LAPORAN JOBSHEET 6

**TRANSMISI DATA MENGGUNAKAN MESSAGE  
QUEUING TELEMETRY TRANSPORT (MQTT)  
PROTOCOL**

SISTEM EMBEDDED

Disusun untuk memenuhi tugas individu mata kuliah

Sistem Embedded Tahun Akademik 2023



Disusun oleh :

**Baiq Julita Noor Fatimah 4.31.21.0.07/TE3A**

**JURUSAN TEKNIK ELEKTRO**

**PROGRAM STUDI S.Tr TEKNIK TELEKOMUNIKASI**

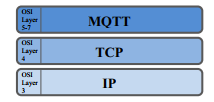
**POLITEKNIK NEGERI SEMARANG**

**2023**

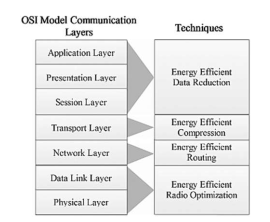
1. **KETERANGAN SINGKAT**

*Message Queuing Telemetry Trasnsport* (MQTT) adalah protokol komunikasi  
yang berjalan di atas *stack* TCP/IP, didesain untuk komunikasi *Machine-toMachine* (M2M), bersifat *open sources* dan *lightweight,* mempunyai *protocol  
overhead* yang rendah (minimum 2 bytes) sehingga berefek pada konsumsi daya  
yang kecil dan mampu bekerja pada *latency* yang tinggi serta *bandwidth* yang  
kecil (Ullas *et al.*, 2014).

MQTT bekerja di atas TCP/IP *stack* di level *application layer* pada standar  
sistem OSI (Pal, Ghosh and Bhattacharya, 2017). Protokol ini menerapkan teknik  
kompresi data dan data *reduction* untuk melakukan efisiensi energi (Ali, Shah and  
Arshad, 2016). Hal tersebut dapat dilihat pada Gambar 5.1 dan Gambar 5.2.

****

Gambar 5.1. OSI *Layer* level protokol MQTT

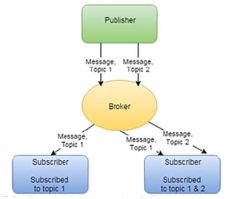
****

Gambar 5.2. Teknik efisiensi energi pada OSI *Layer*(Sumber : Ali, Shah and Arshad, 2016)

Menurut pendapat Pal, Ghosh dan Bhattacharya (2017) MQTT merupakan  
jenis protokol *data-agnostic* yang mana data yang mampu dikirimkan tidak  
terbatas pada jenis data tertentu, namun semua jenis data dapat dikirimkan seperti  
data *binary*, *text*, XML maupun JSON.

Protokol MQTT menggunakan model *publish/subscribe* untuk komunikasi data  
(Krishna *et al.*, 2017). Publisher mempunyai peran sebagai pengirim data atau  
*sender*, sedangkan *subscriber* berperan sebagai penerima dara atau *receivers* dan  
terdapat broker yang berperan sebagai *middleman* yang bertugas meneruskan  
pesan (*message*) dari *publisher* menuju *subscriber*. Pesan di dalam broker  
dibedakan dan difilter menggunakan *topic* yang mana setiap pesan yang  
dikirimkan ke broker diberi *tag* atau penanda oleh *publisher* untuk membedakan  
setiap data (Pal, Ghosh and Bhattacharya, 2017).

Metode *publish/sibscribe* mempunyai kentungan yaitu antara *publisher* dan  
*subscriber* bersifat *loosely coupled,* artinya mereka mampu tetap bekerja tanpa  
saling bergantung satu sama lain atau tidak saling terikat terutama pada tempat,  
waktu dan sinkronisasi. Pal, Ghosh dan Bhattacharya (2017) dalam penelitiannya  
menyebutkan bahwa *Publisher* dan *subscriber* tetap akan bekerja walaupun  
mereka tidak saling mengetahui letak atau keberadaan satu sama lain karena  
adanya *broker* diantara *publisher* dan *subscriber* atau sering disebut dengan *space  
decoupling*.  
*Time decoupling* merupakan keuntungan yang lainnya dari metode ini, yang  
mana antara *publisher* dan *subscriber* tidak harus aktif atau terkoneksi bersamaan.  
*Publisher* tidak akan melakukan *blocking* ketika memproduksi sebuah *event* baru,  
misalnya *subscriber* bisa saja *disconnect* setelah melakukan *subscribe* ke *broker*,  
beberapa saat kemudian *subscriber connect* kembali ke *broker*, maka *subscriber*akan tetap menerima data yang sebelumnya tertunda (*pending*) (Pal, Ghosh and  
Bhattacharya, 2017). Metode tersebut disebut juga *mode offline*.  
*Synchronization decoupling* adalah kondisi di mana pengaturan *event* baik itu  
penerimaan atau pengiriman pesan di sebuah *node* hingga tidak saling  
mengganggu satu sama lain (Alhakbani, Hassan and Ykhlef, 2017). Metode kerja  
*publish/subscribe* yang digunakan protokol MQTT dapat dilihat pada Gambar 5.3.

****

Gambar 5.3. Cara kerja *publish/subscribe*

Khrishna *et al* (2017) dalam penelitiannya menyebutkan bahwa metode  
*publish/subscribe* seperti yang ditunjukkan Gambar 2.4 sangat kontras jika  
dibandingkan dengan metode komunikasi data menggunakan *client/server*tradisional yang mana antara pengirim dan penerima saling bergantung satu sama  
lain (*tightly coupled*). Server tidak dapat mengirim data jika *client* tidak aktif  
secara bersamaan atau sedang tidak melakukan *listening* pada suatu *topic* tertentu.  
Metode tersebut mulai ditinggalkan karena sudah tidak relevan untuk konsep  
komunikasi IoT yang menuntut data dikirim secara *realtime* dan akurat walaupun  
*sender* dan *receiver* berada di daerah dengan koneksi yang *intermittent* atau  
*unreliable*.

Keuntungan yang lain dari penggunaan metode *publish/subscribe* yaitu metode  
ini bersifat *scalable*, artinya kita dapat menambahkan broker lain ke dalam sistem  
IoT untuk mengurangi beban pada broker tunggal dengan cara membagi *topic* di  
antara kedua broker tersebut (Krishna *et al.*, 2017).

Kekurangan dari penggunaan metode ini yaitu meskipun bersifat *loosely  
coupled* terkait dengan waktu, tempat dan sinkronisasi, namun di sisi pengiriman  
data metode ini bersifat *tightly coupled*. Menurut Khrishna *et al* (2017), jika  
terdapat perubahan yang dilakukan pada arah pengiriman data, misalnya  
perubahan *channel* dimana data akan disimpan, maka semua *subscriber* harus  
dimodifikasi agar dapat melakukan *subscribe* pada *topic* terbaru.  
Berdasarkan tujuan protokol MQTT agar mudah untuk diimplementasikan,  
terdapat beberapa sinyal kontrol yang didefinisikan oleh MQTT untuk  
berinteraksi, namun hanya ada enam sinyal kontrol yang menjadi bagian  
terpenting yang dipakai oleh *client*. Keenam sinyal kontrol tersebut adalah  
sebagai berikut.

*1.Connect*Sinyal kontrol ini adalah pesan pertama yang harus merupakan paket *CONNECT*yang mana mengidentifikasikan informasi tentang *client* dan *server* setelah  
koneksi jaringan antara *client* dan *server* terbentuk.

*2. Keep Alive*

*Keep alive* merupakan sebuah spesifikasi berapa lama sebuah *client* dapat aktif  
meskipun tanpa melakukan pengiriman sebuah data sebelum pada akhirnya  
melakukan *disconnecting*. Sinyal kontrol ini diatur pada saat *connect*.

*3. Disconnect*

*Disconnect* merupakan proses menunggu *client* untuk menyelesaikan  
pekerjaannya melakukan *subscribe*. Proses ini juga bagian dari TCP *session* untuk  
melakukan *terminate*.

*4. Subscribe*

*Subscribe* adalah proses mengirim sebuah pesan *SUBSCRIBE* dari *client* ke *server*untuk membuat sebuah proses *subscribe* pada satu atau beberapa *topic*.

*5. Unsubscribe*

*Unsubscribe* merupakan proses mengirim sebuah pesan *UNSUBSCRIBE* dari  
*client* ke *server* untuk membuat sebuah proses *unsubscribe* pada satu atau  
beberapa *topic*.

*6. Publish*

*Publish* adalah proses mengirim sebuah pesan *PUBLISH* dari *client* ke *server* atau  
dari *server* ke *client* untuk mengangkut sebuah pesan atau data.  
MQTT mengizinkan setiap *user* untuk menggunakan *Quality of Service* (QoS)  
yang berbeda sesuai dengan kebutuhannya. Protokol ini mempunyai tiga level  
QoS yang mendefinisikan tiga level berbeda dalam menjamin atau memberikan  
garansi terhadap konsistensi pengiriman pesan. *Subscriber* dan *broker*menyediakan mekanisme penyimpanan dan pengiriman kembali pesan, sehingga  
hal ini meningkatkan konsistensi data akibat kegagalan jaringan, *restart* dari  
aplikasi maupun penyebab lainnya (Fathia and Said, 2017). Tingkatan QoS yang  
diterapkan oleh protokol MQTT adalah sebagai berikut.

1. QoS 0

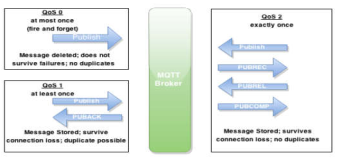
Menurut Fathia dan Said (2017), QoS 0 dapat melakukan pengiriman pesan ke  
*broker* hanya sekali (*at most once / fire and forget*). Pesan yang terkirim  
tergantung dari *reliability stack* TCP atau tergantung dari *realibility* jaringan.  
Apabila terdapat pesan yang mengalami kegagalan dalam proses transmisi, maka  
*broker* tidak akan meminta untuk mengirimkan pesan kembali.

2. Qos 1

QoS 1 melakukan pengiriman pesan ke *broker* setidaknya satu kali (*at least  
one*) dengan pemberian sinyal kontrol ACK (*acknowlegment*) pada setiap pesan  
yang dikirimkan. Pesan yang dikirimkan dijamin akan sampai ke broker, namun  
setiap pesan tersebut tidak dijamin bersih dari adanya duplikasi (Fathia and Said,  
2017).

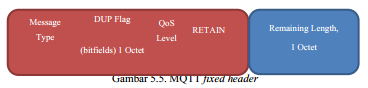
3. QoS 2

Fathia dan Said (2017) mengatakan bahwa QoS 2 merupakan tingkatan  
tertinggi dari QoS MQTT. Pesan yang dikirimkan diterima satu kali (*exactly one*)  
dengan garansi semua pesan yang dikirim pasti diterima dan tidak ada pesan yang  
terduplikasi. MQTT menggunakan ID pada setiap pesan untuk melakukan *filter*agar tidak ada pesan yang terduplikasi. Hal itu dapat dilihat pada Gambar 5.4.

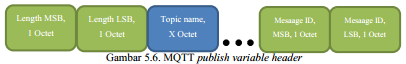
****

Gambar 5.4. Proses komunikasi setiap level QoS protokol MQTT  
(Sumber : <http://www.ossmentor.com/2015/04/internet-of-things-mqtt.html>)

Menurut Pal, Ghosh dan Bhattacharya (2017), pesan di dalam protokol MQTT  
mempunyai dua bagian yaitu *fixed part* dan *variable part*. *Fixed part* merupakan  
bagian penting untuk semua pesan, sedangkan *variable part* merupakan bagian  
yang dapat berubah tergantung dari pesan yang akan dikirim.  
*Fixed part* terdiri dari dua *bytes* yaitu untuk *field* yang pertama terdiri dari jenis  
pesan (*type message*) akan terkirim, DUP Flag , level QoS yang digunakan dan  
*Retain*. Sedangkan untuk *field* yang kedua terdiri dari *Remaining Length* yang  
mengindikasikan jumlah oktet tersisa dari pesan yang mana termasuk dari variabel  
*header* dan *payload* (Pal, Ghosh and Bhattacharya, 2017). Bagian dari *fixed  
header* dapat dilihat pada Gambar 5.5.

****

Pal, Ghosh dan Bhattacharya (2017) berpendapat bahwa *Variable part* atau  
*variable header* merupakan bagian yang ukurannya dapat diubah. Bagian ini  
digunakan untuk keperluan *publish message* yang di dalamnya terdapat nama  
*topic* sebagai ID pesan. *Length field* pada *header* terdiri dari dua *bytes* dan mengindikasikan nomor *octet* yang diperlukan untuk merepresentasikan nama  
*topi*c. Hal tersebut dapat dilihat pada Gambar 5.6.



1. **ALAT DAN BAHAN**
2. ESP32
3. Breadboard
4. Kabel jumper
5. Potensiometer
6. Sensor DHT11
7. LED
8. Multimeter
9. Resistor 1K Ohm
10. **HASIL PEMBAHASAN**
11. **KESIMPULAN**

Kesimpulan dari praktikum mengenai transmisi data menggunakan protokol Message Queuing Telemetry Transport (MQTT) adalah sebagai berikut:

1. MQTT adalah protokol komunikasi ringan yang dirancang untuk mentransmisikan data dalam bentuk pesan antara perangkat-perangkat yang terhubung ke jaringan. Ini sangat berguna dalam pengembangan aplikasi IoT.
2. Dalam praktikum ini, kami telah memahami konsep dasar MQTT, termasuk penggunaan broker MQTT, topik (topic), penerbit (publisher), dan pelanggan (subscriber).
3. Kami telah memahami bagaimana mengkonfigurasi perangkat (dalam hal ini, mungkin menggunakan bahasa pemrograman seperti Python, Node.js, atau perangkat keras seperti ESP32) sebagai penerbit atau pelanggan MQTT.
4. Kami telah mempelajari cara membuat pesan, mengirimkannya melalui broker MQTT, dan menerima pesan tersebut di pelanggan lain yang berlangganan topik yang sama.
5. MQTT mendukung banyak skenario, seperti pengiriman data sensor, pengendalian perangkat jarak jauh, dan komunikasi antara perangkat dalam jaringan IoT.

Kesimpulannya, praktikum ini memberikan pemahaman yang kuat tentang cara menggunakan MQTT untuk mentransmisikan data antara perangkat yang terhubung ke jaringan. MQTT adalah alat yang sangat berguna untuk pengembangan aplikasi IoT dan pemantauan jarak jauh, dan pemahaman tentang konsep MQTT sangat berharga dalam konteks ini.