

Rancang Bangun Perangkat Lunak IoT *Gateway* dari *Field* ke *Cloud* Berbasis *Protocol* Komunikasi MQTT

Billy Hamsen ¹⁾, Indrastanti R. Widiarsari ²⁾

^{1,2)} Fakultas Teknologi Informasi Universitas Kristen Satya Wacana
Jl. Dr. O. Notohamidjojo 1-10, Salatiga 50711, Indonesia
Email: 672016150@student.uksw.edu ¹⁾, indrastanti@uksw.edu ²⁾

Received: 14-04-2021 Riwayat artikel:
Revised: 03-07-2021 Accepted: 13-09-2021

Abstract

Internet of things (IoT) is the most important part of the development of the internet in today's world, the implementation of IoT results in various types of storage media for data or files that will grow to be very large, and will affect file storage which then results in accumulation of data. This study aims to develop the Message Queuing Telemetry Transport (MQTT) protocol, which is a protocol that runs on the TCP/IP stack and is specifically designed for machine to machine. The MQTT work system implements Publish and Subscribe data. the device will be connected to a broker and have a certain topic. This research aims to develop a system of methods by which topics on the MQTT can send messages or data to the middleware. The middleware will use the publish subscribe method to exchange messages or data. In order to receive data from the middleware, there must be a subscriber, where the Internet Gateway Device (IGD) will subscribe and the data will then be stored in the GridFS and MongoDB data storage in order to facilitate the process of getting and posting data for storage media.

Keywords: IoT, Multi-protocol, MongoDB, GridFS, data storage

Abstrak

*Internet of things (IoT) ialah bagian terpenting pada pertumbuhan internet di dunia pada masa ini, penerapan pada IoT memperoleh bermacam-macam media penyimpanan pada data atau file yang akan berkembang menjadi sangat besar, dan akan mempengaruhi penyimpanan file yang kemudian mengakibatkan pertumpukan pada data. Riset ini bertujuan untuk mengembangkan protokol *Message Queuing Telemetry Transport* (MQTT) ialah salah satu aturan yang bekerja pada *stack* TCP/IP serta diprogram khusus dalam *machine to machine*. Cara kerja MQTT melakukan *Publish* serta *Subscribe* data perangkat dapat terkoneksi kepada salah satu *Broker* serta memiliki topik khusus. Riset ini bertujuan untuk mengembangkan sistem metode yang dimana topik pada MQTT bisa mengirimkan pesan atau data kepada *middleware*. *Middleware* akan menerapkan metode *publish subscribe* untuk tukar pesan atau informasi. agar bisa menbisakan data dari *middleware* itu wajib ada *subscriber*, yang kemudian *Internet Gateway Device* (IGD) akan mensubscribe kemudian data berikutnya akan disetor ke dalam data *storage* GridFS dan MongoDB supaya mempermudah proses *get* dan *post* data untuk media penyimpanan.*

Kata Kunci: IoT, Multi-protokol, MongoDB, GridFS, data storage

Pendahuluan

Internet of Things (IoT) memiliki tujuan dalam memperluas dampak dari konektivitas internet dengan membuat suatu *device* bisa dikontrol, dikumpulkannya data dan dikirimkan. *device* pada IoT biasanya mempunyai batasan pada hal muatan penyimpanan serta daya komputasi. Konsep itu membuat *device* IoT membutuhkan hubungan pada sistem dimana sistem mempunyai daya komputasi yang diharapkan [1]. Dengan menggunakan teknologi jaringan pada masa ini diperoleh pertumbuhan yang signifikan yakni kolaborasi diantara control technology serta *Internet of Things* (IoT). *Internet of things* ialah konsep baru dimana memiliki tujuan dari hubungan untuk semuanya yakni mengurangi simpanan dan power. MQTT ialah suatu pindahan aturan yang diolah oleh *International Business Machine* (IBM) terpenting dipakai Internet of Things, MQTT dirancang agar bisa melaksanakan publish / subscribe salah satu transmisi pesan yang tidak berat. MQTT bisa dipakai dalam pengoptimalan jaringan dengan bandwidth rendah serta persediaan computing energi minimal, MQTT dipakai pada beragam-ragam desain aplikasi pada *Internet of Things* (IoT) [2]. Mengenai kemampuan yakni berbagi data, *remote control*, dan yang lainnya, meliputi juga pada benda di dunia nyata. Secara fundamental, IoT terfokus kepada benda yang bisa dinyatakan secara unik sebagai gambaran fisik pada struktur berlandaskan Internet [11]. Hal ini, data IoT bisa disatukan dari beragam sumber seperti dari berbagai data terstruktur serta tidak terstruktur. Dalam menyelesaikan permasalahan volume data serta heterogenitas pada jenis data [3].

Dalam riset ini akan merancang metode yang digunakan dalam media penyimpanan pada data. Protokol *Message Queuing Telemetry Transport* (MQTT) suatu aturan dimana dilaksanakan pada stack TCP/IP serta di desain secara khusus pada *machine to machine*. Cara kerja MQTT melaksanakan *Publish* serta *Subscribe* data. Perangkat akan terkoneksi pada sebuah *Broker* dan mempunyai suatu Topik khusus sistem riset ini di harapkan bisa menyelesaikan penyimpanan dan pengaksesan kedalam data *storage* menjadi semakin cepat dan optimal dengan baik.

Kajian Pustaka

Pada riset yang dilakukan oleh [4], dibangun sebuah rancangan manajemen penyimpanan yang disebut Internet of thing movie database (IOTMDB) dimana berlandaskan NoSQL yang dijadikan cara penyimpanan data IoT yang besar serta heterogen. Sistem IOTMDB terbagi empat, yakni master node dimana berfungsi sebagai manajer pada semua *cluster*, *standby node* dimana sebagai pengganti ketika mengalami error dalam *master node*, *data reception node* menjadi penerima data sensor, dan *slave node* digunakan kedalam menyimpan keseluruhan data.

Riset yang dilakukan oleh [5], bahwa *Gateway* bisa mengkoneksikan aturan komunikasi yang berbeda. *Gateway* bisa memiliki fungsi sebagai protokol *converter* serta dijadikan konektor beragam jenis protokol komunikasi.

Gateway serta pusat data bisa saling terhubung, itu artinya diprlukan salah satu aturan komunikasi agar bisa saling menukar informasi. Salahsatu protokol yang bisa dipakai dalam komunikasi yakni dengan protokol *Message Queuing Telemetry Transport* (MQTT). Protokol *Message Queuing Telemetry Transport* (MQTT) dipakai dikarenakan kemampuan saat dikirimkannya data serta mempunyai ciri-ciri mendukung keandalan dimana dipunyai IoT seperti bisa bekerja dalam hal *low power* serta pemakaian *bandwidth* yang rendah. Dari hasil riset ini diharapkan bisa memberikan sebuah metode untuk menyelesaikan permasalahan pada *system* komunikasi agar bisa saling terhubung satu dengan lainnya.

Riset yang dilakukan oleh [6], untuk mengatasi masalah interoperabilitas yang secara bertahap, sudah bisa menciptakan IoT middleware yang mempunyai keandalan dalam menyelesaikan persoalan *syntactical interoperability*. Dalam riset itu IoT *middleware* bisa mendapatkan data berwujud suhu serta kelembapan yang dikirim melalui protokol komunikasi yang beragam yakni *Message Queuing Telemetry Transport* (MQTT) serta *Constrained Application Protocol* (CoAP) dan juga memakai aturan jaringan Wi-Fi. Lalu data yang dikirim ke *middleware* disimpan temporer dalam redis serta dikirim ke aplikasi memakai protokol *websocket*. Dari riset ini diharapkan dapat memberikan sebuah gambaran bagaimana IoT *middleware* mengatasi interoperabilitas agar dapat menerima data dengan baik.

Riset yang dilakukan oleh [7], Sensor bisa bekerja dengan sangat baik. Yang dimana *Ethernet Shield Micro Controller* yang terkoneksi dengan *blynk cloud* serta internet bisa diakses lewat aplikasi *blynk* kepada *Android*. Dalam riset ini diciptakan alat otomatis yang bisa menyiram dan mengawasi tanaman hidroponik yang bisa dikendalikan dari jarak jauh memakai aplikasi *android* dan juga bisa memahami kondisi yang terjadi pada tanaman. Alasan mengapa memakai aplikasi *android* karena di masa ini pemakai *smartphone* telah banyak, dengan demikian riset ini memakai aplikasi *Android* sebagai media pengendali.

Riset yang dilakukan oleh [8], untuk memahami daya tampung tangki persediaan pertanian gandum apakah telah penuh supaya dapat dilaksanakan tahapan pengambilan serta pengiriman gandum dengan pas. Dilatarbelakangi pada waktu yang diperlukan dalam menimba gandum di tangki serta tahapan pengirimannya hanya terdiri dari 1jalan dan lahan pertaniannya sangat luas dapat mengurangi waktu dengan perlengkapan IoT. Dengan menggunakan *device mobile publisher dan subscriber* dalam memahami apakah jumlah tangki telah penuh. Suatu MQTT *broker* diletakkan pada jalan danjuga tempat persediaan gandum

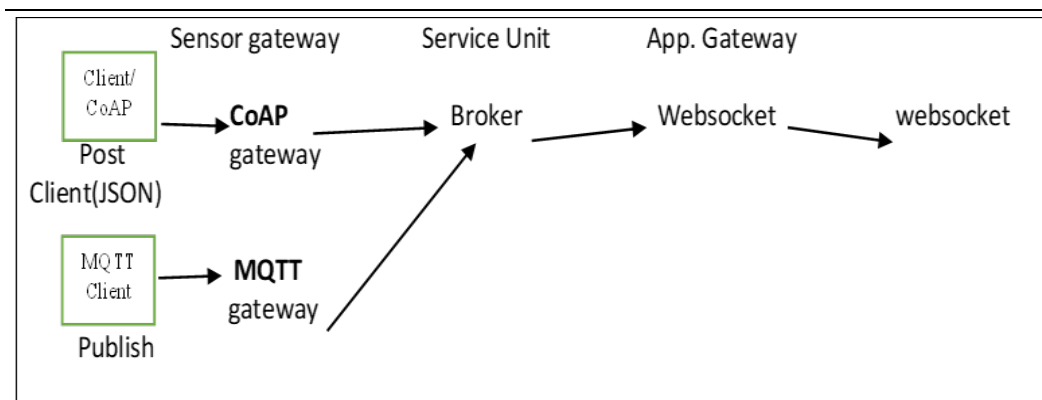
dalam berkomunikasi antar peralatan mobile dan juga tempat persediaan gandum yang telah diikuti dengan perangkat sensor. Lalu petani melakukan permintaan situasi tangki dengan melaksanakan *subscribe* di jalan besar pertanian, serta tangki hendak mengirimkan situasi daya tampung penyimpanannya dengan melaksanakan *publish*. Melalui perantara MQTT *broker* yang sudah dipasangkan dalam lahan pertanian, petani dapat menerima situasi tangki setiap waktu yang diinginkan. Bila tangki telah terisi petani dapat langsung mengambil hasil gandum kemudian langsung mengirimkannya.

Message Queue Telemetry Transport (MQTT) ialah suatu protokol yang bisa dipakai dalam melakukan teori Internet of things (IoT). MQTT mungkin pas dalam menjadi protokol IoT dikarenakan MQTT memiliki sifat *light weighted message* serta dirancang bagi *device* yang mempunyai sumber daya terbatas [9]. Dengan karakteristik yang dipunyai MQTT protokol ini bisa jadi pas supaya dilakukan dalam salah satu sistem media penyimpanan data. Pada masa ini pemakaian protokol MQTT pada IoT cenderung banyak dipakai hanya dalam pengawasan, belum banyak riset yang memakai protokol MQTT dalam melaksanakan kontrol pada perangkat-perangkat dalam jaringan IoT. Tetapi MQTT yang ialah protokol yang berada pada layer aplikasi dan dengan mekanisme protokol MQTT yakni *publish/subscribe* yang bisa mencocokkan pengiriman dan penerimaan pesan bisa dipakai untuk menerapkan kontrol serta pengawasan menyesuaikan pada kemauan pemakai, dikarenakan pengiriman dan penerimaan pesan protokol MQTT yang berlandaskan pokok-pokok yang ditetapkan [3].

Berdasarkan riset-riset yang telah dilakukan sebelumnya, pada riset ini bisa membangun sebuah rancangan pada perangkat lunak *IoT Gateway dari field ke cloud* menggunakan protokol MQTT untuk membantu mengatasi publisher pada batasan publisher yaitu dari 0 sampai 200 serta bisa menangani persoalan didalam penyimpanan data yang besar, laju pertambahan data, dan format data yang tidak beraturan pada file.

Metode Riset

Metode yang digunakan dalam riset ini adalah implementasi aplikasi sistem yang bisa menyimpan data yang telah terkirim pada *middleware*. *Middleware* akan memakai konsep *publish subscribe* untuk pergantian pesan/data. Agar bisa menerima data dari *middleware* itu, diharuskan terdapat subscriber yang pada riset ini *Internet Gateway Device* (IGD) serta data berikutnya akan disimpan dalam data storage GridFS, supaya mempermudah tahapan *get* serta *post* data, Maka dibentuk *API web service*, Dalam mengecek data yang tersimpan pada data *storage GridFS* [6]. Berikut adalah gambaran tentang perancangan sistem pada *Middleware*.



Gambar 1 Perancangan Sistem *Middleware*

Gambar 1 ialah gambar perancangan pada sistem *Middleware* yang mempunyai dua sasaran penting, yang pertama menyiapkan *gateway multi-protokol* dalam pengiriman data sensor lewat aturan CoAP dan MQTT, kedua menyiapkan *gateway* dalam mengirim data yang diterima dari *device* sensor ke *web-app* melewati protokol *websocket*. *Middleware* mempunyai 3 elemen yaitu *Application Gateway*, *Service Unit* dan *Sensor Gateway*. *Application Gateway* menyiapkan antarmuka pada aplikasi dalam tekoneksinya ke *middleware* dan membaca data yang terkirim oleh sensor lewat protokol *websocket*. *Service Unit* memberi tiga manfaat penting yaitu data management, *service delivery* serta *interface definition*. *Sensor Gateway* menyiapkan antarmuka pada *sensor* dalam mengirimkan data ke *middleware* lewat protokol CoAP atau MQTT [7].

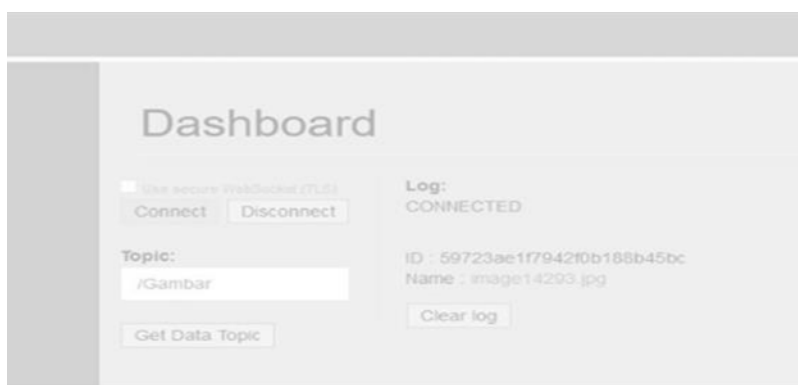
Teknik pengumpulan data melakukan perancangan riset yang membahas mengenai kebutuhan dari pada fungsional sistem itu sendiri seperti:

1. *Middleware* mempunyai tiga elemen yaitu *Application Gateway*, *Service Unit* dan *Sensor Gateway*. *Application Gateway* menyiapkan antarmuka pada aplikasi agar terkoneksi ke *middleware* serta membaca data yang dikirim sensor melewati protokol *websocket*. *Service Unit* menyiapkan tiga fungsi penting yaitu *data management*, *service delivery* serta *interface definition*. *Sensor Gateway* menyiapkan antarmuka pada *sensor* dalam mengirimkan data ke *middleware* lewat protokol CoAP atau MQTT.
2. *Node* sensor bisa mengirimkan data ke *broker* memakai protokol MQTT.
3. *Broker* bisa menerima koneksi yang diciptakan oleh *publisher* (*node* sensor) serta *subscriber*, dan juga bisa menerima data yang dikirim oleh *node* sensor. *Broker* melaksanakan tahapan *authentication* pada tiap koneksi yang hendak diciptakan oleh *publisher* dan *subscriber*, dan juga melaksanakan proses *authorization* kepada tiap data yang diterima oleh *broker* serta data yang hendak dikirim dari *broker* kepada *subscriber*. Bila lolos tahapan itu, itu artinya data dapat diteruskan kepada *subscriber*.

4. *Subscriber* bisa menerima data dari *broker* serta bisa mengelola data yang sudah didapatkan agar diinput kedalam *database* atau tidak.
5. Performansi Pada riset yang melaksanakan Analisis performansi pada protokol CoAP serta MQTT pengecekannya memakai 1 *publisher*, 1 *broker*, serta 1 *middleware*. CoAP serta MQTT mempunyai alur retransmisi yang baik dalam mengatasi *packet loss*. Hal ini yang dijadikan landasan periset dalam melaksanakan riset pada *delay* pengiriman pesan serta jumlah data pesan yang berhasil terkirim. Hingga pengetestan dilaksanakan dengan memperhitungkan *delay* dan simulasi *packet loss* yang beragam diantara 0% sampai 25 [8].

Hasil dan Pembahasan

Penerapan *website* ini dilakukan dengan menciptakan kode program bisa terkoneksi serta pembacaan data dari *data storage* lewat protokol *websocket* seturut pada segmen perencanaan. Lalu menciptakan *website* dimana tertampil dalam *browser* memakai PHP, CSS dan Javascript, seperti terlihat pada Gambar 2.



Gambar 2 Tampilan Dashboard Aplikasi Mongo DB dan GridFS

Implementasi desain bangun perangkat lunak IoT Gateway dari *field* pada *cloud* menggunakan protocol MQTT ini diimplementasikan menggunakan aplikasi MongoDB dan GridFS dengan alat-alat yang praktis dipakai serta tidak asing untuk para pemrogram. Pengetestan fungsinya dilaksanakan dalam mengecek ketersesuaian kegunaan hasil penerapan dengan rancangan. Dari hasil Analisa pengetestan fungsinya bisa diketahui apakah fungsi yang terdapat dalam sistem data storage sudah dilaksanakan secara tepat.

Pada pengetestan fungsinya dilaksanakan dalam mengecek ketersesuaian fungsi serta hasil penerapan pada perancangan yang dibuat. Bisa dilihat apakah sistem *data storage* telah berjalan dengan benar.

Tabel 1 Skenario Pengujian Fungsionalitas dan Hasil Pencapaian

No.	Kebutuhan Fungsional	Pencapaian	Hasil Pencapaian
DC_001	<i>Internet Gateway Device</i> bisa men- <i>subscribe</i> topik dari <i>middleware</i> , yaitu topik/gambar.	<ol style="list-style-type: none"> 1. <i>Middleware</i> sudah berjalan. 2. Pengguna memonitoring log pada <i>middleware</i>. 3. Pengguna menjalankan <i>Internet Gateway Device</i>. 	<i>Internet Gateway Device</i> bisa men- <i>subscribe</i> topik/gambar dari <i>middleware</i> .

```
- Client Publisher publish a message to /Gambar
- Ping from mqtt_ae2f3cae.b0ecf
- client server has connected
- Client server subscribe to /Gambar
- Client server subscribe to home/CO
- Client server has connected
```

Gambar 3 *Subscribe* Topik Gambar

Tabel 2 Pengujian Fungsionalitas dan Hasil Pencapaian *Subscribe* Topik CO

No.	Kebutuhan Fungsional	Pencapaian	Hasil Pencapaian
DC_002	<i>Internet Gateway Device</i> bisa mengambil data berdasarkan topik/CO di <i>middleware</i> .	<ol style="list-style-type: none"> 1. <i>Middleware</i> sudah berjalan. 2. Pengguna bisa menjalankan <i>Internet Gateway Device</i>. 	<i>Internet Gateway Device</i> bisa men- <i>subscribe</i> topik/gambar dari <i>middleware</i> .

```
- Client Publisher publish a message to /Gambar
- Ping from mqtt_ae2f3cae.b0ecf
- client server has connected
- Client server subscribe to /Gambar
- Client server subscribe to home/CO
- Client server has connected
```

Gambar 4 *Subscribe* Topik CO

Tabel 3 Pengujian Fungsionalitas dan Hasil Pencapaian Data Gambar

No.	Kebutuhan Fungsional	Pencapaian	Hasil Pencapaian
DC_003	<i>Internet Gateway Device</i> bisa mengirim data berdasarkan topik ke API yaitu topik/gambar.	<ol style="list-style-type: none"> 1. <i>Middleware</i> sudah berjalan. 2. Pengguna bisa menjalankan <i>Internet Gateway Device</i>. 	<i>Internet Gateway Device</i> berhasil mengiri data gambar ke API.

```
* Restarting with stat
* Debugger is active!
```

Gambar 5 Mengambil Data Gambar

Instalasi Mongo DB

Tata cara untuk menginstal *data storage* MongoDB diperlihatkan pada Gambar 6. Perintah menjalankan MongoDB Setelah *data storage* MongoDB sudah terpasang, kemudian jalankan *data storage* sebagai berikut.

1. Sudo service mongod start
2. Sudo service mongo start. Instalasi MongoDB Beserta perintahnya.

```

sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv
0C49F3730359A14518585931BC711F9BA15703C6

echo "deb [ arch=amd64 ] http://repo.mongodb.org/apt/ubuntu
precise/mongodb-org/3.4 multiverse" | sudo tee
/etc/apt/sources.list.d/mongodb-org-3.4.list

sudo apt-get update

sudo apt-get install -y mongodb-org

```

Gambar 6 Perintah Menjalankan MongoDB

Konfigurasi Pada Internet Gateway Device

Penerapan topologi jaringan dapat membicarakan konfigurasi yang di butuhkan oleh *Internet Gateway Device* dan laptop saat merancang sistem pada pengetesan sistem. Konfigurasi dapat dilaksanakan ialah menyusun alamat IP pada antarmuka *eth0* menjadi static. *Internet Gateway Device* merupakan sistem yang terkoneksi langsung dengan *middleware*, dengan demikian IGD wajib telah terhubung dengan *middleware* dengan IP 10.34.15.56, proses *subscribe* data agar dapat dijalankan, seperti yang ditunjukkan pada Kode Program 1.

Kode Program 1 Pengaturan Koneksi pada *Internet Gateway Device*

```

1. mqttc =
2. mqttc.Client("server",
3. clean_session=False)
4. mqttc.connect("10.34.15.56",1883)

```

Penerapan *Internet Gateway Device* dilakukan dalam menciptakan suatu sistem dimana bisa melaksanakan komunikasi yakni *subscribe* kepada *middleware* dalam memperoleh data sensor. Penerapan dilaksanakan saat data yang di-*subscribe* berlandaskan poin yang ada, yakni/Gambar dan home/CO, seperti yang ditunjukkan pada Kode Program 2.

Kode Program 2 Pengaturan *Subscribe* yang digunakan dalam *Middleware*

```

1. mqttc.on_message = on_message
2. mqttc.subscribe ("/Gambar")
3. mqttc.subscribe ("Home/CO")

```

Program *subscribe* topik pada *middleware* menuturkan tentang *Internet Gateway Device* dapat memperoleh pesan tiap *middleware* mengirimkan pesan dalam tiap poin yang di *subscribe* kepada *middleware* (mqtt.subscribe), seperti yang ditunjukkan pada Kode Program 3.

Kode Program 3 Pengaturan POST data yang Diterima dari *Middleware* ke API *Web Service*

```
4. def on_message (mqtt, obj , msg :
5. if msg.topic==' /Gambar':
6. headers = ("content-type": "application/json")
7. params = msg.payload
8. conn.request("post", "/api/postdata", params, headers)
9. response = conn.getresponse()
10. elif msg.topic=='home/co' :
11. headers = {"content-type": "application/json"}
12. params = msg.palyload
13. conn.request {"post", "/api/postdataco", params, headers)
14. response = conn.getresponse()
15. print response.read()
```

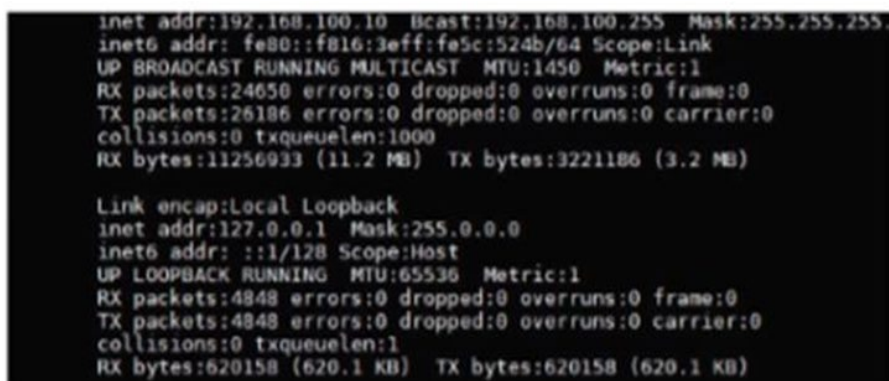
Untuk *methods* GET bentuk penerapan *web service* GET untuk data gambar ditunjukkan pada Kode Program 4.

Kode Program 4 Implementasi *Web Service* GET

```
16. @app2.route('/api/getdata/<string;name.', methods=('GET','POST'))
17. def getdata(name):
18. db = MongoClient().dataGambar
19. fs = gridFS(db)
20. with open(name, "wb" as fInput:
21. getdata=fs.find_one({"filename": name}).id
22. x=fs.get(getdata).read()
23. y=json.loads(x)
24. z=pickle.loads(y["Data"])
25. finput.write(z)
26. retrun "sukses"
```

Konfigurasi pada data storage

Data storage ini ditingkatkan di sebuah *Virtual Private Server* (VPS) pada IP eksternal 10.34.0.33. memperlihatkan pengaturan IP di dalam *data storage*. Konfigurasi *data storage* dapat dilihat pada Gambar 7.



```
inet addr:192.168.100.10 Bcast:192.168.100.255 Mask:255.255.255.
inet6 addr: fe80::f816:3eff:fe5c:524b/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1450 Metric:1
RX packets:24650 errors:0 dropped:0 overruns:0 frame:0
TX packets:26186 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:11256933 (11.2 MB) TX bytes:3221186 (3.2 MB)

Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:65536 Metric:1
RX packets:4848 errors:0 dropped:0 overruns:0 frame:0
TX packets:4848 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1
RX bytes:620158 (620.1 KB) TX bytes:620158 (620.1 KB)
```

Gambar 7 Konfigurasi Data Storage

Packet Loss

Packet loss ialah jumlah paket nan hilang dalam sebuah jaringan paket dikarenakan adanya tabrakan (*collision*), penuhnya daya tampung jaringan, serta turunnya paket nan diakibatkan habisnya TTL (*Time to Live*) pada paket. Parameter *packet loss* mendeskripsikan situasi dimana memperlihatkan banyaknya total paket yang hilang, bisa diakibatkan *collision* dengan *congestion* dalam jaringan. *Value packet loss* disesuaikan pada versi TIPHON (*Telecommunications and Internet Protocol Harmonization Over Networks*) [9] seperti pada Tabel 4.

Tabel 4 Nilai *Packet Loss* Sesuai dengan Versi TIPHON

Kategori Degradasi	<i>Packet Loss</i>	Indeks
Sangat Bagus	0%	4
Bagus	3%	3
Sedang	15%	2
Jelek	25%	1

Tahapan mengecek *packet loss* bisa memakai persamaan: $Packet\ loss = \frac{Packet\ total\ ter-capture - Packet\ terkirim}{Packet\ total\ tercapture} \times 100\%$

Pengetesan *Packet loss* mendapatkan 0%. Hal tersebut diperlihatkan pada tinjauan serta Analisis memakai *device* asistensi *wireshark*. Diperlihatkan bagian jumlah bit *frame* data yang terkirim dari satu node sama dengan banyaknya bit data yang diperoleh oleh *node* yang lain.

Delay

Delay ialah waktu yang diperlukan data dalam mencapai jarak asal ke tujuan. *Delay* bisa terpengaruhi lewat jarak, media fisik, kongesti dan juga waktu tahapan yang lama. *delay* ialah penjumlahan bermacam waktu tunda pada ujung ke ujung dalam jaringan Internet. *delay* memberi pengaruh mutu layanan (*QoS*) dikarenakan waktu tunda mengakibatkan sebuah paket makin lama sampai pada tujuan menyarankan *delay* tidak lebih besar dari 250 ms pada beberapa aplikasi, pada batas 500 ms dalam berkomunikasi multimedia supaya bisa diperoleh. Namun, pada aplikasi suara yakni *VoIP*, limit 3 *delay* maksimum ialah 300 ms. Saat memperhitungkan *delay* bisa memakai rumus berikut [9].

$$Delay\ (Sec) = \frac{\text{Waktu paket yang dikirim}}{\text{Jumlah packet}}$$

Tabel 5 Standar (QoS) Kualitas Layanan *Delay*

<i>Delay</i> (ms)	Kualitas
50 – 250	Baik
250 – 300	Cukup, masih bisa diterima
> 500 – 1000	Buruk

Pada struktur sistem yang diterapkan, pengetesan *delay* di lakukan memakai aplikasi *wireshark*. Pada *wireshark* diperlihatkan tentang time delta dimana diambil tiap tahapan pengiriman data, yakni pengiriman data pada node *publisher* ke server *broker* serta kebalikannya maupun server *broker* ke node *subscriber*. Hasil kalkulasi rata-rata memperlihatkan besaran *delay* yang diperoleh sebesar 0.039294125 *second*.

1. Pengujian tahap 1
Publisher 100 = 56,73 200 = 173,38 300 = 450,37 *Average Delay (Ms)*
2. Pengujian tahap 2
Publisher 100 = 91,81 200 = 521,92 300 = 786,104 *Average Delay (Ms)*
3. Pengujian tahap 3
Publisher 100 = 112,07 200 = 710,332 300 = 495,172 *Average Delay (Ms)*
4. Pengujian tahap 4
Publisher 100 = 87,08 200 = 168,546 300 = 243,882 *Average Delay (Ms)*

Pengujian performa

Pengujian performa dilakukan untuk mengetes kinerja dari sistem. Pengetesan berorientasi kepada *delay* yang dialami dalam melaksanakan *publish* oleh *publisher*, dan tingkat skalabilitas banyaknya *publisher* yang diatasi oleh sistem tiap detiknya. Pengetesan hendak dilakukan memakai *tool* Jmeter untuk menunjukkan banyaknya *publisher* yang dibutuhkan saat pengetesan pada sistem.

Tabel 6 Skenario Pengujian Performa pada Sistem

Kode	Parameter	Keterangan	Hasil
PP_001	<i>Delay</i> yang terjadi saat pengiriman data oleh <i>publisher</i> hingga diterima <i>subscriber</i> tanpa melakukan input pada <i>database</i> .	Menggunakan <i>tools</i> JMeter untuk melakukan pengiriman data dengan variasi jumlah <i>publisher</i> sebesar 100, 200, dan 300 <i>publisher</i> .	< 1000 Ms
PP_002	Jumlah <i>publisher</i> yang bisa ditangani tiap detik.	Menggunakan <i>tools</i> JMeter untuk melakukan pengiriman data dengan variasi jumlah <i>publisher</i> sebesar 100, 200, dan 300 <i>publisher</i> .	> 100 <i>publisher</i>

Dalam melakukan proses pengujian diperlukan *Plugin* yaitu MQTT Protokol suport yang mungkin bisa berpengaruh pada pengetesan dalam variasi *publisher plugin*. *Plugin* akan berpengaruh pada pembuatan *thread publisher* yang sudah terhubung dengan sistem. Pada pengetesan yang dilaksanakan, jumlah variasi *publisher* yang dipakai yakni 100, 200, dan 300 *publisher*. Setiap variasi dilaksanakan sebanyak tiga kali dalam memperoleh hasil yang akurat.

Dalam pengetesan sistem data dikirimkan mulai dari node *publisher* pada komputer *broker server* yang terpasang pada *Mosquitto* MQTT hendak dikirimkan

ke *subscribe* yang hendak dipergunakan bagi pemakai layanan. Pengiriman dilaksanakan setelah tahap memasukkan data pada *node publisher* dilaksanakan, tahap ini dapat di *looping* bila terdapat data kendaraan parkir yang diinput kedalam sistem. Lalu dilaksanakan perhitungan kerja untuk paket data dimana dikirim mulai dari *node publisher* ke asal *broker* serta *node subscribe* pada parameter *delay*, serta *packet loss*. Pengetesan dilaksanakan memakai *software Wireshark* agar bisa mengetahui lalu lintas data yang terdapat pada sistem.

```

tcp 0 0 103.27.250.42:1883 120.188.79.84:44188 TIME_WAIT -
tcp 0 0 103.27.250.42:1883 120.188.79.84:44189 TIME_WAIT -
tcp 0 0 103.27.250.42:1883 120.188.79.84:44185 TIME_WAIT -
tcp 0 0 103.27.250.42:1883 120.188.79.84:44149 TIME_WAIT -
tcp 0 0 103.27.250.42:1883 120.188.79.84:44171 TIME_WAIT -
tcp 0 0 103.27.250.42:1883 120.188.79.84:44189 TIME_WAIT -
tcp 0 0 103.27.250.42:1883 120.188.79.84:44193 TIME_WAIT -
tcp 0 0 103.27.250.42:1883 120.188.79.84:44197 TIME_WAIT -
tcp 0 0 103.27.250.42:1883 120.188.79.84:44219 TIME_WAIT -
tcp 0 0 103.27.250.42:1883 120.188.79.84:44215 TIME_WAIT -
tcp 0 0 103.27.250.42:1883 120.188.79.84:44208 TIME_WAIT -
tcp 0 0 103.27.250.42:1883 120.188.79.84:44143 TIME_WAIT -
tcp 0 0 103.27.250.42:1883 120.188.79.84:44180 TIME_WAIT -
tcp 0 0 103.27.250.42:1883 120.188.79.84:44145 TIME_WAIT -
tcp 0 0 103.27.250.42:1883 120.188.79.84:44169 TIME_WAIT -
tcp 0 0 103.27.250.42:1883 120.188.79.84:44175 TIME_WAIT -
tcp 0 0 103.27.250.42:1883 120.188.79.84:44179 TIME_WAIT -
tcp 0 0 103.27.250.42:1883 120.188.79.84:44181 TIME_WAIT -
tcp 0 0 103.27.250.42:1883 120.188.79.84:44174 TIME_WAIT -
tcp 0 0 103.27.250.42:1883 120.188.79.84:44167 TIME_WAIT -
tcp 0 0 103.27.250.42:1883 120.188.79.84:44155 ESTABLISHED 26885/mosquitto
tcp 0 0 103.27.250.42:1883 120.188.79.84:44181 TIME_WAIT -
tcp 0 0 103.27.250.42:1883 120.188.79.84:44161 TIME_WAIT -
tcp 0 0 103.27.250.42:1883 120.188.79.84:44190 TIME_WAIT -
tcp 0 0 103.27.250.42:1883 120.188.79.84:44168 TIME_WAIT -
tcp 0 0 103.27.250.42:1883 120.188.79.84:44177 TIME_WAIT -
tcp 0 0 103.27.250.42:1883 120.188.79.84:44213 TIME_WAIT -
tcp 0 0 103.27.250.42:1883 120.188.79.84:44188 TIME_WAIT -
tcp 0 0 103.27.250.42:1883 120.188.79.84:44192 TIME_WAIT -
tcp 0 0 103.27.250.42:1883 120.188.79.84:44184 TIME_WAIT -
tcp 0 0 103.27.250.42:1883 120.188.79.84:44194 TIME_WAIT -
tcp 0 0 103.27.250.42:1883 120.188.79.84:44152 TIME_WAIT -
tcp 0 0 103.27.250.42:1883 120.188.79.84:44202 TIME_WAIT -
tcp 0 0 103.27.250.42:1883 120.188.79.84:44176 TIME_WAIT -
tcp 0 0 103.27.250.42:1883 120.188.79.84:44203 TIME_WAIT -
tcp 0 0 103.27.250.42:1883 120.188.79.84:44214 TIME_WAIT -

```

Gambar 8 Server Performansi Broker MQTT Mosquito

Berlandaskan hasil skenario pengetesan dalam Tabel 3 waktu *delay* yang dialami sebanding dengan banyaknya *publisher* yang melaksanakan *publish* data. Waktu *delay* yang dialami pada pengiriman data pada *publisher* sejumlah 100 sertas 200 *publisher*, mempunyai waktu *delay* di bawah 1000 ms. Sementara dalam pengetesan dengan banyaknya *publisher* sejumlah 300, memperlihatkan waktu *delay* di atas 1000 ms, hingga hasil itu bisa dibilang lebih buruk bila dikomparasi pada hasil dengan banyaknya *publisher* di bawah 300 *publisher*. Dari pemaparan diatas bisa diambil kesimpulan, sistem bisa mengatasi banyaknya *publisher* sejumlah < 300 *publisher* pada satu waktu.

Tabel 7 Hasil Pengujian Waktu *Delay*

<i>Publisher</i>	<i>Average Delay (ms)</i>	<i>Publisher</i>	<i>Average Delay (ms)</i>	<i>Publisher</i>	<i>Average Delay (ms)</i>
100	56,73	200	173,38	300	450,37
100	91,81	200	521,92	300	786,104
100	112,7	200	710,332	300	495,172
100	87,08	200	168,544	300	243,882

Pengetesan kinerja pertama yakni pengetesan *delay* yang dialami pada pengiriman data oleh *publisher* sehingga diperoleh *subscriber* dengan tidak melaksanakan masukkan pada *database* PP_001. Pengetesan dilaksanakan dengan memakai variasi *publisher* sejumlah 100, 200, serta 300 *publisher* yang mengakses pada waktu yang sama.

Tabel 8 Hasil Pengujian Skalabilitas

<i>Publisher</i>	<i>Responded</i>	<i>Success Rate %</i>	<i>Responded < 1 s</i>
100	100	100%	100
100	100	100%	100
100	100	100%	100
100	100	100%	100
200	200	100%	200
200	200	100%	56
200	200	100%	67
200	200	100%	91
300	300	100%	251
300	293	99%	1
300	295	99%	29
300	296	99%	27

Dari Tabel 8 bisa diketahui hasil pengetesan yang sudah dilaksanakan PP_002, memperlihatkan yakni sistem memiliki kemampuan dalam mengatasi banyaknya *publisher* hingga sejumlah 300 *publisher*, serta *responded* sebesar 296 dengan taraf keberhasilan 99%. Selain itu, dari hasil pengetesan memperlihatkan, dengan pengetesan sejumlah 300 *publisher*, rata-rata banyaknya *publisher* yang ditangani di bawah 1 detik sejumlah 27 *publisher*.

Taraf skalabilitas yang terdapat pada sistem dalam mengatasi banyaknya *publisher* di dalam 1 detik, rata-rata *publisher* yang ditangani pada waktu satu detik mendapat 91 hingga 27 *publisher* pada ragam banyaknya *publisher* sejumlah 300 dengan taraf keberhasilan 99% dalam dua atau tiga kali percobaan dalam waktu yang sama. Secara umum dijelaskan pada Tabel 3 dimana waktu *delay* menunjukkan di atas < 1000 ms maka hasil ini dibidang lebih buruk bila dibandingkan dengan banyaknya *publisher* di bawah < 300.

Tabel 9 Hasil Pengujian Fungsional pada Sistem *Data Storage*

Kode	Fungsi	Status
DC_001	<i>Internet Gateway Device</i> (IGD) bisa men- <i>subscribe</i> topik dari <i>middleware</i> , yaitu topik/gambar.	Valid
DC_002	<i>Internet Gateway Device</i> (IGD) bisa mengambil data berdasarkan topik pada <i>middleware</i> , yaitu topik/gambar.	Valid
DC_003	<i>Internet Gateway Device</i> (IGD) bisa mengirim data berdasarkan topik ke API yaitu topik/gambar, home/CO.	Valid
DC_004	API <i>web service</i> bisa menerima data dari <i>Internet Gateway Device</i>	Valid
DC_005	API <i>web service</i> bisa mengirim data dari <i>Internet Gateway Device</i> ke GridFS.	Valid
DC_006	API <i>web service</i> bisa menyimpan data dari <i>Internet Gateway Device</i> ke GridFS.	Valid
DC_007	GridFS bisa menyimpan data pada sensor.	Valid
DC_008	GridFS bisa menghapus data pada sensor.	Valid
DC_009	IoT App bisa menerima permintaan menampilkan data berdasarkan topik, topik/gambar, home/CO.	Valid

Pengetesan fungsional dilaksanakan dalam mengetahui ketersediaan fungsi-fungsi hasil penerapan dan juga perancangan. Berdasarkan hasil analisa pengetesan fungsionalitas bisa diketahui apa fungsi yang terdapat pada sistem data storage sudah terlaksana dengan tepat

Simpulan

Berdasarkan riset dan pengujian yang dilakukan maka bisa disimpulkan bahwa penyimpanan pada media data menggunakan protokol *Message Queuing Telemetry Transport* (MQTT) bisa meminimalisir penyimpanan data yang besar sehingga bisa di atasi dengan baik, oleh karena itu dengan adanya *middleware* yang di bagi menjadi dua file yaitu file metadata dan *chunks* proses penyimpanan dan pengaksesan ke dalam *data storage* menjadi semakin cepat dan optimal secara keseluruhan.

Saran dari riset ini untuk harapan ke depannya ialah data yang ada tidak bisa disimpan serta harus mengulang memasukkan data dari awal. Hal ini dibutuhkan sebab data yang dikirimkan masih menggunakan program *wireshark* dengan kode program yang masih menyesuaikan kebutuhan dari sistem.

Daftar Pustaka

- [1] O. B. Pratama, A. Bhawiyuga, and K. Amron, "Pengembangan Perangkat Lunak IoT Cloud Platform Berbasis Protokol Komunikasi HTTP," *J. Pengemb. Teknol. Inf. dan Ilmu Komput. e-ISSN*, vol. 2548, no. 9, p. 964X, 2018.
- [2] H. F. Handi and G. E. Setyawan, "Sistem Pemantauan Menggunakan Blynk dan Pengendalian Penyiraman Tanaman Jamur Dengan Metode Logika Fuzzy," *J. Pengemb. Teknol. Inf. dan Ilmu Komput. e-ISSN*, vol. 2548, p. 964X, 2019.
- [3] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *J. internet Serv. Appl.*, e-ISSN, vol. 1, no. 1, pp. 7–18, 2010.
- [4] M. F. Rozi, E. S. Pramukantoro, and K. Amron, "Analisis Performansi dan Skalabilitas pada Event-Based IoT Middleware," *J. Pengemb. Teknol. Inf. dan Ilmu Komput. e-ISSN*, vol. 2548, p. 964X, 2017.
- [5] S. A. Budianto, A. Bhawiyuga, and D. P. Kartikasari, "Penerapan Perangkat Mobile Publisher Subscriber Sebagai Perantara Pengiriman Data Sensor Dari Lapangan Ke Pusat Data," *J. Pengemb. Teknol. Inf. dan Ilmu Komput. e-ISSN*, vol. 2548, p. 964X, 2018.
- [6] G. Arganata, E. S. Pramukantoro, and W. Yahya, "Pengembangan Sistem Penyimpanan Data Berbasis MongoDB dan GridFS Untuk Menyimpan Data Yang Beragam Dari Node Sensor," *J. Pengemb. Teknol. Inf. dan Ilmu Komput. e-ISSN*, vol. 2548, p. 964X, 2017.
- [7] H. Anwari, E. S. Pramukantoro, and M. H. Hanafi, "Pengembangan Iot Middleware Berbasis Event-Based dengan Protokol Komunikasi CoAP, MQTT dan Websocket," *J. Pengemb. Teknol. Inf. dan Ilmu Komput. e-ISSN*, vol. 2548, p. 964X, 2017.

-
- [8] D. Thangavel, X. Ma, A. Valera, H.-X. Tan, and C. K.-Y. Tan, "Performance evaluation of MQTT and CoAP via a common middleware," in *2014 IEEE ninth international conference on intelligent sensors, sensor networks and information processing (ISSNIP)*, 2014, pp. 1–6.
- [9] G. Y. Saputra, A. D. Afrizal, F. K. R. Mahfud, F. A. Pribadi, and F. J. Pamungkas, "Penerapan protokol MQTT pada teknologi WAN (studi kasus sistem parkir Univeristas Brawijaya)," vol. Vol. 12, N, no. Jurnal Informatika Mulawarman Vol. 12, No. 2 September 2017 69 *e-ISSN* 2597-4963 dan *p-ISSN* 1858-4853, 2017.
- [10] M. Safii and V. Vidy, "Perancangan Bangun Alat Monitoring Notifikasi Tegangan Genset Berbasis Internet of Things Dan Sms Gateway," *Sebatik*, vol. 23, no. 1, pp. 178–184, 2019, doi: 10.46984/sebatik.v23i1.466.
- [11] R. Rizky, Z. Hakim, A. M. Yunita, and N. N. Wardah, "PENERAPAN TEKNOLOGI IOT (INTERNET OF THINK) PADA RUMAH PINTAR BERBASIS MIKROKONTROLER ESP 8266," vol. 4, no. 2, pp. 278–281, 2020.
- [12] A. B. Pulungan and M. Oktavianda, "PARKING INFORMATION SYSTEM BASED ON INTENET OF THINGS (IOT) No Jarak sebenarnya Jarak pengukuran pada sensor ultrasonik (Cm) Tegan gan," vol. 13, no. 352, 2020.