Optimasi Ukuran Data pada Sistem Monitoring Baterai Lithium Berbasis IoT Menggunakan Protokol MQTT dan Metode Data Mapping

1st Tsaqib Sayyidan Sendjaja
Fakultas Informatika
Universitas Telkom
Bandung,Indonesia
tsaqibsayyidans@student.telkomuniversity.ac.id

2nd Fazmah Arif Yulianto
Fakultas Informatika
Universitas Telkom
Bandung,Indonesia
fazmaharif@telkomuniversity.ac.id

***Abstrak* — Tujuan penelitian ini adalah mengoptimasi ukuran data yang dikirim dari perangkat IoT pada sistem monitoring baterai lithium ke server. Dalam penelitian dilakukan perbandingan ukuran data dari beberapa skenario yang terdiri dari penggunaan protokol HTTP dan MQTT serta penerapan data mapping pada format data JSON. Penelitian disimulasikan menggunakan Raspberry Pi sebagai perangkat IoT dan perangkat lunak Wireshark untuk pengukuran data. Hasil pengukuran menunjukkan pengubahan protokol menjadi MQTT dan pengunaan data mapping berhasil menurunkan ukuran data yang dikirimkan dari IoT ke server sebesar 13,62%, 73,77% dan 87,62% untuk masing-masing skenario HTTP data mapping, MQTT JSON dan MQTT data mapping dibandingkan dengan skenario HTTP JSON. Penurunan ukuran data menyebabkan berkurangnya biaya jaringan pada sistem monitoring baterai lithium karena biaya jaringan mengikuti besarnya ukuran data.**

***Kata kunci*— IoT, HTTP, MQTT, data mapping**

I. PENDAHULUAN

Saat ini para operator telekomunikasi di Indonesia seperti Telkomsel, XL, 3, Indosat, dan lain-lain sudah memiliki jaringan telekomunikasi yang mencakup hampir seluruh pelosok wilayah di Indonesia. Jaringan telekomunikasi tersebut terdiri dari BTS (Base Transceiver Station), sistem transmisi, BSC (Base Station Controller) dan MSC (Mobile Switching Center).

Perangkat-perangkat jaringan telekomunikasi tersebut harus selalu beroperasi termasuk saat pasokan listrik PLN berhenti agar layanan telepon dan data tetap dapat digunakan oleh pelanggan. Untuk memenuhi kebutuhan itu digunakan baterai lithium sebagai perangkat daya cadangan. Untuk memastikan kinerja baterai tersebut optimal, diperlukan suatu sistem monitoring untuk memantau kondisi baterai [1].

Kondisi baterai direpresentasikan dengan parameter-parameter yang terdiri dari SoC (State of Charge), tegangan baterai, arus baterai, tegangan minimum sel baterai, tegangan maksimum sel baterai, jumlah siklus, backup time, temperatur maksimum sel baterai, temperatur minimum sel baterai, dan lain-lain. Sistem monitoring tersebut juga harus memiliki kemampuan online monitoring, bisa diakses dimana saja kapan saja dan memiliki jangkauan yang luas di seluruh lokasi di Indonesia. Untuk memenuhi persyaratan tersebut, maka dibutuhkan suatu sistem monitoring berbasis IoT (Internet of Things) yang menggunakan jaringan yang bisa diakses di seluruh wilayah di Indonesia seperti jaringan selular. Namun, penggunaan jaringan tersebut tentu dikenakan biaya. Semakin besar data sistem monitoring yang ditransmisikan maka semakin besar pula biaya yang dikenakan.

Saat ini sudah ada sistem monitoring baterai lithium yang berbasis HTTP dan JSON. Sistem monitoring yang sudah beroperasi ini menggunakan jaringan selular yang dikenakan biaya untuk mengirimkan data dari perangkat IoT ke server. Di sisi lain, banyak juga sistem monitoring lainnya yang menggunakan protokol MQTT. MQTT memiliki keunggulan pada ukuran header dan overhead protokol yang lebih kecil dibandingkan dengan HTTP [2] [3]. Berdasarkan hal tersebut, ukuran data akan dipengaruhi oleh protokol serta jumlah karakter JSON yang digunakan.

Tujuan dari penelitian ini adalah untuk menemukan solusi yang dapat mengoptimalkan data yang ditransmisikan pada sistem monitoring baterai lithium berbasis IoT dengan memilih protokol komunikasi dan/atau menggunakan metode pengkodean terhadap data yang dikirimkan.

II. KAJIAN TEORI

A. Ringkasan Kajian Pustaka

[2]Proceeding Bharati Wukkadada, Kirti Wankhede, Ramith Nambiar, Amala Nair membandingkan protokol HTTP dan MQTT untuk menentukan mana yang memiliki latency lebih rendah dan lebih modular.

[4]Jurnal Harpreet Singh Padda, Gulabchand K. Gupta membahas cara mengkompres JSON menggunakan data maps. Hasil pengkompresan tersebut dibandingkan dengan format data lain. Hasil dari jurnal ini menunjukkan bahwa penerapan data maps cukup efektif terhadap ukuran data JSON jika paket data memiliki lebih banyak pasangan key-value dan special character dan hasil kompresi data JSON menggunakan data maps berbanding lurus dengan banyaknya pasangan key-value dan special character pada paket data JSON.

[3]Nindithia Putri Windryani, Dr. Nyoman Bogi A. K., S.T., MSEE., Ratna Mayasari, S.T., M.T. membahas tentang

implementasi protokol MQTT untuk IoT Platform Patriot sebagai solusi yang lebih baik dibandingkan HTTP. MQTT merupakan protokol komunikasi yang sangat sederhana dan ringan. Protokol MQTT juga didesain untuk alat berkemampuan terbatas, bandwidth yang rendah, latency yang tinggi dan jaringan yang kurang dapat diandalkan. Hasil pengujian QoS dengan parameter delay diperoleh rata-rata delay MQTT QoS 0 sebesar 0,0017s, QoS 1 sebesar 0,0628805s, MQTT QoS 2 sebesar 0,16987s dan HTTP sebesar 0,124591s. Packet loss yang di dapatkan sebesar 0% untuk MQTT QoS 1, QoS 2 dan HTTP sedangkan QoS 0 sebesar 13,3333%, nilai throughput protokol MQTT lebih rendah +/-324,7943 Bytes/s dibandingkan protokol HTTP sehingga protokol MQTT dapat lebih reliable berjalan pada keadaan bandwidth rendah atau latency tinggi dibandingkan protokol HTTP.

[5]Matz Andreas Philipp, Fernandez-Prieto Jose-Angel, Cañada-Bago Joaquin, dan Birkel Ulrich membahas tentang pengukuran yang sistematis terhadap physical layer pada Narrowband-IoT (NB-IoT) yang merupakan bagian dari teknologi akses Low-Power Wide Area Networks (LPWANs), yang menawarkan solusi dengan konsumsi energi yang efisien dengan jangkauan yang jauh untuk aplikasi perangakat IoT. Selain itu juga dijelaskan bahwa aspek radio parameter sangat mempengaruhi QoS dari application layer. Hasil penelitian menunjukan bahwa secara teoritis, NB-IoT bisa memenuhi tujuan aspek komersial dari Third Generation Partnership Project (3GPP) dengan memberikan area cakupan yang luas sehingga meningkatkan sensitivitas receiver, tetapi di saat yang bersamaan juga meningkatkan system latency. Data rate maksimum juga stabil. Secara umum NB-IoT reliable dan fleksibel untuk aplikasi IoT bahkan pada kondisi propagasi radio yang tidak ideal.

[6]Turc Traian mengusulkan dalam makalahnya untuk menggunakan HTTP pada aplikasi sebagai solusi terhadap pertumbuhan perangkat IoT yang sangat pesat, standar dan spesifikasi perangkat IoT yang seringkali tidak tersedia dan tidak mendukung protokol yang rumit. Studinya menggunakan sebuah perangkat IoT yang mengimplementasikan subset minimal HTTP, koneksi ke sebuah URL dan membaca data yang diterima dari URL. Antar perangkat IoT tidak berkomunikasi secara langsung tetapi berkomunikasi melalui aplikasi berbasis web dengan akses ke database yang menyimpan data yang dikumpulkan oleh perangkat-perangkat Iot tersebut.

[7]Gonzalez Isaias, Calderon Antonio Jose, Folgado, Francisco Javier menjelaskan bahwa aktivitas monitoring dan akuisisi data diperlukan untuk menunjang pengawasan dan tracking dari baterai lithium. Dalam makalah mereka menjelaskan suatu sistem monitoring bernama Grafana yang berbasis aplikasi IoT digunakan untuk memonitor baterai lithium. Pengguna Grafana bisa mengakses grafik dan data numerik secara real time tentang baterai lithium seperti arus, tegangan, suhu, state of charge, dan lain-lain.

[8]McManus Sean, Cook Mike merilis buku panduan tentang penggunaan Raspberry Pi yang akan digunakan dalam penelitian. Raspberry memenuhi persyaratan untuk dijadikan sebagai model perangat IoT untuk monitoring baterai lithium sekaligus diinstal Wireshark sebagai alat untuk mengukur QoS aplikasi IoT tersebut.

[9]Atmoko R A, Riantini R, and Hasin M K mengusulkan penggunaan protokol MQTT sebagai protokol komunikasi untuk IoT. Studinya menggunakan data suhu dan kelembapan, dengan akuisisi data secara real time dan disimpan dalam database MySQL. Studi ini juga dilengkapi dengan aplikasi antar muka berbasis web dan mobile untuk monitoring secara online. Kesimpulannya menunjukan bahwa penggunaan protokol MQTT memberikan kualitas data yang baik dan andal.

B. MQTT

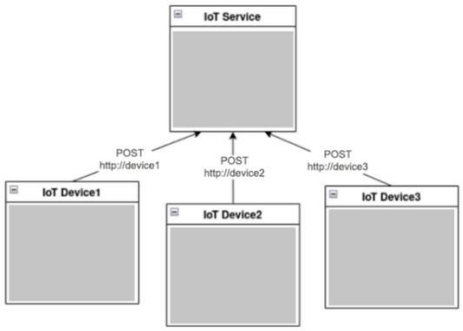
MQTT adalah suatu protokol komunikasi dengan fitur-fitur yang dibuat spesifik untuk solusi IoT. Beberapa karakteristik utamanya adalah:

1. Menggunakan koneksi TCP
2. Bertujuan untuk meminimalkan data overhead untuk setiap paket MQTT
3. Nilai data terakhir untuk perangkat dapat disimpan (retained message)
4. Notifikasi jika client mengalami disconnect.
5. Arah message bisa Bi-directional. Data yang dikirim dari dan ke perangkat dapat menggunakan koneksi TCP yang sama.
6. Publish subscribe routing, yang membuat penambahan baik pembuat data atau pengkonsumsi data menjadi mudah.

MQTT commands adalah Connect, Subscribe, Publish, Unsubscribe, dan Disconnect. Suatu MQTT Topics adalah unit distribusi di mana semua clients dapat Publish dan Subscribe. Semua subscriber yang memiliki otorisasi untuk ke suatu topik akan menerima semua message yang terpublikasikan.

C. Penggunaan HTTP untuk IoT

HTTP (Hypertext Transfer Protocol) protokol untuk transfer dokumen pada World Wide Web dan membuat web browser bisa digunakan. Clients dapat melakukan request: GET, PUT, DELETE and POST pada suatu server dengan URL tertentu. Pada pemakaian biasa, suatu web browser akan mengambil halaman-halaman web dari suatu server dengan metode GET, sedangkan pada penggunaan IoT, HTTP digunakan unutk membuat semua perangkat dapat mengirimkan POST ke suatu server yang mewakili suatu layanan berbasis IoT.



GAMBAR 2.1.
Penggunaan HTTP pada IoT (Sumber: https://www.hivemq.com/blog/mqtt-vs-http-protocols-in-iot-iiot/)

Berikut Karakteristik HTTP:

TABEL 2.1.
Karakteristik HTTP
(Sumber: https://www.hivemq.com/blog/mqtt-vs- http-protocols-in-iot-iiot/)

Full name	Hyper Text Transfer Protocol
Architecture	Request response
Command targets	URIs
Underlying Protocol	TCP/IP
Secure connections	TLS + username/password (SASL support possible)
Client observability	Unknown connection status
Messaging Mode	Synchronous
Message queuing	Application needs to implement
Message overhead	8 bytes minimum (header data is text - compression possible)
Message Size	No limit but 256MB is beyond normal use cases anyway.
Content type	Text (Base64 encoding for binary)
Message distribution	One to one
Reliability	Diimplementasi pada aplikasi

D. Perbandingan Unjuk Kerja MQTT dengan HTTP

Berdasarkan jumlah Data Overhead dan response time, MQTT memiliki keunggulan bila dibandingkan dengan HTTP. Hal ini karena pada MQTT, dalam satu koneksi TCP tidak perlu di-setup overhead untuk setiap message sementara pada HTPP harus di-setup overhead untuk setiap message yang dikirim. Hal ini mengakibatkan kebutuhan bandwidth dan response time pada MQTT bisa lebih kecil dibandingan HTTP.

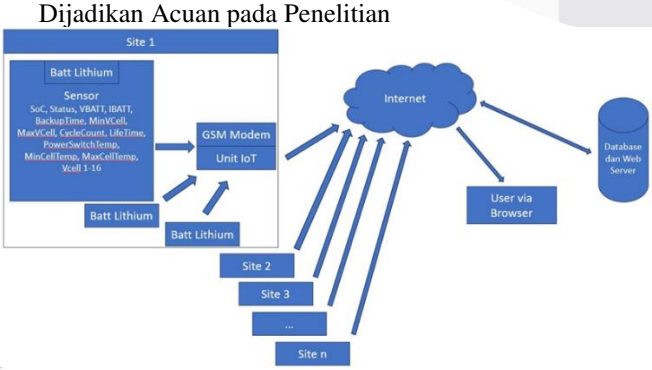
E. Keuntungan HTTP pada IoT

Keuntungan utama penggunaan HTTP untuk IoT adalah protokol dan teknologinya sudah dikenal luas oleh para developers. Libraries untuk aplikasi pada clients dan server juga sudah banyak tersedia. Walaupun MQTT memiliki keunggulan dari sisi kehematan bandwidth dan response time, tetapi penggunaan HTTP yang luas pada sistem IoT juga menunjukan bahwa penggunaan HTTP pada sistem IoT masih merupakan suatu solusi yang andal dan mudah diterapkan dan dapat diterima oleh para pengguna.

Pada makalah penelitian [3]Nindithia Putri Windryani, Dr. Nyoman Bogi A. K, S.T., MSEE., Ratna Mayasari, S.T., M.T. dijelaskan bahwa packet loss pada sistem IoT Berbasis Patriot dengn menggunakan HTTP adalah nol. Hal ini menunjukan bahwa HTTP cukup andal digunakan untuk sistem IoT. Selain itu nilai response time untuk HTTP juga masih lebih baik dibandingkan MQTT pada QoS 2.

III. PERANCANGAN SISTEM

A. Sistem Monitoring Baterai Lithium yang Sudah Ada dan Dijadikan Acuan pada Penelitian



GAMBAR 3.1.
Diagram blok sistem monitoring baterai lithium yang sudah ada

Pada sistem yang sudah ada, sensor akan membaca kondisi baterai lithium dan sekitarnya. Kondisi yang dipantau terdiri dari SoC (State of Charge), status, tegangan baterai, arus baterai, backup time, tegangan minimum sel baterai, tegangan maksimum sel baterai, jumlah siklus, life time, temperature mosfet baterai, temperatur minimum sel baterai, temperatur maksimum sel baterai dan tegangan tiap sel baterai. Kondisi yang dibaca oleh sensor kemudian diproses oleh perangkat IoT dan diubah menjadi format JSON. Perangkat IoT kemudian akan mengirimkan data JSON tersebut ke server. Sistem yang sudah ada memonitor kondisi baterai lithium yang sudah banyak terpasang di seluruh pelosok Indonesia. Sistem yang sudah ada menggunakan protokol HTTP karena developer sistem yang sudah ada sudah terbiasa dengan dan mengenal protokol dan teknologi HTTP secara mendalam. Selain itu, libraries untuk aplikasi pada clients dan server juga sudah banyak tersedia.

Data yang dikirim dari node IoT ke server pada sistem monitoring baterai lithium yang sudah ada dapat dilihat pada tabel berikut.

TABEL 3.1.
Parameter sistem monitoring baterai lithium

Parameter Sistem Monitoring Baterai Lithium				
Parameter	Keterangan	Nilai Min	Nilai Max	Ukuran data JSON (B)
SOC (%)	State of charge (persentase kapasitas baterai terisi)	0	100	1-5
STATUS	Fungsionalitas baterai	BATT. FAIL	READY	5-10
VBATT (V)	Tegangan pack baterai	42,9	54,6	2-5
IBATT (A)	Arus pack baterai	-150	150	1-7
BACKUP TIME (HOUR)	Durasi baterai dapat digunakan sebagai cadangan	0	depend on load [arus batt (-)]	1-5
MIN VCELL (V)	Tegangan minimum sel baterai	0	4,2	1-5
MAX VCELL (V)	Tegangan maksimum sel baterai	0	4,2	1-5
CYCLE COUNT	Jumlah siklus charge-discharge	0	65.535	1-5
LIFE TIME	Durasi baterai beroperasi sejak dipasang	0	4.294.967.295	1-13
POWER SWITCH TEMP (C)	Temperatur mosfet baterai	0	100	1-4
MIN CELL TEMP (C)	Suhu minimum sel baterai	0	100	1-4
MAX CELL TEMP (C)	Suhu maksimum sel baterai	0	100	1-4
VCELL1-16 (V)	Tegangan tiap sel baterai	0	4,2	1-5

Pengiriman data dari perangkat IoT pada sistem yang sudah ada menggunakan format JSON sebagai berikut.


```
{
  "SOC":88.45,
  "BATT_VOLTAGE":52.99,
  "BATT_CURRENT":-0.05,
  "MIN_CELL_VOLT":4.073,
  "MAX_CELL_VOLT":4.08,
  "CYCLE_COUNT":6,
  "BACKUP_TIME":17.69,
  "MAX_CELL_TEMP":34,
  "MIN_CELL_TEMP":33.6,
  "POWER_SWITCH_TEMP":40.3,
  "LIFE_TIME":6620.05,
  "NOM_CAPACITY":200,
  "status_batt":3,
  "V_CELL_1":4.075,
  "V_CELL_2":4.073,
  "V_CELL_3":4.075,
  "V_CELL_4":4.075,
  "V_CELL_5":4.075,
  "V_CELL_6":4.075,
  "V_CELL_7":4.076,
  "V_CELL_8":4.076,
  "V_CELL_9":4.073,
  "V_CELL_10":4.08,
  "V_CELL_11":4.08,
  "V_CELL_12":4.076,
  "V_CELL_13":4.076,
  "V_CELL_14":0.052,
  "V_CELL_15":0.050,
  "V_CELL_16":0.051,
  "time_unix":1621835748,
  "rssi":-79,
  "tegangan":52.3,
  "suhu":0,
  "humi":0
}
```

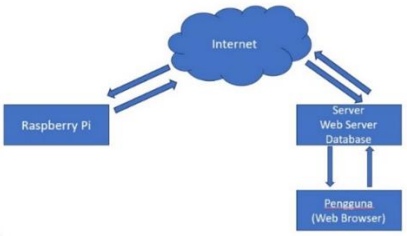

GAMBAR 3.2.
Format data sistem yang sudah ada

Pada sistem yang ada, data JSON berisi 34 pasangan key-value dari kondisi baterai dan sekitarnya yang dipantau. Data JSON dikirim dengan tipe string sehingga tiap karakter pada gambar di atas memiliki ukuran 1 byte. Pengiriman data

dilakukan dengan interval 5 menit pada sistem yang sudah ada.

B. Penelitian

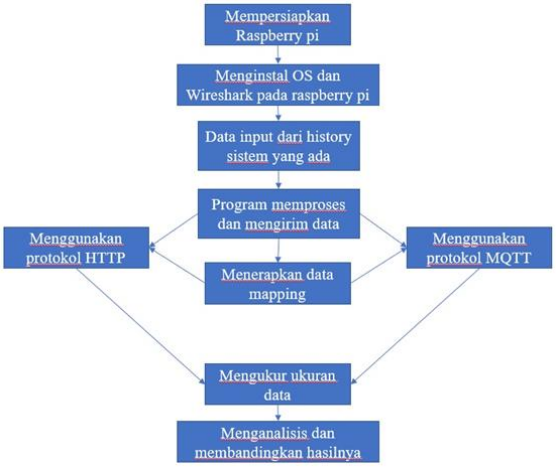
Pada penelitian, dibuat model dari sistem yang sudah ada seperti pada gambar berikut:



GAMBAR 3.3.
Diagram sistem yang akan digunakan pada penelitian

Pada penelitian, perangkat IoT akan digantikan oleh Raspberry Pi untuk pengolahan data. Data yang dikirimkan pada simulasi diambil dari histori data sistem yang sudah ada agar dapat menggambarkan skenario sesungguhnya. Parameter yang digunakan pada penelitian sebanyak 28 parameter yang merepresentasikan kondisi baterai lithium, sementara parameter kondisi lingkungan (suhu dan kelembaban), rssi, time_unix dan informasi daya sistem (nom capacity dan tegangan sistem) tidak digunakan karena tidak ada kaitannya dengan kondisi baterai lithium. Pada simulasi, digunakan server sama seperti sistem yang sudah ada yang menggunakan OS Linux Ubuntu, HTTP server menggunakan Apache, PHP 7.3, MySQL/MariaDB dan Laravel serta MQTT server menggunakan Java 1.8.

Berikut flow diagram skenario penelitian:

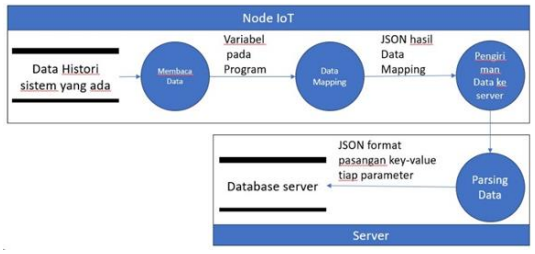


GAMBAR 3.4.
Skenario penelitian

Pada penelitian akan dibandingkan ukuran data dari beberapa skenario berbeda dari kombinasi pengubahan protokol dan pemrosesan data serta penerapan data mapping.

Pada penelitian akan menggunakan 2 jenis protokol, yaitu HTTP dan MQTT dengan QoS 0 karena memiliki ukuran data yang paling kecil dibanding QoS 1 dan 2 dan jaringan yang digunakan cukup andal. Namun, penggunaan QoS 0 memiliki risiko adanya data yang tidak sampai.

Berikut proses *data mapping*:



GAMBAR 3.5.
Proses pada data

Untuk mengurangi ukuran data, digunakan metode data mapping dengan tujuan mengurangi jumlah karakter pada JSON menjadi hanya memiliki satu pasang key-value. Metode ini diterapkan pada tahap pemrosesan dengan cara nilai dari masing-masing parameter digabung ke dalam satu pasang key-value yang diawali dan diakhiri “tanda bintang” dan nilai tiap parameter dipisahkan dengan “tanda tagar”. Pada server, JSON yang diterima dikembalikan ke masing-masing pasangan key-value parameter semula. Algoritma data mapping pada node IoT dan server dipaparkan sebagai berikut:

Algoritma program node IoT:

a. Program membaca data histori yang disimpan dalam format xlsx yang memiliki 28 parameter.
b. Tiap parameter di-assign ke variabel masing-masing.
c. Program mengakses nilai tiap parameter secara berurutan.
d. Nilai tiap parameter digabungkan ke dalam satu variabel baru bertipe string yang diawali dan diakhiri tanda bintang dan nilai tiap parameter dipisah tanda tagar.
e. Program mengirim data ke server dalam format data mapping hasil proses pada poin d.
f. Program mengulangi poin c-e sampai semua data terkirim.

Algoritma program server:

g. Server menerima data JSON dengan format pada poin d.
h. Data JSON di-assign ke variabel bertipe string dengan menghilangkan tanda bintang.
i. Isi variabel pada poin h diubah ke dalam bentuk array yang elemennya diambil dari pecahan string yang dipisahkan oleh tanda tagar.
j. Array hasil dari poin i di-assign tiap elemennya ke masing-masing variabel sesuai dengan parameternya.
k. Variabel dan nilai parameter disimpan ke database dan kemudian ditampilkan di web.

Proses data mapping membentuk data dalam format JSON yang hanya memiliki satu pasang key-value untuk 28 parameter baterai lithium dengan menggabungkan nilai tiap parameter ke dalam satu string yang dipasangkan dengan satu key. Hal ini membuat jumlah karakter yang dikirim dari IoT ke server berkurang.

Berikut data JSON tanpa data mapping.

JSON DATA:{'SOC': 91.53, 'STATUS': 'READY', 'vBatt': 53.04, 'iBatt': 0.03, 'backupTime': 13.73, 'minVCell': 4.067, 'maxVCell': 4.092, 'cycleCount': 6, 'lifeTime': 10656.61, 'powerSwitchTemp': 30.2, 'minCellTemp': 28.1, 'maxCellTemp': 28.9, 'vCell1': 4.067, 'vCell2': 4.077, 'vCell3': 4.077, 'vCell4': 4.077, 'vCell5': 4.078, 'vCell6': 4.079, 'vCell7': 4.08, 'vCell8': 4.08, 'vCell9': 4.092, 'vCell10': 4.084, 'vCell11': 4.084, 'vCell12': 4.082, 'vCell13': 4.081, 'vCell14': 0.047, 'vCell15': 0.047, 'vCell16': 0.047}

Berikut JSON hasil data mapping yang digunakan pada penelitian.

JSON DATA:{"data": "*91.53#READY#53.04#0.03#13.73#4.067#4.092#6#10656.61#30.2#28.1#28.9#4.067#4.077#4.077#4.077#4.078#4.079#4.08#4.08#4.092#4.084#4.084#4.082#4.081#0.047#0.047#0.047*"}

Berikut potongan program bagian proses data mapping pada node IoT dan parsing pada server.

```
data = "*{}#{}#{}#{}#{}#{}#{}#{}#{}#{}#{}#{}#{}#{}#{}#{}#{}#{}#{}#{}#{}#{}#{}#{}#{}#{}#{}#{}*".format(
    soc[idx], status[idx], vBatt[idx], iBatt[idx],
    backupTime[idx], minVCell[idx], maxVCell[idx], cycleCount[idx],
    lifeTime[idx], powerSwitchTemp[idx], minCellTemp[idx], maxCellTemp[idx],
    vCell1[idx], vCell2[idx], vCell3[idx], vCell4[idx],
    vCell5[idx], vCell6[idx], vCell7[idx], vCell8[idx],
    vCell9[idx], vCell10[idx], vCell11[idx], vCell12[idx],
    vCell13[idx], vCell14[idx], vCell15[idx], vCell16[idx],
```

GAMBAR 3.6.
Potongan program node IoT bagian proses data mapping

```
public function store($client_id, Request $request)
{
    //insert into inbox
    \DB::table('inbox')->insert([
        'client_id' => $client_id,
        'payload'   => $request->get('data'),
        //'updated_at' => Carbon::now()
    ]);

    //update_client table
    \DB::table('clients')
        ->where('id', '=', 1)
        ->update([
            'name'       => $client_id,
            'updated_at' => Carbon::now(),
        ]);

    $json_data = trim($request->get('data'), '*');

    //update monitoring
    [
        $soc, $status, $vbatt, $ibatt, $backup_time, $min_vcell, $max_vcell, $cycle_count, $life_time, $power_switch_temp,
$min_cell_temp, $max_cell_temp, $vcell1, $vcell2, $vcell3, $vcell4, $vcell5, $vcell6, $vcell7, $vcell8, $vcell9, $vcell10, $vcell11,
$vcell12, $vcell13, $vcell14, $vcell15, $vcell16,
    ] = explode("#", $json_data);

    $monitoring_data = explode("#", $json_data);
```

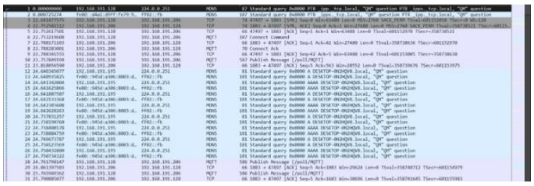
GAMBAR 3.7.
Potongan program server untuk parsing JSON data mapping

Tiap skenario simulasi diukur ukuran datanya menggunakan Wireshark. Pada Raspberry Pi dipasang Wireshark untuk menangkap traffic data antara perangkat IoT dengan server. Kemudian, hasil pengukuran dari tiap skenario dibandingkan dan dianalisis.

IV. EVALUASI

Pengiriman data dari perangkat IoT ke server menggunakan jaringan internet baik menggunakan protokol HTTP maupun MQTT selalu dimulai dengan proses set-up koneksi yang terdiri dari proses routing sampai koneksi berhasil dibangun. Jika kondisi jaringan tidak ideal, terkadang koneksi terputus atau muncul delay yang cukup besar. Protokol HTTP dan MQTT akan mengirimkan beberapa retransmisi di luar paket data yang dikirimkan. Pengukuran pada Wireshark akan menangkap semua paket data yang terkirim dari IoT ke server termasuk routing dan retransmission. Berikut hasil tangkapan Wireshark terhadap data yang dikirim dari IoT ke server dengan protokol HTTP dan MQTT menggunakan data histori selama 1 bulan dari 1 pack baterai lithium yang dimonitor pada sistem yang sudah ada. Data histori ini diperoleh dari pihak perusahaan yang memiliki sistem monitoring baterai lithium dalam format xlsx yang terdiri dari 28 kolom dan 8898 baris data.

GAMBAR 4.1.
Hasil capture Wireshark menggunakan protokol HTTP



GAMBAR 4.2.
Hasil capture Wireshark menggunakan protokol MQTT

A. Hasil Pengukuran Data

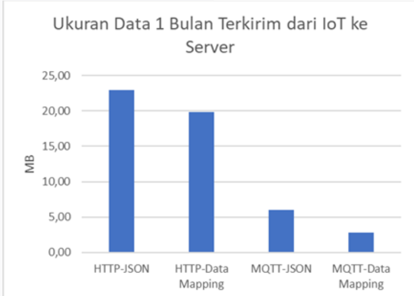
Pada penelitian digunakan empat skenario yaitu HTTP JSON, HTTP data mapping, MQTT JSON dan MQTT data mapping. Skenario HTTP JSON merupakan simulasi pengiriman data yang menggunakan protokol HTTP dan format data JSON seperti pada sistem yang sudah ada (tanpa data mapping). HTTP data mapping menggunakan protokol HTTP dan format data JSON hasil proses data mapping. MQTT JSON menggunakan protokol MQTT dan format data JSON tanpa data mapping. MQTT data mapping menggunakan protokol MQTT dan format data JSON hasil data mapping. Hasil pengukuran dan perbandingan ukuran data dari keempat skenario tersebut dapat dilihat pada tabel dan grafik berikut.

TABEL 4.1.
Tabel ukuran data hasil pengukuran

	Hasil Pengukuran dari Wireshark			
	HTTP-JSON	HTTP-Data Mapping	MQTT-JSON	MQTT-Data Mapping
Ukuran Data (MB)	22,92	19,80	6,01	2,84
Packet Loss	0	0	0	0
Jumlah Paket	115.302	111.362	22.221	18.806

TABEL 4.2.
Tabel penghematan ukuran data

Penghematan Ukuran Data	Selisih (MB)	Penghematan (%)
MQTT-JSON terhadap HTTP-JSON	16,91	73,77%
MQTT-data mapping terhadap HTTP-data mapping	16,96	85,66%
HTTP-data mapping terhapad HTTP-JSON	3,12	13,62%
MQTT-data mapping terhadap MQTT-JSON	3,17	52,79%
MQTT-data mapping terhadap HTTP-JSON	20,08	87,62%



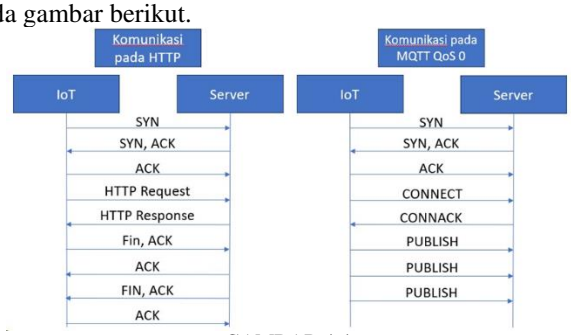
GAMBAR 4.3.
Grafik perbandingan ukuran data hasil pengukuran

B. Analisis Hasil Pengukuran Data

Pengiriman data menggunakan HTTP JSON memperoleh ukuran data sebesar 22,918 MB. Pengiriman data menggunakan HTTP data mapping memperoleh ukuran data sebesar 19,796 MB, yang berarti 13,62% lebih rendah dari HTTP JSON. Pengiriman data menggunakan MQTT JSON memperoleh ukuran data sebesar 6,012 MB, yang berarti 73,77% lebih rendah dari HTTP JSON. Sementara pengiriman data menggunakan MQTT data mapping memperoleh ukuran data sebesar 2,838 MB yang berarti 87,62% lebih rendah dari HTTP JSON dan 52,79% lebih rendah dari MQTT JSON.

Pengubahan protokol dari HTTP ke MQTT menghasilkan penurunan ukuran data yang cukup signifikan yaitu 16,91 MB pada format JSON dan 16,96 MB pada format data mapping. Penurunan ini dapat dicapai karena ukuran overhead protokol MQTT jauh lebih kecil dibanding HTTP.

Hal tersebut diakibatkan HTTP perlu membuka dan menutup koneksi setiap proses pengiriman data sementara MQTT cukup sekali membuka dan menutup koneksi seperti dilihat pada gambar berikut.



GAMBAR 4.4.
Komunikasi pada HTTP dan MQTT

Penerapan data mapping menghasilkan penurunan ukuran data sebesar 3,12 MB pada protokol HTTP dan sebesar 3,17 MB pada protokol MQTT. Penurunan ini dapat dicapai karena jumlah karakter JSON yang dikirimkan lebih sedikit dengan menggunakan data mapping.

C. Dampak Terhadap Biaya

Dari hasil pengukuran di atas, disimulasikan dampaknya terhadap biaya menggunakan konfigurasi 1 site menggunakan 3 pack baterai lithium (Power consumption per 1 site = 3000 Watt, backup time 4 jam pada tegangan 48 V DC membutuhkan 3 pack baterai lithium masing-masing 100AH) dapat dilihat hasilnya pada tabel berikut.

TABEL 4.3.
Simulasi perbandingan biaya tiap skenario

No.	Skenario	Ukuran Data/pack Batt Li(MB)	Ukuran Data/ 3 pack Batt Li(MB)	Paket M2M up to (MB) operator selular	Biaya/bulan /site (Rp)	Biaya/100 site/bulan (Rp)
1	HTTP-JSON	22,92	68,75	75	45.000	4.500.000
2	HTTP-data mapping	19,80	59,39	75	45.000	4.500.000
3	MQTT-JSON	6,01	18,04	25	25.000	2.500.000
4	MQTT-data mapping	2,84	8,51	10	20.000	2.000.000

Berdasarkan simulasi biaya keempat skenario di atas, penerapan protokol MQTT dan data mapping pada sistem monitoring baterai lithium dapat menurunkan biaya jaringan karena bisa menggunakan paket M2M up to 10 MB operator seluler yang lebih murah.

V. KESIMPULAN

Pengubahan protokol menjadi MQTT dan pengunaan data mapping berhasil menurunkan ukuran data yang dikirimkan dari IoT ke server sebesar 13,62%, 73,77% dan 87,62% untuk masing-masing penerapan HTTP data mapping, MQTT JSON dan MQTT data mapping terhadap HTTP JSON.

Penurunan ukuran data menyebabkan berkurangnya biaya jaringan pada sistem monitoring baterai lithium karena biaya jaringan mengikuti besarnya ukuran data.

REFERENSI

[1] M. Dwiyaniti, L. R. N. Kusuma, Silawardono, S. L. Kusumastuti, A. R. Wiguna and Tohazen, "A Real-time Performance Monitoring of IoT-based on Lithium-Ion Battery Pack," in *2022 International Conference on Informatics Electrical and Electronics (ICIEE)*, Yogyakarta, 2022.

[2] B. Wukkadada, K. Wankhede, R. Nambiar and A. Nair, "Comparison with HTTP and MQTT In Internet of Things (IoT)," in *Proceedings of the International Conference on Inventive Research in Computing Applications (ICIRCA 2018)*, Coimbatore, 2018.

[3] N. P. Windryani, N. Bogi and R. Mayasari, "Analisa Perbandingan Protokol MQTT dengan HTTP pada IoT Platform Patriot," *e-Proceeding of Engineering,* vol. VI, no. 2, pp. 3192-3199, 2019.

[4] H. S. Padda and G. K. Gupta, "Compressing JSON using Data maps," *International Journal of Emerging Trends & Technology in Computer Science (IJETTCS),* vol. V, no. 1, pp. 59-61, 2016.

[5] A. P. Matz, J.-A. Fernandez-Prieto, J. Canada-Bago and U. Birkel, "A Systematic Analysis of Narrowband IoT Quality of Service," *Sensors,* vol. XX, no. 6, pp. 1-26, 2020.

[6] T. Turc, "Internet of Things Based on HTTP," *Scientific Bulletin of the Petru Maior University of Tirgu Mures,* vol. XV, no. 2, pp. 4-8, 2019.

[7] I. Gonzales, A. J. Calderon and F. J. Folgado, "IoT Real Time System for Monitoring Lithium-Ion Battery Long-Term Operation in Microgrids," *Journal of Energy Storage,* vol. LI, pp. 1-16, 2022.

[8] S. McManus and M. Cook, Raspberry Pi for Dummies 3rd Edition, Hoboken: John Wiley & Sons, Inc, 2017.

[9] R. A. Atmoko, R. Riantini and M. K. Hasin, "IoT real-time data acquisition using MQTT protocol," *Journal of Physics Conference Series,* vol. DCCCLIII, no. 1, 2017.