke browser web melalui React JS. Dengan adanya solusi pada penelitian ini, maka diharapkan Dinas Perhubungan kota Pekanbaru dapat mengetahui permasalahan lampu APILL di lapangan secara cepat, sehingga bisa segera mengambil tindakan relevan yang diperlukan.

Kata kunci: *APILL, Simpang Bersinyal, ReactJS, MQTT*

[2] Abstract

The majority of signalized intersections in Pekanbaru city are still regulated by analog controllers. The main problem that occurs with this condition is the occurrence of long traffic jams if there is damage to the traffic light controller. Thus, the Pekanbaru City Transportation Department cannot act immediately once a problem occurs. Related to these problems, an integrated system has been developed that functions to manage and monitor traffic light conditions in real time. This research is focused on discussing the use of ReactJS and MQTT. ReactJS is used to display traffic light light signal data and provide real time notifications into a web browser. Meanwhile, MQTT is used as a medium to manage data communication between the detection module installed on the traffic light controller in the field to the Back-End server. This data is then displayed to the web browser via React JS. With the solution in this research, it is hoped that the Pekanbaru City Transportation Office can find out the problems of traffic light lamps in the field quickly, so that they can immediately take the relevant actions needed.

Keywords: *traffic light, ReactJS, MQTT*

Dokumen diterima pada 25 September, 2021
Dipublikasikan pada 24 Desember, 2021

1. Pendahuluan

Kota Pekanbaru adalah kota yang terletak tepat di tengah pulau Sumatera serta menjadi ibu kota dan kota terbesar di Provinsi Riau, Indonesia. Kota ini merupakan salah satu sentra ekonomi terbesar di Pulau Sumatra, dan termasuk sebagai kota dengan tingkat pertumbuhan, migrasi dan urbanisasi yang tinggi. Saat ini Kota Pekanbaru sedang berkembang pesat menjadi kota dagang yang multi-etnik, keberagaman ini telah menjadi modal sosial dalam mencapai kepentingan bersama untuk dimanfaatkan bagi kesejahteraan masyarakatnya. Saat ini Pekanbaru telah menjadi kota metropolitan, yaitu dengan nama Pekansikawan, (Pekanbaru, Siak, Kampar, dan Pelalawan). Pekanbaru dihubungkan oleh jaringan jalan yang tersambung dari arah Padang di sebelah barat, Medan di sebelah utara, dan Jambi di sebelah selatan. Terminal Bandar Raya Payung Sekaki merupakan pusat pelayanan transportasi antar kota dan antar provinsi, yang telah direncanakan pemerintah setempat menjadi sarana orientasi dan perpindahan antar moda transportasi dengan akses ke sistem jaringan transportasi regional, bandara, dan pelabuhan.

Semua kegiatan masyarakat di kota Pekanbaru tersebut memerlukan infrastruktur lalu lintas dan transportasi yang memadai. Secara tradisional, Pekanbaru telah membangun jaringan jalan-jalan di dalam kota, baik jalan primer ataupun jalan sekunder. Jaringan jalan tersebut terus dikembangkan atau diperlebar sesuai kebutuhan. Akan tetapi, langkah tersebut tersebut belum cukup untuk memenuhi kebutuhan masyarakat dalam berlalu lintas. Selain keterbatasan lahan jalan untuk menampung pertumbuhan pengguna jalan yang semakin meningkat dari waktu ke waktu, lalu lintas di Pekanbaru juga belum didukung sepenuhnya oleh penerapan Teknologi Informasi secara menyeluruh. Hal ini ditandai dengan semakin macet dan semrawut-nya lalu lintas di dalam kota Pekanbaru pada waktu-waktu tertentu setiap hari, terutama di area simpang bersinyal.

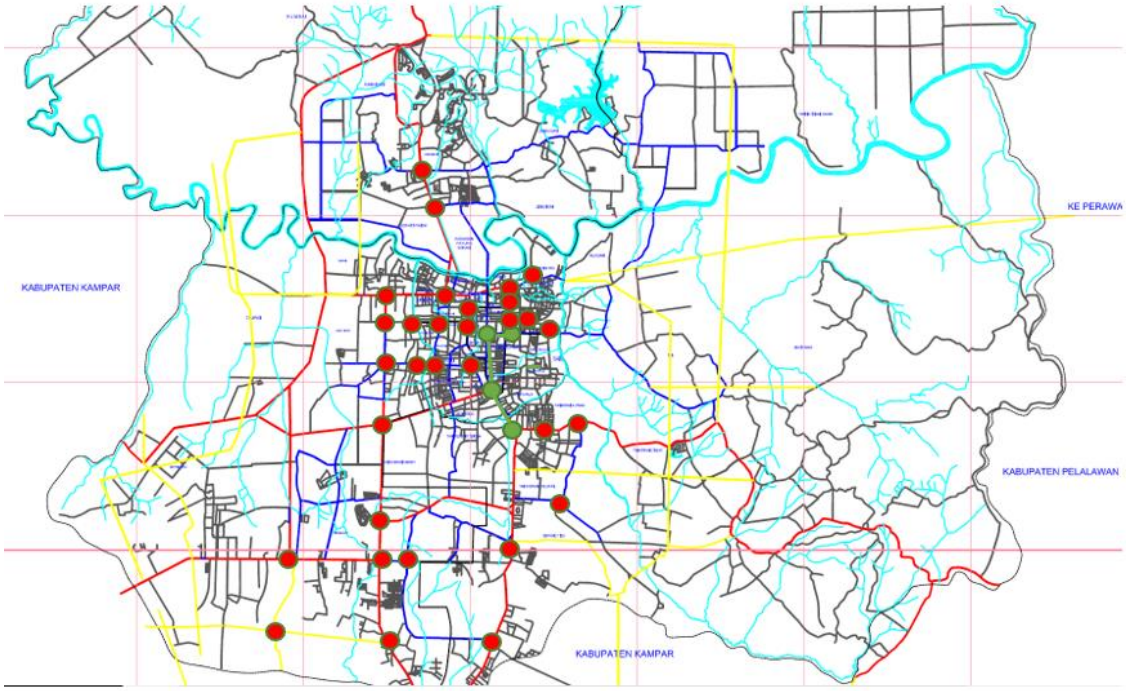
Per tahun 2020 ini, terdapat 35 simpang bersinyal yang telah dilayani oleh APILL. Dari 35 APILL yang ada, terdapat 4 simpang yang sudah dilewati kabel Fiber Optik serta menerapkan ATCS (Area Traffic Control System) berbasis sensor, yaitu:

1. Simpang 3 Sudirman – Harapan Raya
2. Simpang 3 Sudirman – Nangka
3. Simpang 3 Sudirman – Gajah mada
4. Simpang 3 Gajah Mada – Diponegoro

Sedangkan, 31 APILL lainnya masih menggunakan kontroler analog. Gambar sebaran APILL pada simpang bersinyal ditunjukkan pada Gambar 1 berikut, dimana titik warna merah merupakan APILL berbasis ATCS, sedangkan titik warna hijau merupakan APILL analog. Pada APILL berbasis ATCS, durasi lama waktu lampu disesuaikan dengan panjang kendaraan yang dibaca oleh beberapa sensor yang berada di sekitar simpang.

Dengan kondisi seperti ditunjukkan pada Gambar 1, kemacetan di jam-jam tertentu masih sering ditemukan pada beberapa simpang bersinyal, seperti:

1. Simpang 4 Rumbai
2. Simpang 4 Delima
3. Simpang 3 Garuda Sakti
4. Simpang 3 Tabek Gadang
5. dan beberapa simpang yang lain.



Gambar 1. Peta sebaran APILL kota Pekanbaru

Kemacetan diperparah lagi apabila terjadi masalah pada APILL. Hal ini diakibatkan informasi yang masuk ke Dinas Perhubungan kota Pekanbaru terkait permasalahan APILL masih mengandalkan laporan dari petugas kepolisian atau masyarakat. Sehingga, Dinas Perhubungan kota Pekanbaru tidak bisa langsung bertindak dengan segera begitu permasalahan terjadi.

Terkait dengan permasalahan tersebut, maka telah dikembangkan sebuah sistem terintegrasi yang berfungsi untuk melakukan manajemen dan monitoring kondisi APILL secara real time. Pada penelitian ini difokuskan untuk membahas pemanfaatan ReactJS dan MQTT. ReactJS dimanfaatkan untuk menampilkan data-data sinya lampu APILL serta memberikan notofikasi secara real time ke dalam browser web. Sedangkan MQTT digunakan sebagai media untuk mengelola komunikasi data antara modul deteksi yang dipasang pada controller APILL dilapangan ke server Back-End. Data ini lah yang kemudian ditampilkan ke browser web melalui React JS. Dengan adanya solusi pada penelitian ini, maka diharapkan Dinas Perhubungan kota Pekanbaru dapat mengetahui permasalahan lampu APILL di lapangan secara cepat, sehingga bisa segera mengambil tindakan relevan yang diperlukan.

2. Tinjauan Pustaka

2.1 ReactJS

React, sering ditulis juga React.js atau ReactJS merupakan JavaScript *library* yang dikembangkan oleh Facebook untuk memfasilitasi pembuatan daripada komponen antarmuka yang interaktif, *statefull*, serta mudah untuk digunakan ulang. ReactJS sangatlah cocok digunakan untuk *rendering* antarmuka yang kompleks dengan performa tinggi [1]. JavaScript *library* ini sendiri telah digunakan oleh Facebook untuk bagian *newsfeed* mereka. Selain itu, banyak situs-situs terkenal juga yang menggunakan ReactJS ini sebagai salah satu alat yang digunakan dalam mengembangkan situs tersebut, seperti Netflix, Paypal, Vevo, dan masih banyak lagi [1]. Hal ini membuktikan bahwa ReactJS merupakan salah satu *library* JavaScript yang sangat berkembang, banyak digunakan, serta sangat handal dalam melaksanakan tugasnya sebagai UI-*rendering* JavaScript *library*.

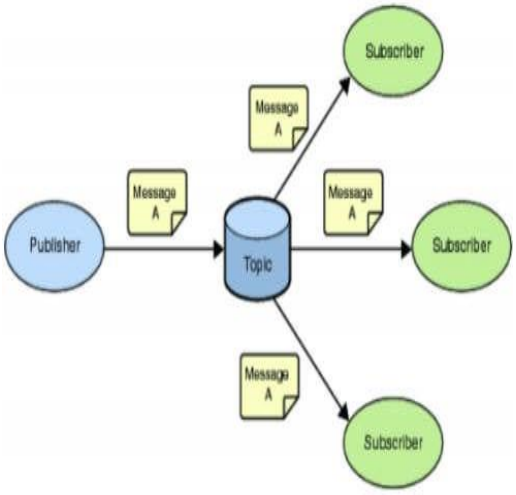
React merupakan bagian *view* dari konsep MVC *(model-view-controller)* [1], yang berarti React hanya mengurusi bagian tampilan antarmuka dengan pengguna saja, tanpa mengurusi bagian cara mendapatkan data ataupun hubungan ke basis data. Dalam pengembangannya, banyak sekali hal yang telah dapat dijalankan oleh React seperti menganimasikan suatu obyek dengan efek transisi, menjalankan permainan di *web browser* yang sepenuhnya diprogram dengan menggunakan React, ataupun validasi form yang berjalan secara *real-time* sembari pengguna mengisikan data pada form tersebut. Pemakaian ReactJS dalam sebuah situs dapat dilihat dengan menggunakan alat tambahan pada Google Chrome yang bernama React Developer Tools yang dapat mendeteksi keberadaan atau penggunaan ReactJS dalam suatu *web* [2] [3] [4]

2.2 MQTT

Protokol MQTT (Message Queue Telemetry Transport) adalah protokol pesan ringan (lightweight) berbasis publish-subscribe digunakan di atas protokol TCP/IP. Protokol ini mempunyai ukuran paket data *low overhead* kecil (minimal 2 gigabyte) dengan konsumsi catu daya kecil. MQTT bersifat terbuka, simpel dan didesain agar mudah untuk diimplementasikan, yang mampu menangani ribuan *client* jarak jauh dengan hanya satu server. Karakteristik ini membuatnya ideal untuk digunakan dalam banyak situasi, termasuk lingkungan terbatas seperti dalam komunikasi *Machine to Machine* (M2M) dan konteks *Internetof Things* (IOT) dimana dibutuhkan kode footprint yang kecil dan/atau jaringan yang terbatas. Pola pesan publish-subscribe membutuhkan broker pesan. Broker bertanggung jawab untuk mendistribusikan pesan ke klien tertarik berdasarkan topik pesan [5]

Berikut merupakan fitur protokol MQTT [6]:

1. Publish/subscribe message pattern yang menyediakan distribusi message dari satu ke banyak dan decoupling aplikasi.
2. Messagging transport yang agnostic dengan isi dari payload.
3. Menggunakan TCP/IP sebagai konektivitas dasar jaringan.
4. Terdapat tiga level Qualities of Service (Qos) dalam penyampaian pesan :
   a. “At most once”, di mana pesan dikirim dengan upaya terbaik dari jaringan TCP/IP. Kehilangan pesan satau terjadi duplikasi dapat terjadi.
   b. “At least once”, dapat dipastikan pesan tersampaikan walaupun duplikasi dapat terjadi.
   c. “Exactly once”, dimana pesan dapat dipastikan tiba tepat satu kali



Gambar 2. Arsitektur MQTT

Berdasarkan berbagai keunggulan khusus MQTT tersebut, banyak peneliti berusaha mengimplementasikan protokol tersebut di berbagai bidang, seperti implementasi MQTT dan smartphone sebagai alat untuk peringatan dini gempa bumi [7]. MQTT digunakan karena lebih efisien dalam mengkonsumsi bandwith, serta menggunakan daya baterai yang tidak besar, sehingga sangat cocok dalam situasi darurat seperti gempa bumi.

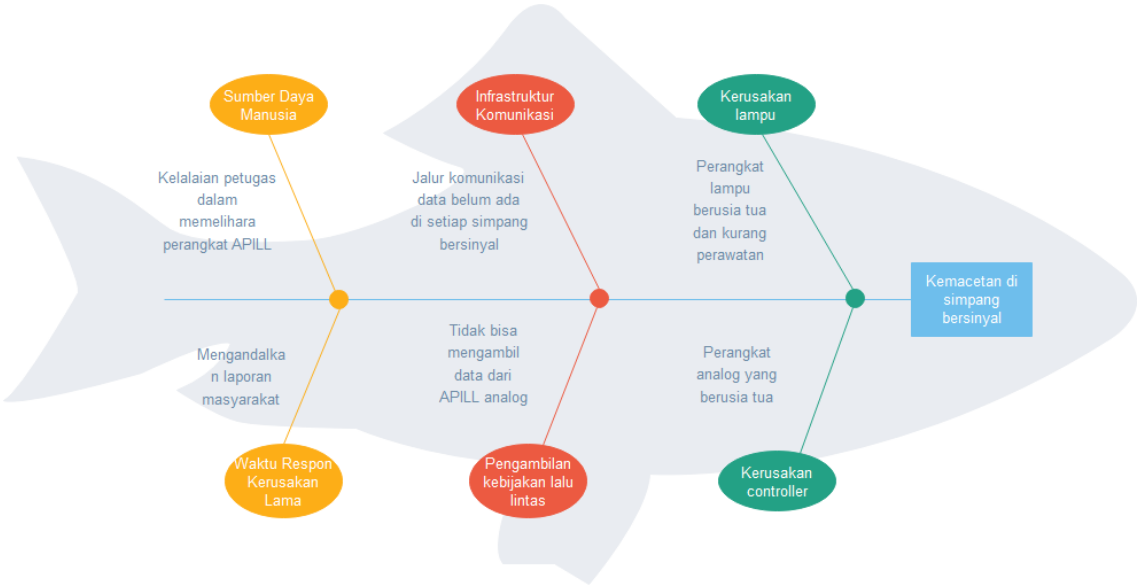
MQTT juga diterapkan untuk memonitor detak jantung pasien secara jarak jauh [8], dimana mengkombinasikan teknologi wifi modul ESP8266 pada Arduino microcontroller dan protokol MQTT untuk mentransmisikan data. MQTT dipilih sebagai protokol karena sangat ringan digunakan untuk berkomunikasi.

Sistem yang dikembangkan memiliki fungsi untuk memonitor detak jantung pasien secara real time. Sistem ini dirancang menggunakan infrared dan photo detector untuk mendeteksi dan mengambil datasinyal detak jantung, lalu mengirimkan datanya kepada broker MQTT yang berjalan pada Raspberry pi, sebuah komputer dengan harga murah seukuran kartu kredit. Hasilnya, penerapan teknologi ini sangat membantu petugas kesehatan untuk memonitor detak jantung pasien.

Beberapa tahun belakangan ini, implementasi Internet of Things (IoT) pada peralatan rumah tangga semakin marak. Tren ini juga didukung oleh murahnya peralatan smartphone yang digunakan sebagai alat input dan kontroler bagi IoT.MQTT sebagai protokol komunikasi dengan bandwith rendah turut mendukung implementasi IoT, seperti otomasi peralatan rumah tangga menggunakan teknologi Global System for Mobile Communication (GSM) [9] [10] [11].

3. Metode Penelitian

Tahap pertama dari metode penelitian yang dikembangkan adalah menjabarkan potensi masalah yang muncul. Potensi masalah yang muncul sehingga mengakibatkan macet pada simpang bersinyal. Detail potensi masalah ditunjukkan melalui *fishbone diagram* pada Gambar 3 sebagai berikut:



Gambar 3. Fishbone diagram masalah kemacetan di simpang bersinyal.

Seperti ditunjukkan pada Gambar 3, terdapat 6 potensi permasalahan yang mengakibatkan kemacetan pada simpang bersinyal, yaitu: a). Sumber Daya Manusia, b). Infrastruktur Komunikasi, c). Kerusakan lampu APILL, d). Kerusakan controller APILL, e). Waktu respon kerusakan yang lama, serta e). Pengambilan kebijakan lalu lintas yang kurang tepat.

Penelitian ini berfokus pada monitoring kerusakan controller. Untuk merealisasikan penelitian ini, dikembangkan metode penelitian sebagai berikut:

1. *Membangun server MQTT*

   MQTT merupakan protokol yang digunakan untuk mentransmisikan data melalui suatu server perantara yang disebut middleware atau server broker. Untuk itu, langkah pertama sebelum menggunakan MQTT adalah membangun servernya terlebih dahulu. Server yang dibangun berjalan di atas Sistem Operasi Linux.

2. *Mendesain skema topic pada MQTT*

   Setelah server MQTT bisa diakses oleh client, maka langkah selanjutnya adalah mendesain skema topic yang akan dijadikan lintasan komunikasi antara modul deteksi controller APILL di lapangan dengan server untuk menampilkan visualisasi.

3. *Uji coba topic menggunakan mekanisme publish / subscribe*

   Perangkat yang dikembangkan pada tahap (2) memerlukan pengujian menggunakan client tools. Client tools yang digunakan adalah MQTT Explorer.

4. *Membangun server front-end berbasis NodeJS dan ReactJS*

   Server front-end merupakan *endpoint* dari data yang dikirimkan melalui MQTT server. Fungsi utama bagian ini adalah menampilkan data-data yang dikirimkan oleh MQTT ke dalam bentuk visualisasi yang relevan. Server front-end dibangun menggunakan ReactJS menggunakan environment NodeJS.

5. *Mendesain layout visualisasi lampu APILL pada ReactJS*

   Setelah ReactJS berjalan lancar, langkah selanjutnya adalah mengembangkan visualisasi pola lampu APILL menggunakan komponen yang relevan.

6. *Menghubungkan komunikasi data MQTT dan ReactJS*

   Setelah komponen visualisasi APILL selesai dikembangkan, langkah selanjutnya adalah menghubungkan dengan data sinyal lampu APILL yang dikirimkan dari MQTT. Lampu pada komponen visualisasi ReactJS akan bergerak sesuai dengan pola sinyal lampu yang dikirimkan oleh MQTT.

7. *Pengujian dan kalibrasi di lingkungan laboratorium*

   Selanjunya, pengujian di laboratorium dilakukan untuk mengevaluasi apakah sinyal lampu yang dikirimkan akurat sesuai dengan sinyal lampu yang dikirimkan oleh MQTT.

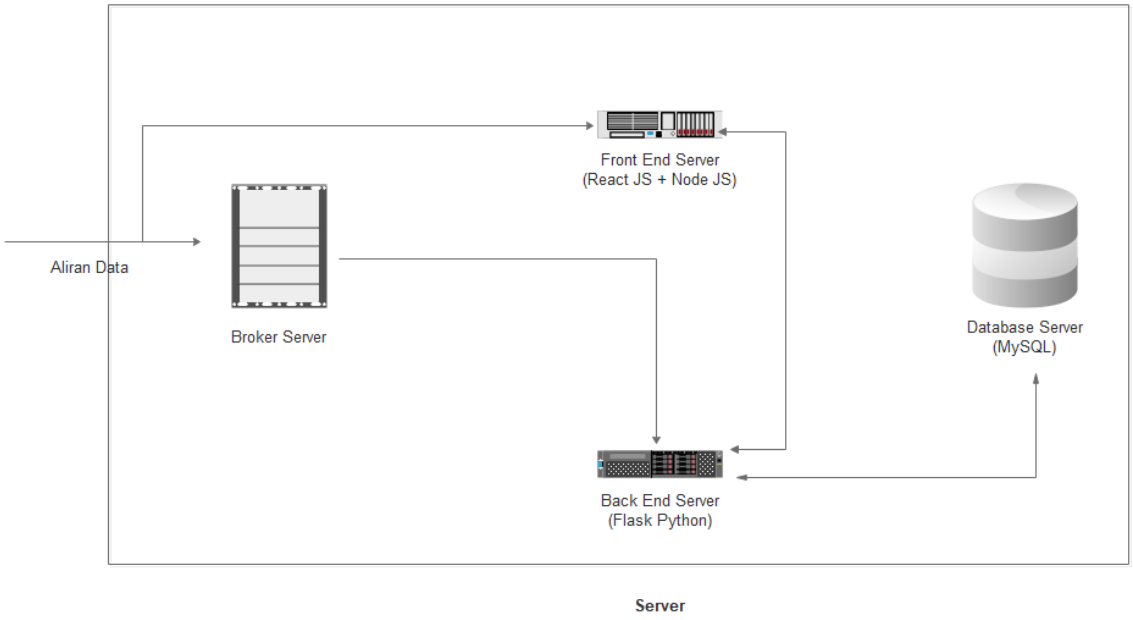
8. *Pengujian dan kalibrasi di lingkungan operasional*

   Langkah terakhir adalah melakukan pegujian di lingkungan operasional. Makna lingkungan operasional adalah sistem ditempatkan pada server yang bisa dilihat oleh public, sedangkan data yang dikirimkan oleh MQTT ke ReactJS benar-benar berasal dari data tangkapan modul deteksi di lapangan.

4. Hasil dan Pembahasan

4.1 Arsitektur sistem

Sistem yang dikembangkan menggunakan 4 node berbeda, yaitu Broker Server (MQTT Server), Backend Server, Front-End Server, dan Database Server. Arsitektur sistem yang dikembangkan ditunjukkan pada Gambar 4 berikut:

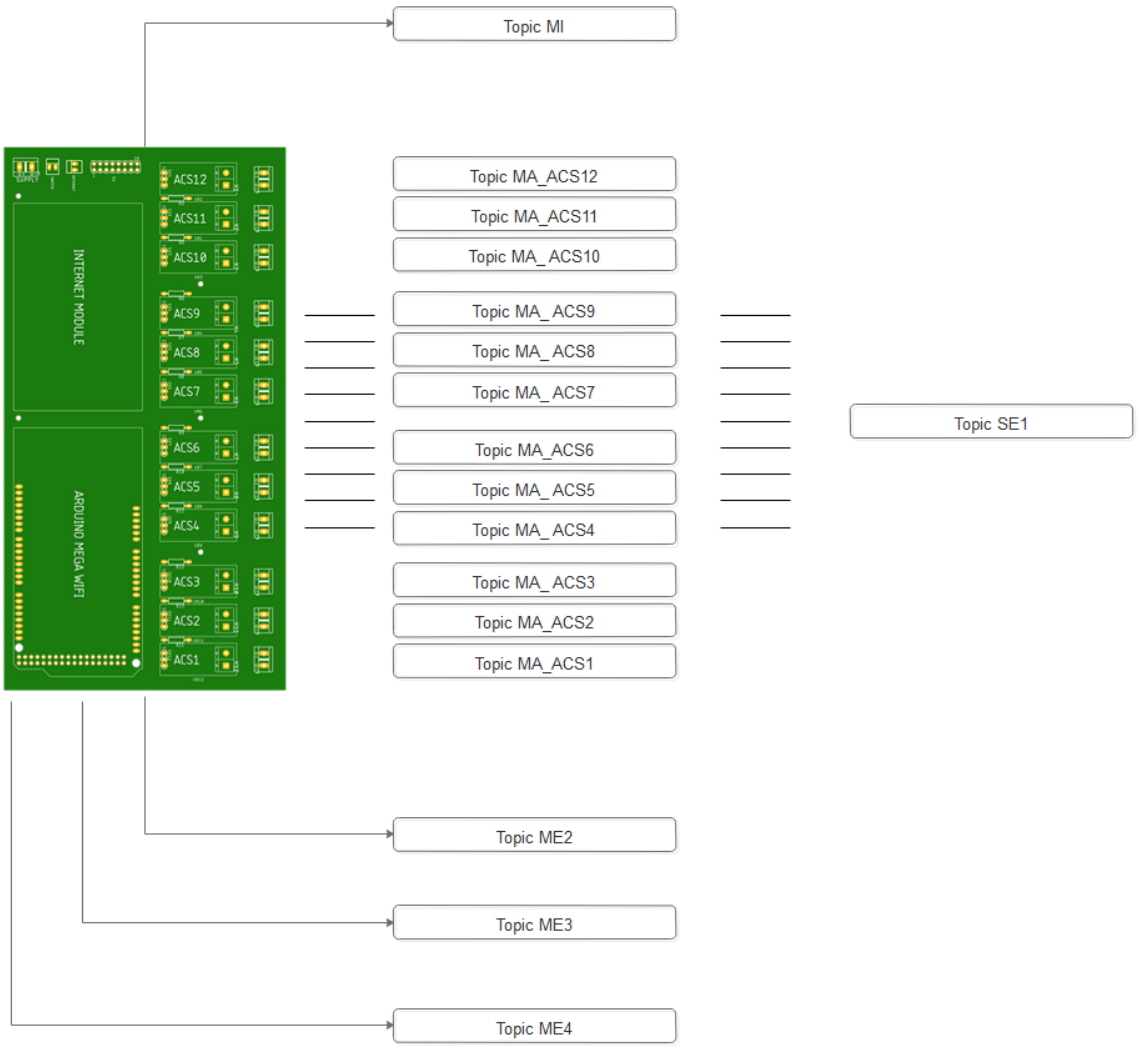


Gambar 4. Arsitektur sistem yang dikembangkan.

Berdasarkan gambar tersebut, setiap aliran data dari modul deteksi diterima oleh 2 komponen server yang berbeda, yaitu server broker dan server front-end. Selanjutnya, server broker akan mengirimkan data atau notifikasi ke komponen server lainnya, yaitu Back End server. Front End server digunakan untuk memvisualisasikan data notifikasi real time, sedangkan Back End server digunakan untuk menyimpan data ke dalam database. Fokus pada penelitian ini adalah pada komunikasi antara Broker Server (MQTT server) dengan Front-End server (ReactJS + NodeJS).

4.2 Desain dan implementasi skema topic pada MQTT

Di dalam server MQTT, mekanisme komunikasi data menggunakan pola publish – subscribe. Mekanisme publish – subscribe dikelola menggunakan serangkaian topic (channel) komunikasi. Berikut ini merupakan desain topic yang dikembangkan pada MQTT server:



Gambar 5. Desain skema topic pada MQTT

Penjelasan detail dari setiap topic yang dikembangkan dapat dilihat pada Tabel 1 berikut:

Tabel 1. Penjelasan Topic yang dikembangkan pada MQTT

Nama Topic	Pengirim	Keterangan	Format JSON

MI	Modul Deteksi	Inisiasi koneksi. Trigger-nya adalah saat Mikro baru dinyalakan dan terhubung ke server.
		
		Status
		1: Connected
			{
			"id_simpang": "xx",
			"nama_simpang": "xx",
			"status": 1
			}

ME2	Modul Deteksi	Mendeteksi error tipe 2. Trigger deteksinya adalah arus untuk 12 lampu terdeteksi 0 A dalam waktu bersamaan.
		
		Status
		0: Error tipe 2 terdeteksi
		1: Error tipe 2 resolved
			{
			"id_simpang": "xx",
			"nama_simpang": "xx",
			"status": 0
			}

Nama Topic	Pengirim	Keterangan	Format JSON

ME3	Modul Deteksi	Mendeteksi error tipe 3. Trigger deteksinya adalah pola 12 lampu tidak sesuai dengan state normal yang tertera pada tabel.
		
		Status
		0: Error tipe 3 terdeteksi
		1: Error tipe 3 resolved
			{
			"simpang_id": "S01",
			"simpang_nama": "Simpang SKA",
			"status": 1,
			"status_klik": 0,
			"nilai_arus": 1
			}

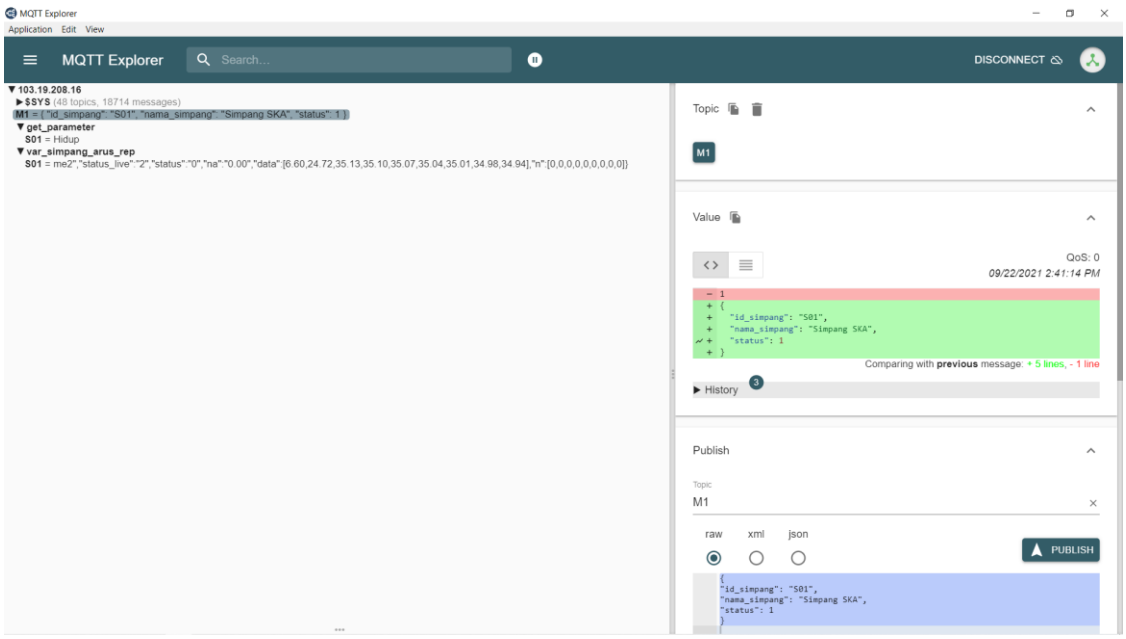
ME4	Modul Deteksi	Mendeteksi error tipe 4. Trigger deteksinya terdapat >1 lampu pada sebuah simpang yang mempunyai nilai di luar range normal dalam waktu bersamaan.
		
		Status
		0: Error tipe 4 terdeteksi
		1: Error tipe 4 resolved
			{
			"id_simpang": "**xx**";
			"id_kaki_simpang": "**xx**";
			"nama_simpang": "**xx**";
			"nama_kaki_simpang": "**xx**";
			"nilai_arus": "**xx**";
			"status": **[0|1]**
			}

MA_{xxxx}	Modul Deteksi	(t) Periodik per state APILL
		
		Status_arus
		0: lampu mati atau menyala di luar range normal
		1: lampu nyala di dalam range normal
			{
			"id_simpang": "xx",
			"id_lampu": "yy",
			"nama_simpang": "xx",
			"nama_lampu": "xx",
			"nilai_arus": "xx",
			"status_arus": 0
			}

SE1	Server (Backend)	Mendeteksi error tipe 1. Trigger deteksinya adalah count baris data pada sebuah simpang mengeluarkan nilai yang sama dalam n loop.
		
		Status
		0: Error tipe 1 terdeteksi
		1: Error tipe 1 resolved
			{
			"simpang_id": "S01",
			"simpang_nama": "Simpang SKA",
			"status": 0,
			"status_klik": 0
			}

4.3 Pengujian komunikasi menggunakan MQTT Explorer

MQTT explorer digunakan untuk menguji fungsionalitas dari server MQTT. Fungsi dianggap berjalan apabila format pesan yang telah ditentukan berhasil diterima (published) oleh MQTT server. Gambar 6 berikut ini merupakan tampilan salah satu format data JSON yang dikirimkan melalui MQTT explorer:



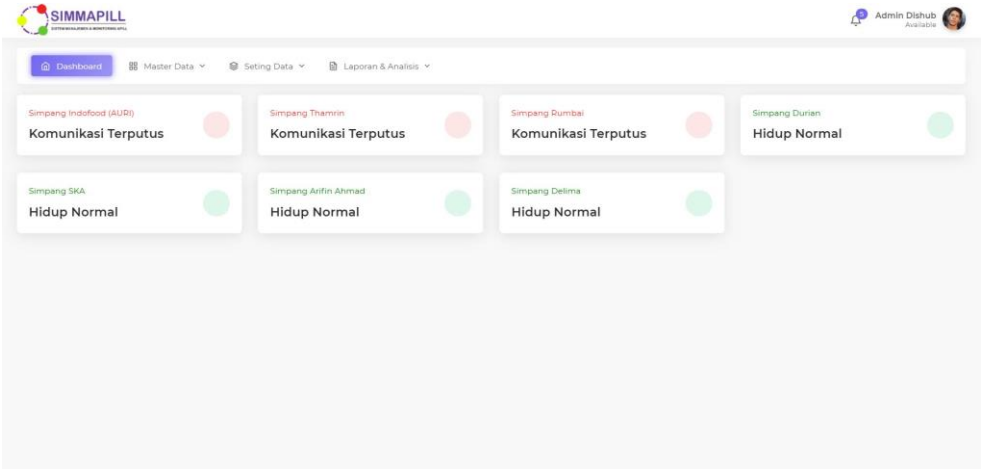
Gambar 6. Pemanfaatan MQTT Explorer untuk menguji konektivitas ke MQTT server

4.4 Tampilan visualisasi APILL pada ReactJS

Terdapat 2 tampilan utama untuk proses monitoring sinyal lampu APILL di server front-end yang dikembangkan dengan ReactJS, yaitu tampilan notifikasi status sinyal per simpang, serta status sinyal setiap lampu pada APILL secara waktu nyata. Pada notifikasi status sinyal per simpang, terdapat 3 kondisi yang dimonitor, yaitu:

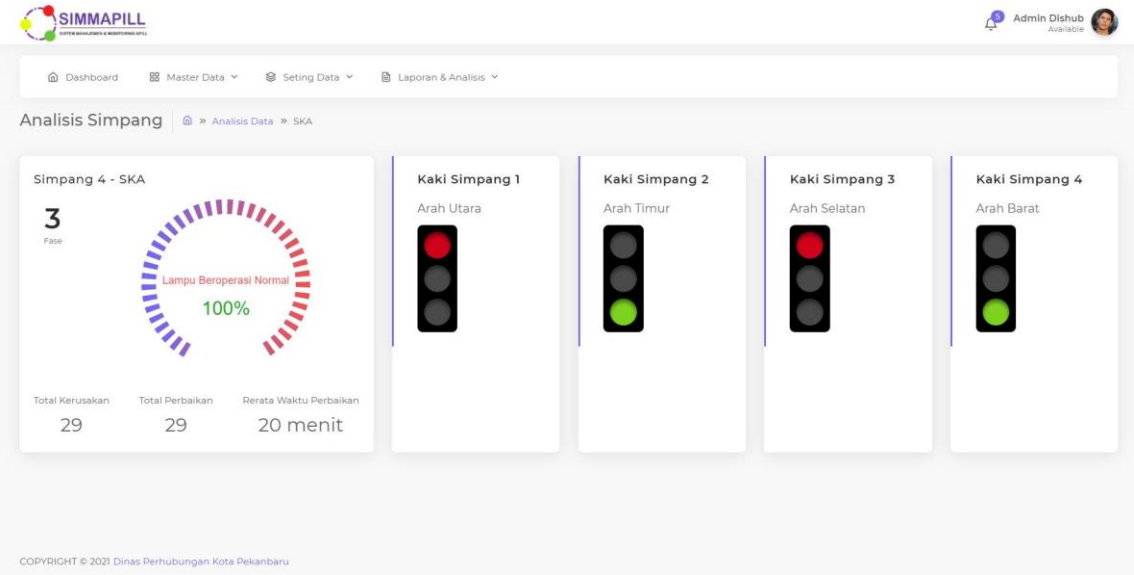
a. Normal, berarti tidak terjadi permasalahan pada simpang terkait.
b. Komunikasi terputus, berarti controller APILL tidak hidup, atau terdapat permasalahan koneksi internet dari modul deteksi lapangan ke server
c. Logika controller bermasalah, berarti controller APILL hidup tetapi terjadi permasalahan pada sinyal lampu yang dikeluarkan.

Tampilan visualisasi untuk notifikasi status sinyal per simpang secara waktu nyata ditunjukkan pada Gambar 7 berikut:



Gambar 7. Visualisasi notifikasi status per simpang secara waktu nyata

Status sinyal setiap lampu pada APILL secara waktu nyata menggambarkan kondisi lampu yang sedang berjalan pada setiap simpang yang diamati. Pergerakan lampu yang ada di visiualisasi menggambarkan pergerakan sinyal lampu yang sedang terjadi di lapangan. Visualisasi terkait sinyal lampu ditunjukkan pada Gambar 7:



Gambar 7. Visualisasi lampu APILL secara waktu nyata

4.6 Pengujian dan Analisis

4.6.1. Pengujian akurasi sinyal data pada visualisasi

Arah dari pengujian dan analisis yang dilakukan adalah menguji tingkat akurasi visualisasi yang ditampilkan terhadap sinyal data yang dikirimkan dari modul deteksi sinyal lampu APILL di lapangan. Target ideal yang diharapkan adalah bahwa visualisasi pada tampilan web di ReactJS sesuai sepenuhnya dengan sinyal yang dikirimkan.

Langkah pengujian yang dilakukan adalah sebagai berikut:

1. Mencatat sinyal yang dikirimkan oleh modul deteksi di lapangan mengirimkan sinyal secara kontinyu setiap 3 detik ke dalam database server pada backend. Sinyal yang dikirimkan adalah per lampu, dimana nilai-nya adalah 0 dan 1. Nilai 0 dan 1 tersebut dikonversi oleh modul deteksi berdasarkan batas bawah yang diberlakukan secara tunggal untuk setiap simpang. Gambar 8 merupakan salah satu tampilan data yang didapatkan beserta nilai konversi yang tercatat ke dalam database server pada backend dalam kurun waktu kurang lebih 1 menit, sedangkan Gambar 9 merupakan nilai konversi sinyalnya.

	MA1	MA2	MA3	MA4	MA5	MA6	MA7	MA8	MA9	MA10	MA11	MA12
1	0.28937879	0.23269697	0.35895454	0.39525	0.23045454	0.22836364	0.29206818	0.20740152	0.22436364	0.2959697	0.19163636	0.22135606
2	0.23291666	0.24604166	0.25666666	0.26999999	0.22	0.18000001	0.17	0.17	0.15000001	0.16	0.18000001	0.18000001
3	1.2424992	0.3046069	0.75831895	1.13020933	0.29897731	0.84401087	2.20818632	0.28115372	0.67048098	0.61670981	0.26588367	0.63290828
4	1.26254173	0.30688849	0.88208547	1.06952148	0.30106165	1.06200312	2.12776987	0.28547073	0.76482528	0.57709548	0.27209437	0.82613621
5	4.37526747	4.50444938	4.45803398	4.50915589	4.4446782	4.50109819	4.47365271	4.58473177	5.0630665	5.17199893	5.21791842	5.33621642
6	0.39518988	0.40637131	0.39886076	0.40033755	0.40021097	0.39379747	0.40582279	0.4007173	0.39746836	0.40181434	0.39358649	0.40151899
7	0.23937879	0.18269697	0.30895454	0.34525	0.18045454	0.17836364	0.24206818	0.15740152	0.17436364	0.2459697	0.14163636	0.17135606
8	0.18291666	0.19604166	0.20666666	0.21999999	0.17	0.13000001	0.12	0.12	0.10000001	0.11	0.13000001	0.13000001
9	1.1924992	0.2546069	0.70831895	1.08020933	0.24897731	0.79401087	2.15818632	0.23115372	0.62048098	0.56670981	0.21588367	0.58290828
10	1.21254173	0.25688849	0.83208547	1.01952148	0.25106165	1.01200312	2.07776987	0.23547073	0.71482528	0.52709548	0.22209437	0.77613621
11	4.32526747	4.45444938	4.40803398	4.45915589	4.3946782	4.45109819	4.42365271	4.53473177	5.0130665	5.12199893	5.16791842	5.28621642
12	0.34518988	0.35637131	0.34886076	0.35033755	0.35021097	0.34379747	0.35582279	0.3507173	0.34746836	0.35181434	0.34358649	0.35151899
13	0.18937879	0.13269697	0.25895454	0.29525	0.13045454	0.12836364	0.19206818	0.10740152	0.12436364	0.1959697	0.09163636	0.12135606
14	0.13291666	0.14604166	0.15666666	0.16999999	0.12	0.08000001	0.07	0.07	0.05000001	0.06	0.08000001	0.08000001
15	1.1424992	0.2046069	0.65831895	1.03020933	0.19897731	0.74401087	2.10818632	0.18115372	0.57048098	0.51670981	0.16588367	0.53290828
16	1.16254173	0.20688849	0.78208547	0.96952148	0.20106165	0.96200312	2.02776987	0.18547073	0.66482528	0.47709548	0.17209437	0.72613621
17	4.27526747	1.69783922	1.72194976	1.75158115	1.67511457	1.69108643	1.70718123	1.71428353	1.8782995	1.93992766	1.91771376	1.96969716
18	0.27249511	0.26170331	0.30482732	0.32186251	0.25022184	0.2340537	0.25596366	0.2260396	0.223944	0.25259468	0.22174096	0.23429168
19	0.53826488	0.21111518	0.40798005	0.54848644	0.19981062	0.36745817	0.84008484	0.16951841	0.29828154	0.30755984	0.16250668	0.29475478
20	0.86265253	0.23584569	0.58235703	0.7732436	0.22334632	0.645338	1.4519854	0.19554149	0.47843542	0.40126843	0.18932602	0.49634817
21	2.24343613	1.65531493	1.9828128	2.18629557	1.63157239	2.08570406	2.8865363	1.66711874	2.11612426	2.07193474	1.86863215	2.21508697
22	1.96099969	1.68923639	1.8629934	1.94300497	1.66531694	1.93563292	2.28574846	1.70697327	2.02512005	2.00030292	1.91119976	2.13795721
23	1.61994538	1.64783922	1.67194976	1.70158115	1.62511457	1.64108643	1.65718123	1.66428353	1.8282995	1.88992766	1.86771376	1.91969716
24	0.22249511	0.21170331	0.25482732	0.27186251	0.20022184	0.1840537	0.20596366	0.1760396	0.173944	0.20259468	0.17174096	0.18429168

Gambar 8. Data arus yang dikirimkan dalam durasi sekitar 1 menit

Menggunakan nilai ambang arus 2.5 Ampere (di bawah 2.5 Ampere dianggap lampu mati, di atas 2.5 Amper dianggap lampu hidup), didapatkan tabel status sebagai berikut:

x: 0.25

	MA1	MA2	MA3	MA4	MA5	MA6	MA7	MA8	MA9	MA10	MA11	MA12
1	1	0	1	1	0	0	1	0	0	1	0	0
2	0	0	1	1	0	0	0	0	0	0	0	0
3	1	1	1	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1	1	1	1
5	1	1	1	1	1	1	1	1	1	1	1	1
6	1	1	1	1	1	1	1	1	1	1	1	1
7	0	0	1	1	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0
9	1	1	1	1	0	1	1	0	1	1	0	1
10	1	1	1	1	1	1	1	0	1	1	0	1
11	1	1	1	1	1	1	1	1	1	1	1	1
12	1	1	1	1	1	1	1	1	1	1	1	1
13	0	0	1	1	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0
15	1	0	1	1	0	1	1	0	1	1	0	1
16	1	0	1	1	0	1	1	0	1	1	0	1
17	1	1	1	1	1	1	1	1	1	1	1	1
18	1	1	1	1	1	0	1	0	0	1	0	0
19	1	0	1	1	0	1	1	0	1	1	0	1
20	1	0	1	1	0	1	1	0	1	1	0	1
21	1	1	1	1	1	1	1	1	1	1	1	1
22	1	1	1	1	1	1	1	1	1	1	1	1
23	1	1	1	1	1	1	1	1	1	1	1	1
24	0	0	1	1	0	0	0	0	0	0	0	0

Gambar 9. Konversi data arus ke dalam bentuk sinyal 0 dan 1

2. Selain mengirimkan data sinyal lampu ke database server pada backend, modul deteksi secara bersamaan juga mempublish ke MQTT server melalui topic yang telah didesain. Data yang dipublish ke MQTT hanyalah data konversi. Topic tersebut di-subscribe oleh ReactJS untuk keperluan menangkap sinyal dan visualisasi data. Dalam periode waktu yang sama dengan langkah pengujian (1), didapatkan kondisi data seperti ditunjukkan oleh Gambar 10.

	MA1	MA2	MA3	MA4	MA5	MA6	MA7	MA8	MA9	MA10	MA11	MA12
1	1	0	1	1	0	0	1	0	0	1	0	0
2	0	0	1	1	0	0	0	0	0	1	0	0
3	0	0	1	1	0	0	0	0	0	1	0	0
4	0	0	1	1	0	0	0	0	0	0	0	0
5	1	1	1	1	1	1	1	1	1	1	1	1
6	1	1	1	1	1	1	1	1	1	1	1	1
7	1	1	1	1	1	1	1	1	1	1	1	1
8	1	1	1	1	1	1	1	1	1	1	1	1
9	0	0	1	1	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0
11	1	1	1	1	0	1	1	0	1	1	0	1
12	1	1	1	1	1	1	1	1	1	1	1	1
13	0	0	1	1	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0
15	1	0	1	1	0	1	1	0	1	1	0	1
16	1	0	1	1	0	1	1	0	1	1	0	1

Gambar 10. Konversi data arus ke dalam bentuk sinyal 0 dan 1

3. Membandingkan hasil perekaman data pada langkah (1) dan langkah (2). Ditemukan kondisi sebagai berikut:

   Jumlah baris yang beririsan (sama) *: 13*
   Jumlah baris basis (yang lebih banyak, terekam ke database) *: 24*
   *Tingkat akurasi: (11/24) * 100% =* ***: 54%***

4. Mengulang langkah (1) sampai dengan (3) untuk jarak waktu yang lain. Pada pengujian ini dilakukan sebanyak 5 *range* satuan waktu. Ditemukan kondisi sebagai berikut:

	Range pengujian	Akurasi
1	1	0.54
2	2	0.63
3	3	0.55
4	4	0.67
5	5	0.63

Rerata **0.604**

4.6.2. Analisis

Berdasarkan kondisi pengujian pada (4.6.1), persentasi akurasi sinyal yang ditampilkan oleh front-end ReactJS berada di kisaran 60%. Kondisi perbedaan yang ditemukan pada kelompok sinyal yang diterima adalah sebagai berikut:

a. Pada range waktu yang sama, jumlah kelompok sinyal yang diterima oleh server back-end (langsung dari modul deteksi) tidak sama dengan jumlah kelompok sinyal yang diterima oleh server front-end (melalui MQTT server).
b. Terdapat kelompok sinyal yang diterima back-end, tetapi tidak muncul di front-end. Demikian juga sebaliknya, tidak semua kelompok sinyal pada front-end muncul pada back-end.

Sesuai dengan kondisi (a) dan (b), penyebab yang muncul adalah digunakannya 2 mekanisme komunikasi yang berbeda antara modul controller ke back-end server (menggunakan mekanisme request-reply) serta modul controller ke MQTT server dan front-end server (menggunakan mekanisme publish – subscribe). Pada mekanisme publish – subscribe, kemungkinan munculnya *data loss* tanpa diketahui dan tanpa bisa dikontrol sangat tinggi.

5. Kesimpulan

ReactJS dan MQTT telah dapat dipergunakan atau dikombinasikan untuk menampilkan notifikasi dan data sinyal lampu APILL secara waktu nyata. Akan tetapi, tingkat akurasi ataupun konsistensi sinyal lampu yang ditampilkan belum tinggi. Hal ini dikarenakan komunikasi data antara back-end server yang menyimpan ke database mempunyai jalur dan mekanisme yang berbeda dengan front-end server untuk visualisasi. Untuk meningkatkan akurasi, maka kemungkinan perbaikan solusi yang bisa dilakukan pada penelitian selanjutnya adalah mengubah skema komunikasi data antara modul controller, back-end server, MQTT server, dan front-end server menjadi satu jalur. Selain itu, kemungkinan perbaikan solusi yang bisa juga dilakukan pada penelitian selanjutnya adalah menempatkan MQTT server dan front-end server berada di server yang sama, dengan tujuan mengurangi potensi *data loss*.

Daftar Pustaka

[1] A. Kumar and R. K. Singh, "Comparative Analysis of AngularJS and ReactJS," *International Journal of Latest Trends in Engineering and Technology*, vol. 7, no. 4, 2016.

[2] V. A. M and P. Sonpakti, ReactJS by Example - Building Modern Web Application with React, Birmingham: Packt Publishing, 2016.

[3] A. Ivanov, Z. Aufar, T. Tatyana and K.-H. Hsia, "Online Monitoring and Visualization with ROS and ReactJS," in *2021 International Siberian Conference on Control and Communications (SIBCON)*, Kazan, 2021.

[4] A. Javeed, "Performance Optimization Techniques for ReactJS," in *2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, Coimbatore, 2019.

[5] V. Lampkin, W. T. Leong, L. Olivera, S. Rawat, M. Subrahmanyam and R. Xiang, Building Smarter Planet Solutions with MQTT and IBM Websphere MQ Telemetry, IBM Redbooks, 2012.

[6] D. Locke, MQ Telemetry Transport (MQTT) V3.1 Protocol, 2010.

[7] A. M. Zambrano, I. Perez, C. Palau and M. Esteve, 2016.

[8] K. Chooruang and P. Mangkalakeeree , "Wireless Heart Rate Monitoring System using MQTT," in *International Electrical Engineering Congress*, Chiang Mai, 2016.

[9] T. Haltalkar , S. Bhore , K. Borawake and S. Naik , "GSM Base Home Automation using MQTT," *International Journal of Engineering Tecnology, Management and Applied Science*, vol. 3, no. 9, 2015.

[10] H. W. Chen and F. J. Lin, "Converging MQTT Resources in ETSI Standards Based M2M Platform," in *2014 IEEE International Conference on Internet of Things (iThings), and*

IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom), Taipei, 2014.

[11] O. Sadio, I. Ngom and C. Lishou, "Lightweight Security Scheme for MQTT/MQTT-SN Protocol," in *2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*, Granada, 2019.