

**MODUL EKRIPI-DEKRIPI, SERVER, DAN PENJADWALAN
ARMADA PADA FMCS**

TUGAS AKHIR

Oleh

SHAH DEHAN LAZUARDI

NIM: 13213111



**PROGRAM STUDI TEKNIK ELEKTRO
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2017**

**MODUL EKRIPI-DEKRIPI, SERVER, DAN PENJADWALAN
ARMADA PADA FMCS**

Oleh:

SHAH DEHAN LAZUARDI

Tugas Akhir ini telah diterima dan disahkan sebagai persyaratan untuk
memperoleh gelar

SARJANA TEKNIK

di

**PROGRAM STUDI TEKNIK ELEKTRO
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG**

Bandung, 7 Mei 2017

Disetujui oleh:

Pembimbing I,

Pembimbing II,

Pembimbing III,

Ir. Arief Syaichu Rohman,
MEngSc, Ph.D

Arif Sasongko, S.T.,
M.Sc,Ph.D

Ir. Agung Darmawan

ABSTRAK

PENJADWALAN ARMADA MENGGUNAKAN FMCS

Oleh

Shah Dehan Lazuardi

NIM: 13213111

PROGRAM STUDI TEKNIK ELEKTRO

Belum adanya jadwal dan rendahnya kenyamanan yang tetap masih menjadi masalah utama pada angkutan umum seperti Angkutan kota dan bis. Angkutan kota dan bis beroperasi pada jadwal yang tidak tetap bergantung pada kondisi lalu lintas saat itu. Masalah tersebut dapat diselesaikan menggunakan sistem penjadwalan pada Fleet Monitoring & Controlling System dan Guided Bus. Sistem penjadwalan armada akan menjaga tiap armada agar sesuai jadwalnya masing masing. Sistem penjadwalan ini membutuhkan input berupa posisi tiap armada dan output berupa kecepatan tiap fleet agar sesuai dengan jadwalnya. Data yang dikirimkan dari tiap armada akan di enkripsi terlebih dahulu lalu dikirimkan ke server menggunakan protokol MQTT setelah itu data tersebut didekripsi pada GUI dan di parsing sehingga data posisi dapat diolah pada algortima penjadwalan. Pada hasil pengujian sistem ini data berhasil dienkripsi lalu dikirmkan pada server setelah itu di dekripsi pada GUI dan algortima penjadwalan dapat mejaga fleet sesuai dengan jadwalanya.

Kata kunci - dekripsi, enkripsi, FMCS, MQTT, penjadwalan.

ABSTRACT
FLEET SCHEDULLING USING FMCS

By

Shah Dehan Lazuardi

NIM: 13213111

DEPARTMENT OF ELECTRICAL ENGINEERING

Scheduling is become main problem of public transportation and bus. Public transportation and bus operate in unfixed schedule depends on traffic condition. Scheduling problem can be solved by using scheduling system in Fleet Monitoring & Controlling System for Guided Bus. Scheduling system keeps each fleet on their schedule. Scheduling system needs position as input and calculate the speed of each fleet so they will stick to their schedule. The encrypted Data will be sent to the server then from each fleet by using MQTT protocols then the encrypted Data will be decrypted in GUI and parsed so the position data can be calculated by scheduling algorithm. In the system testing, encrypted data can be sent to server then those data can be decrypted in GUI and scheduling algorithm can keep fleet on their schedule.

Keywords - decrypt, encrypt, FMCS, MQTT, Scheduling.

KATA PENGANTAR

Puji dan syukur penulis panjatkan kehadirat Allah SWT yang telah melimpahkan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan tugas akhir yang berjudul “Fleet Monitoring and Control System” tepat pada waktunya. Buku ini adalah laporan pengerjaan tugas akhir yang dilakukan pada tahun ajaran 2016/2017. Buku tugas akhir ini dibuat sebagai upaya pemenuhan syarat kelulusan program sarjana Program Studi Teknik Elektro di Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung.

Pelaksanaan tugas akhir dan penyelesaian buku ini tentu tidak akan dapat terlaksana dengan baik tanpa adanya dukungan dari pihak-pihak di sekitar penulis. Oleh karena itu, pada kesempatan kali ini penulis ingin mengucapkan terima kasih kepada pihak-pihak yang telah membantu terlaksananya kerja praktik ini, antara lain:

- Ayah, Ibu, dan Adik yang telah memberikan dukungan yang sangat besar kepada penulis selama menuntut ilmu di ITB;
- Bapak Ir. Arief Syaichu Rohman, MEngSc, Ph.D., Arif Sasongko, ST, M.Sc, Ph.D., dan Agung Darmawan selaku dosen pembimbing penulis yang telah memberikan bimbingan dan nasihatnya kepada penulis;
- Bapak Arif Sasongko, S.T., M.Sc.,Ph.D., selaku ketua program studi Teknik Elektro ITB beserta Tim Tugas Akhir Teknik Elektro ITB yang telah memberikan arahan selama pengerjaan tugas akhir;
- Aulia Hening Darmasti dan Ali Zaenal Abidin selaku rekan satu kelompok pengerjaan tugas akhir yang telah bersedia bekerjasama dengan penulis selama kegiatan tugas akhir ini dilakukan;
- Teman-teman di ruang riset mandiri, yaitu Audi, Irham, Fadhil, Pakde, Iki, Tandut, Ducun, Ikarus, Santok, Eki, Juhan, Vito, Binal, Andin, Rambo, Fadel, Abi, Azmi, Ipeh, Fitra, Dian, Adin, Nanur, Febris, Jusri dan Ridhan. yang telah berjuang bersama melaksanakan kegiatan tugas akhir selama ini;
- Teman-teman di ruang tugas akhir lainnya yang sama-sama saling membantu saat penyusunan tugas akhir;

- Teman-teman di Teknik Elektro ITB yang telah menempuh kegiatan perkuliahan bersama-sama hingga saat ini;
- Astarina N. dan Gerry Almanda Y. yang telah membantu dalam pembuatan video tugas akhir.
- Uwi yang terus memberikan semangat dan nasihat selama mengerjakan tugas akhir ini.
- Semua pihak yang tidak bisa disebutkan satu-satu yang telah membantu kami dalam penyelesaian laporan ini.

Pada penulisan buku ini, penulis menyadari bahwa buku tugas akhir ini masih memiliki banyak kekurangan. Oleh karena itu, penulis mengharapkan kesediaan pihak-pihak yang terkait untuk memberi kritik dan saran yang bersifat membangun agar dapat lebih baik di kesempatan yang akan datang. Semoga buku ini dapat bermanfaat bagi penulis sendiri khususnya maupun pembaca pada umumnya.

Bandung, 7 Mei 2017

Penulis

DAFTAR ISI

ABSTRAK	ii
ABSTRACT	iii
KATA PENGANTAR.....	iv
DAFTAR ISI.....	vi
BAB I PENDAHULUAN.....	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Tujuan Tugas Akhir.....	2
1.4 Lingkup Permasalahan	2
1.5 Metodologi	2
1.6 Sistematika Penulisan.....	3
BAB II MQTT dan AES	5
2.1 Message Queuing Telemetry Transport (MQTT)	5
2.2 Advanced Encryption Standard (AES).....	6
BAB III SPESIFIKASI DAN PERANCANGAN	8
3.1 FMCS	8
3.2 Spesifikasi Subsistem	8
3.3 Perancangan Subsistem	9
BAB IV IMPLEMENTASI DAN HASIL PENGUJIAN.....	21
4.1 Implementasi	21
4.2 Prosedur dan Hasil Pengujian.....	27
BAB V KESIMPULAN	34
5. 1. Kesimpulan.....	34
5. 2. Saran	34
DAFTAR PUSTAKA	35

BAB I

PENDAHULUAN

1.1 Latar Belakang

Indonesia adalah salah satu negara dengan jumlah penduduk terpadat keempat di dunia, setelah Cina, India, dan Amerika Serikat. Jumlah penduduk Indonesia, 257 juta orang, berbanding lurus dengan penggunaan kendaraan pribadi maupun umum. Penggunaan kendaraan pribadi mengalami peningkatan tiap tahunnya. Dari data yang dirilis Badan Pusat Statistik (BPS) Provinsi DKI Jakarta Tahun 2015 terkait dengan statistik transportasi, jumlah kendaraan yang melintas di Provinsi DKI Jakarta terus mengalami peningkatan pada periode 2010-2014, dengan rata-rata peningkatan sebesar 9,93% per tahun. Penyumbang tertinggi peningkatan persentase tersebut adalah kendaraan sepeda motor dengan rata-rata peningkatan per tahun sebesar 10,54%, diikuti dengan peningkatan persentase mobil penumpang yaitu 8,75%. Berbeda jauh dengan pertumbuhan jumlah angkutan umum yang pada tahun 2013-2014 hanya mengalami peningkatan sebesar 1,74%.

Pelayanan yang diberikan oleh transportasi umum kurang memuaskan membuat pengguna transportasi umum beralih pada transportasi pribadi. Angkutan kota, bus dan bajaj memiliki daya tarik yang lemah untuk menggerakkan masyarakat agar berpindah moda transportasi. Khususnya bus dan angkutan kota yang menunggu penumpang penuh terkadang memakan waktu yang lama. Tidak jarang bus dan angkutan kota melanggar rambu-rambu lalu lintas seperti berhenti di tempat yang dilarang berhenti. Hal ini diperparah dengan belum baiknya penataan kota dan sistem lalu lintas membuat transportasi umum tidak nyaman untuk digunakan karena tidak memiliki jadwal yang tetap.

Dari masalah masalah tersebut masyarakat dibutuhkan suatu sistem transportasi yang memiliki penjadwalan tetap, sistem tersebut dapat memantau dan memberikan perintah pada armada agar tepat pada jadwalnya sehingga masyarakat dapat menggunakan transportasi umum dengan nyaman tanpa waktunya terbuang sia sia.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan pada bagian sebelumnya, maka rumusan masalah adalah sebagai berikut:

- Monitoring setiap armada.
- Server sebagai media komunikasi antara control station dan armada.
- Keamanan informasi antara armada dan server sehingga perintah terjamin berasal dari control station.

1.3 Tujuan Tugas Akhir

Dari pemaparan rumusan masalah maka tujuan tugas akhir ini adalah:

- Perancangan dan implementasi algoritma penjadwalan yang dapat menjaga armada tepat pada jadwalanya.
- Perancangan dan implementasi server untuk bertukar informasi antara armada dan *control station*.
- Perancangan dan implementasi keamanan informasi.

1.4 Lingkup Permasalahan

Dengan melihat besarnya masalah pada sistem penjadwalan tersebut maka diperlukan pembatasan masalah pada tugas akhir ini. Pembatasan masalah dilakukan dengan sistem penjadwalan diperuntukan bagi Guided Bus yang bebas macet dan pengemudi Guided Bus yang selalu patuh terhadap perintah control station. Pembatasan masalah akan diuraikan lebih lengkap pada bagian spesifikasi produk.

1.5 Metodologi

Dalam melakukan perancangan dan implementasi tugas akhir ini penulis merangkum langkah-langkah tersebut dalam susunan metodologi, yaitu:

1. Penentuan Topik

Penentuan topik dilakukan dengan berdiskusi dalam tim dengan mempertimbangkan kesukaran masalah serta keahlian anggota tim

2. Tinjauan Pustaka

Pada langkah ini dilakukan pencarian solusi umum dari masalah-masalah dan mengkaji solusi tersebut. Informasi tinjauan pustaka berasal dari internet maupun dari *paper* yang berhubungan dengan masalah tugas akhir ini.

3. Penentuan Spesifikasi

Pada tahap ini solusi yang telah dipilih diturunkan menjadi spesifikasi kuantitatif maupun kualitatif.

4. Perancangan

Pada langkah ini dilakukan perancangan implementasi dari solusi agar memenuhi spesifikasi yang sebelumnya ditentukan.

5. Implementasi Sistem

Pada langkah ini rancangan yang telah dirancang diimplementasikan sesuai dengan rancangan awal.

6. Pengujian

Langkah ini berisi tentang serangkaian uji coba pada produk untuk mengetahui ketercapaian produk terhadap spesifikasi yang sebelumnya telah ditentukan

7. Simpulan

Langkah ini berisi tentang hasil pengujian serta jawaban dari tujuan tugas akhir.

1.6 Sistematika Penulisan

Sistematika penulisan buku tugas akhir ini adalah sebagai berikut.

- BAB I PENDAHULUAN

Penulis mengemukakan latar belakang, rumusan masalah, tujuan, lingkup permasalahan, metodologi yang digunakan dalam pengerjaan tugas akhir, dan sistematika penulisan.

- BAB II TINJAUAN PUSTAKA

Pada bab ini akan diuraikan hal-hal yang berkaitan dengan solusi dari permasalahan pada tugas akhir ini.

- BAB III SPESIFIKASI DAN PERANCANGAN

Bab ini berisi tentang solusi, spesifikasi dari solusi, dan perancangan implementasi dari solusi tersebut.

- **BAB III IMPLEMENTASI DAN PENGUJIAN**

Pada bab ini akan diuraikan langkah implementasi solusi serta pengujian solusi terhadap spesifikasi yang sebelumnya ditentukan

- **BAB V KESIMPULAN**

Bab ini berisi tentang simpulan dari tugas akhir ini serta saran pengembangan bagi tugas akhir ini.

BAB II

MQTT dan AES

Untuk berkomunikasi melalui internet dibutuhkan suatu server dengan protokol komunikasi tertentu. Pesan dikirimkan ke server lalu server meneruskan pesan ke tujuan. Pesan tersebut harus di enkripsi agar tidak dapat dibaca oleh sembarang orang, hanya yang memiliki password yang dapat membaca pesan tersebut

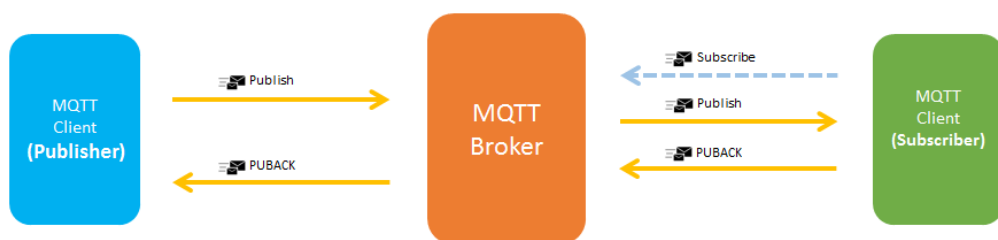
2.1 Message Queuing Telemetry Transport (MQTT)

Protokol MQTT merupakan protokol yang khusus di rancang untuk komunikasi *machine to machine* atau sederhananya untuk komunikasi dengan device atau mesin yang tidak memiliki alamat khusus, didesain untuk daerah/lokasi dengan resource jaringan yang terbatas.

Protokol ini memiliki kemampuan publish dan subscribe sehingga dapat digunakan untuk komunikasi 2 arah baik antara server ataupun dengan device yang lain. Terdapat 3 jenis QoS Level dalam MQTT.

1. QoS level 0

Pesan akan terkirim tanpa adanya acknowledge atau biasa disebut fire and forget, tidak ada jaminan pesan akan diterima oleh client dan pengirim juga tidak tahu apakah pesan itu diterima atau tidak.

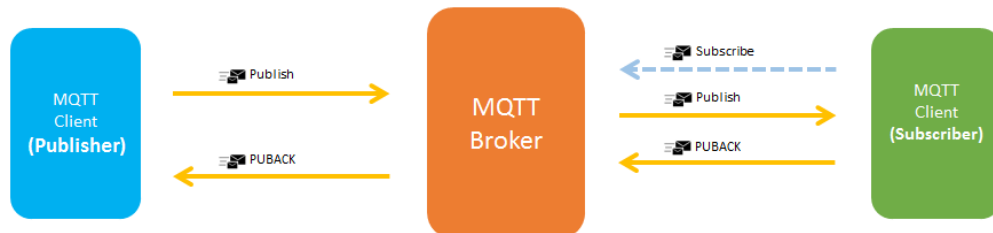


Gambar 1 Diagram QoS level 0

2. QoS level 1

Pesan akan dijamin sampai ke client / penerima paling tidak sekali jika menggunakan QoS level 1, namun jika ada kendala dalam jaringan & timeout, pesan bisa saja terkirim lebih dari satu kali. Pengirim akan tetap menyimpan pesan hingga mendapat

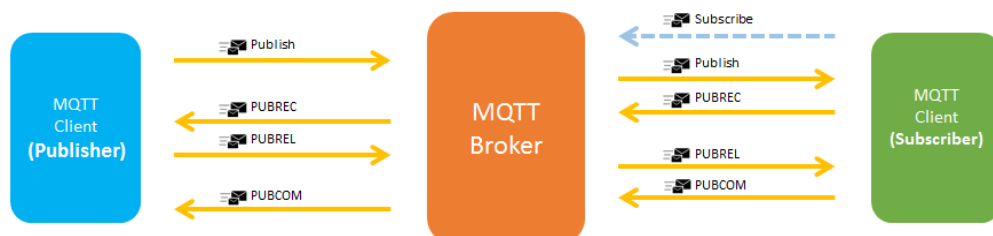
respon PUBACK dari penerima, jika pengirim tidak menerima PUBACK dalam waktu tertentu, pesan akan dikirim lagi.



Gambar 2 Diagram QoS level 1

3. QoS level 2

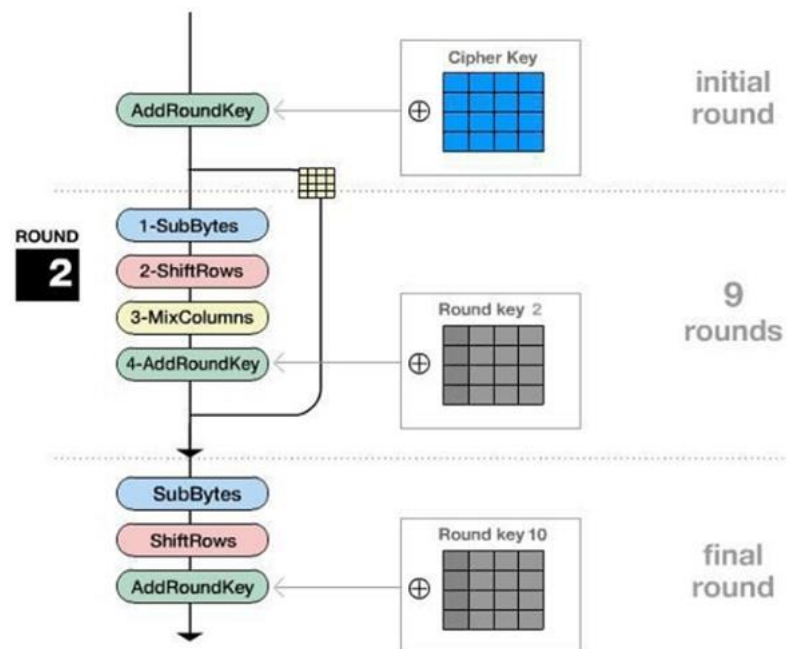
QoS tertinggi dalam MQTT adalah level 2, dimana pesan akan dijamin sampai dari pengirim ke penerima hanya sekali. QoS level ini adalah metode teraman dalam mengirimkan pesan namun juga yang paling lambat karena dilakukan pengecekan di pengirim maupun di penerima. Pengirim juga dapat mengetahui apakah pesan telah terkirim atau tidak.



Gambar 3 Diagram QoS level 2

2.2 Advanced Encryption Standard (AES)

Advanced Encryption Standard (AES) merupakan algoritma cryptographic yang dapat digunakan untuk mengamankan data. Algoritma AES adalah blok ciphertext simetrik yang dapat mengenkripsi (encipher) dan dekripsi (decipher) informasi. Enkripsi merubah data yang tidak dapat lagi dibaca disebut ciphertext; sebaliknya dekripsi adalah merubah ciphertext data menjadi bentuk semula yang kita kenal sebagai *plaintext*. Algoritma AES is menggunakan kunci kriptografi 128, 192, dan 256 bits untuk mengenkrip dan dekrip data pada blok 128 bits.



Gambar 4 diagram AES-128

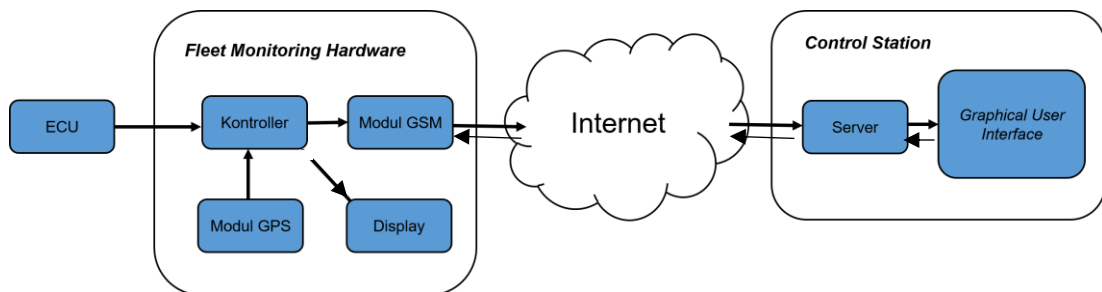
BAB III

SPESIFIKASI DAN PERANCANGAN

Pada bagian ini akan dijelaskan FMCS secara keseluruhan, spesifikasi sistem, dan perancangan berdasarkan spesifikasi tersebut.

3.1 FMCS

Fleet monitoring control system merupakan sistem monitoring armada guided bus yang memiliki arsitektur seperti gambar di bawah ini



Gambar 5 arsitektur FMCS

FMCS dapat di bagi menjadi dua bagian besar yaitu *fleet hardware* dan *control station*. *Fleet hardware* bertugas untuk membaca kondisi serta menentukan posisi armada data tersebut dikirimkan ke *control station* melalui jaringan internet. Pada bagian *control station* terdapat server dan GUI. Server berfungsi sebagai media berkomunikasi antara fleet hardware dengan GUI sedangkan GUI berfungsi untuk memantau armada.

3.2 Spesifikasi Subsistem

Dalam FMCS penulis mendesain dan implementasikan 3 bagian yaitu modul enkripsi dekripsi, implementasi server, dan algoritma penjadwalan. Adapun spesifikasi dari ketiga bagian tersebut adalah

1. Spesifikasi server
 - a. dapat meneruskan pesan yang diterima menggunakan protocol MQTT. Server harus dapat meneruskan pesan ke GUI dan sebaliknya agar fleet hardware dan GUI dapat berkomunikasi dengan baik.

- b. Latency server kurang dari sama dengan 5 ms. *Latency* server harus kecil sehingga pemrosesan data dapat dilakukan realtime.
2. Spesifikasi algoritma penjadwalan
 - a. Algoritma penjadwalan dapat menjaga armada pada jadwalnya. Algoritma ini berfungsi untuk menjaga armada pada jadwalnya secara otomatis.
3. Spesifikasi modul enkripsi-dekripsi
 - a. Data pesan antara armada dan control station hanya dapat dibaca oleh armada dan control station saja sehingga pihak lain tidak dapat mengetahui isi pesan antara *fleet hardware* dengan *GUI*

3.3 Perancangan Subsistem

Perancangan dapat dibagi menjadi tiga bagian yaitu keamanan informasi, server, dan algoritma penjadwalan.

3.2.1 Perancangan Server

3.2.1.1 Protokol Komunikasi

Untuk menghubungkan antara hardware yang terletak pada armada dengan *control station*, dibutuhkan jaringan internet yang menggunakan suatu protokol komunikasi tertentu agar *ECU monitoring* dan *control station* dapat berkomunikasi melalui internet.

Terdapat beberapa protokol komunikasi yang menghubungkan *device* (pada armada) dan server, yaitu MQTT dan XMPP. Kedua protokol ini memiliki kelebihan dan kekurangan seperti ditunjukkan pada tabel berikut.

Jenis	MQTT	XMPP
Transport	TCP	TCP
Messaging	Publish/Subscribe	Request/Response
SECURITY	medium	high
Protocol (tanpa enkripsi)	28 bytes	491 bytes
Protocol (dengan enkripsi)	68 bytes	308 bytes

QoS	Ada QoS	Tidak ada QoS
tipe	Asynchronous	Synchronous
<i>Resources Utilization</i>	Rendah	Tinggi
<i>Latency</i>	Millisecond ^[3]	second ^[4]

tabel 1 tabel perbandingan MQTT dan XMPP

Dari tabel di atas, MQTT menggunakan *resource utilization* yang lebih rendah dibandingkan XMPP sehingga MQTT tidak membutuhkan microcontroller yang memiliki clock tinggi. Selain itu, MQTT bersifat *asynchronous* yaitu server dan *device* tidak harus dalam keadaan siap untuk menerima pesan. Berbeda dengan XMPP yang menggunakan metoda *polling* dalam bertukar informasi antara *device* dan server. Jaringan GSM yang memungkinkan data hilang atau rusak pada proses komunikasi dengan server, MQTT memiliki protokol data yang lebih sedikit dibandingkan dengan XMPP sehingga MQTT lebih tahan terhadap jaringan yang tidak stabil. Selain itu MQTT didukung dengan fitur QoS sehingga apabila ada data yang hilang pada proses komunikasi dengan server, maka server akan meminta data yang hilang tersebut sampai data tersebut diterima oleh server. *Latency* yang dimiliki oleh XMPP berkisar pada orde second sedangkan *latency* yang dimiliki oleh MQTT berkisar pada orde millisecond. Spesifikasi dari FMCS mengharuskan hardware mengupdate data tiap 0.6 detik sehingga MQTT dipilih sebagai protokol yang digunakan untuk komunikasi antara fleet dengan control station.

3.2.1.2 Komputer Server

Untuk menerima data dari fleet dibutuhkan server yang berada di control station bertugas mengirimkan data ke armada-armada juga menerima data-data dari armada tersebut.

Menurut hasil benchmark yang dilakukan oleh Scalagent^[3] dengan kondisi sebagai berikut:

- Jumlah client 1000 client
- Terdapat 10000 message/s
- 1 subscriber

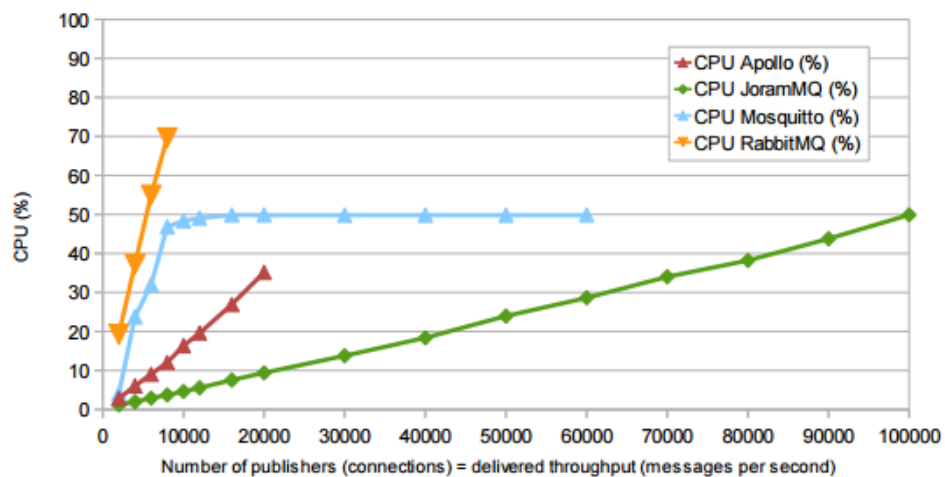
Digunakan server dengan spesifikasi sebagai berikut:

Tabel 2 Perbandingan server

Jenis	parameter
CPU	Intel Core 2 Duo CPU E8400 3.00GHz
RAM	4 GB
HDD	SATA 7200 RPM
Network	Gigabit switch

tabel 3 table spesifikasi computer server benchmark scaleagent

Hasil *benchmark* menunjukan rata rata CPU usage adalah 50%. Seperti pada tabel dibawah.



Gambar 6 Hasil benchmark CPU usage terhadap jumlah pesan

Kondisi ini dijadikan asumsi dalam menentukan spesifikasi server yang digunakan dalam FMCS ini adalah :

- Jumlah client sekitar 40 client
- Terdapat maksimal 80 message/s (message setiap 0.7 detik tiap client)
- 1 subscriber

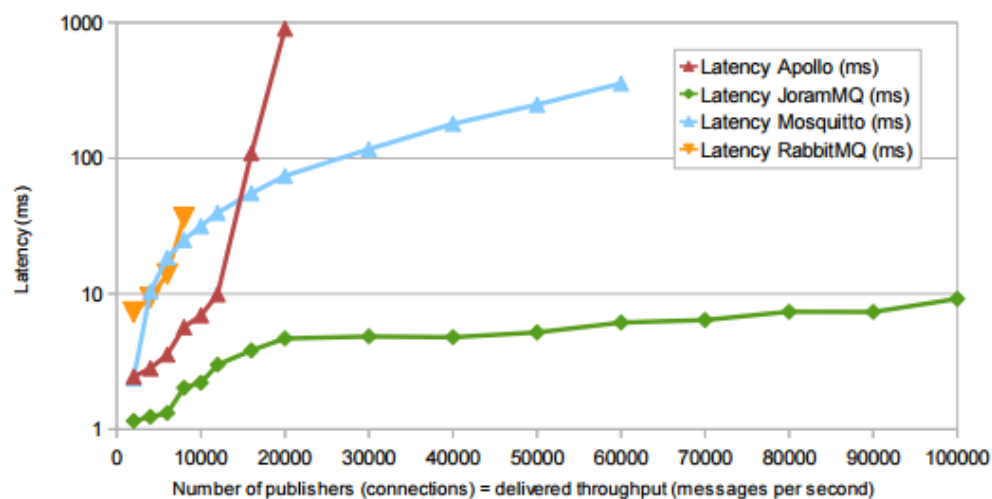
Maka spesifikasi komputer

- Intel i5-4300M 2.5 Ghz
- 8 GB RAM
- FS1104 fiberhome router

Dapat menanggung beban server sesuai asumsi diatas.

3.2.2.3 Message Broker

Message Broker merupakan *broker* yang melanjutkan pesan dari pengirim ke penerima. Broker MQTT ini menyediakan aplikasi/program yang dapat di install pada server. Terdapat beberapa broker yang menyediakan server MQTT, baik gratis maupun berbayar. Pertimbangan dalam pemilihan broker ini antara lain adalah latensi server. Berikut grafik yang menunjukkan latensi hasil pengujian oleh Scalagent ^[3] menggunakan beberapa broker.



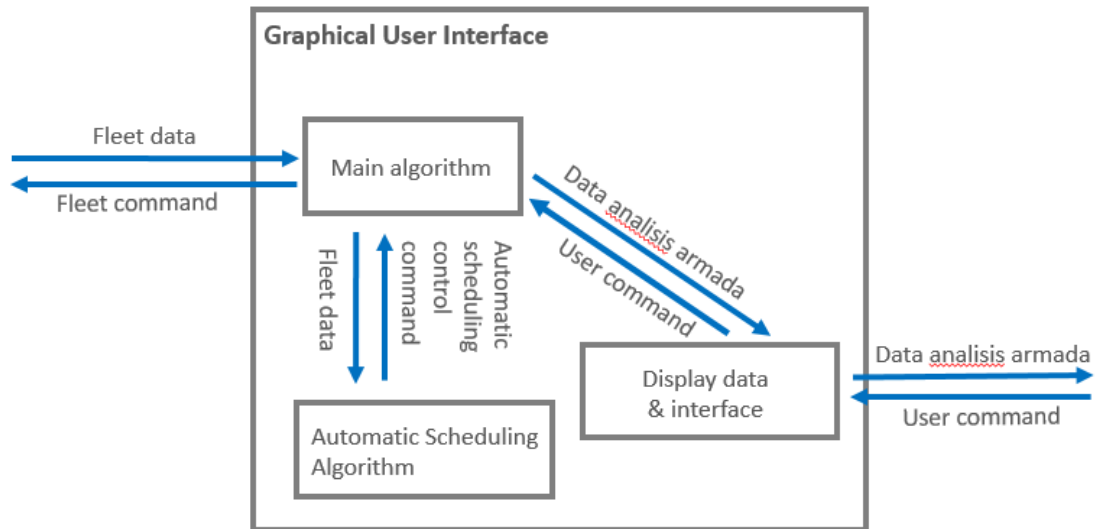
Gambar 7 Hasil benchmark latency terhadap jumlah pesan

Broker	Biaya	Platform
Mosquitto	gratis	Windows
JoramMQ	gratis	Java
RabbitMQ	\$99	Cloud
Appollo	gratis	Cloud

Dari gambar di atas, dilihat kondisi pada klien sekitar 1000 dengan 10000 message/s dan 1 *subscriber*. Dari keempat server yang dites, dapat dilihat server Mosquitto memiliki latensi di bawah 10 ms pada kondisi tersebut. Oleh Karena pertimbangan

latensi dan biaya server serta platform yang digunakan server, dipilih server Mosquitto.

3.2.2 Perancangan Algoritma Penjadwalan



Gambar 8 DFD level 2 GUI

GUI dapat dibagi menjadi tiga bagian yaitu *main algorithm*, *scheduling algorithm* dan *display data*. *Main algorithm* berfungsi untuk menangkap pesan dari server lalu melakukan enkripsi terhadap pesan tersebut. *Display data* bertanggung jawab untuk menampilkan data data yang didapat ke *user*. *Scheduling algorithm* berfungsi untuk mengatur penjadwalan armada.

Algoritma ini bertujuan agar persebaran armada merata pada jalur trayeknya sehingga penumpang tidak perlu menunggu armada terlalu lama. Algoritma ini memiliki beberapa pendekatan:

- Pengaturan jarak antar bus dengan mengatur *window time*
Pada algoritma ini jarak antar bus dijaga tetap sehingga bus tersebar merata. Jarak antar bus dapat dikontrol dengan mengatur waktu berhenti dan berangkat setiap bus pada masing masing halte (*window time*). Apabila jarak antar dua bus berkurang maka *window time* bus belakang akan bertambah sedangkan *window time* pada bus depan akan berkurang sehingga jarak kedua bus ini kembali pada jarak seharusnya. Sebaliknya apabila terdapat 2 bus yang saling berjauhan maka

window time bus belakang akan berkurang sedangkan *window time* pada bus depan akan bertambah sehingga jarak kedua bus ini kembali pada jarak seharusnya.

- Pengaturan jarak antar bus dengan mengatur kecepatan

Pada pendekatan ini jarak antar bus dijaga tetap dengan mengatur kecepatan bus dan membuat *window time* yang konstan. Jika terdapat 2 bus, misal bus A dan bus B, yang saling berjauhan maka kecepatan salah bus A dipercepat dengan memperhitungkan jarak bus tersebut dengan bus lainnya, apabila menaikkan kecepatan bus A berdampak menjauhkan bus A dengan bus di belakangnya maka bus B harus menurunkan kecepatan sedangkan bus A mempertahankan kecepatannya.

Dalam pemilihan algoritma ini terdapat asumsi yaitu bus berjalan pada jalur khusus yang bebas macet. Adapun untuk memilih algoritma yang paling baik untuk penjadwalan dapat dilihat dari kelebihan dan kekurangan dari tiap algoritma tersebut seperti pada tabel dibawah ini.

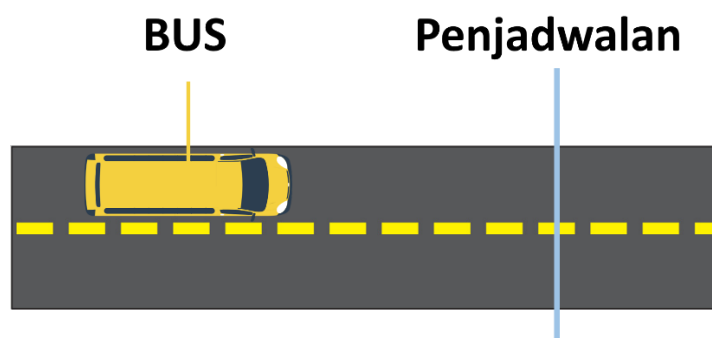
keterangan	Pendekatan jarak antar bus dengan mengatur kecepatan		Pendekatan jarak antar bus dengan mengatur <i>window time</i>	
	+	-	+	-
konsumen	Tidak membuat penumpang menunggu di halte			Membuat penumpang menunggu di halte
implementasi		Jika dilakukan manual susah untuk menyesuaikan kecepatan	Mudah impementasi oleh pengemudi	
Kehandalan sistem	Window untuk menyesuaikan scheduling banyak			Window untuk menyesuaikan scheduling sedikit

tabel 4 tabel perbandingan algoritma penjadwalan

Terdapat beberapa poin yang menjadi pertimbangan yaitu konsumen, konsumen lebih suka jika kendaraan melaju walaupun dengan kecepatan rendah dibandingkan dengan

jika kendaraan diam. Pada implementasinya algoritma dengan pengaturan jarak antar bus dengan mengatur *window time* lebih mudah di implementasikan karena pengemudi hanya mengatur kapan bus berhenti dan melaju. Sedangkan pada pengaturan jarak antar bus dengan mengatur kecepatan lebih sulit di implementasikan karena pengemudi harus menyesuaikan kecepatan setiap saat dengan kecepatan yang diperintahkan oleh *control station*. poin yang menjadi pertimbangan selanjutnya adalah kehandalan sistem, yang dimaksud dengan kehandalan sistem adalah kemampuan sistem untuk mengontrol armada apabila terjadi ketidaksesuaian penjadwalan dengan implementasi di lapangan. pengaturan jarak antar bus dengan mengatur kecepatan memiliki lebih banyak kesempatan untuk menyesuaikan dengan jadwal seharusnya karena bus lebih banyak menghabiskan waktu dalam perjalanan dibandingkan dengan berhenti di halte. Dari uraian di atas dipilih pengaturan jarak antar bus dengan mengatur kecepatan.

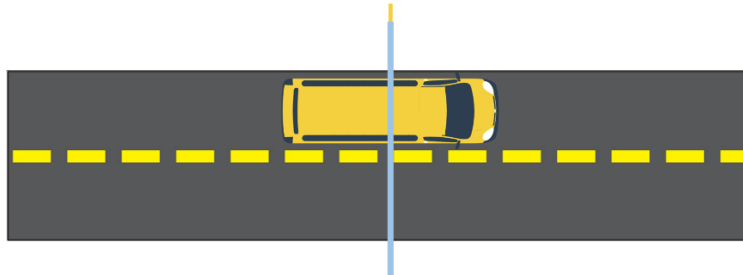
pengaturan jarak antar bus dengan mengatur kecepatan dapat dilakukan dengan mendesain algoritma penjadwalan yang dibuat adalah dengan menentukan jadwal *fleet* tiap waktu. Penjadwalan *fleet* dilakukan dengan menentukan posisi *fleet* tiap waktu, ketika terdapat perbedaan diantara posisi *fleet* dengan posisi *fleet* yang telah terjadwal maka algoritma akan menentukan kecepatan *fleet* untuk mengejar ketertinggalan/mengurangi kelebihan posisi *fleet*.



Gambar 44 *fleet* tertinggal dari penjadwalan

Gambar diatas menunjukkan posisi *fleet* tertinggal dari penjadwalan maka algoritma akan menambah kecepatan *fleet* untuk mengejar ketertinggalan.

Penjadwalan BUS

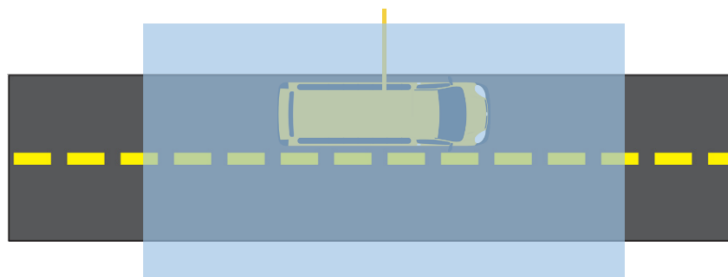


Gambar 46 *fleet* sama dengan penjadwalan

Ketika posisi *fleet* telah sesuai dengan penjadwalan maka kecepatan *fleet* dan kecepatan penjadwalan sama.

Kecepatan yang ditentukan oleh algoritma apabila terdapat perbedaan posisi akan dikirimkan ke pengemudi *fleet* setelah itu pengemudi harus menaikkan/menurunkan kecepatan *fleet*-nya. Pada praktek nyatanya akan sangat sulit untuk menyamakan posisi *fleet* dengan posisi penjadwalan sehingga perlu ditambahkan ditambahkan toleransi bagi *fleet* seperti gambar di bawah ini.

Penjadwalan BUS



Gambar 47 Toleransi penjadwalan

Fleet dianggap sesuai dengan jadwal apabila posisi *fleet* berada didalam toleransi itu.

Jika posisi penjadwalan bertemu dengan halte maka penjadwalan akan berhenti dengan durasi tertentu pada halte tersebut. Setelah itu penjadwalan akan melanjutkan menentukan posisi selanjutnya.

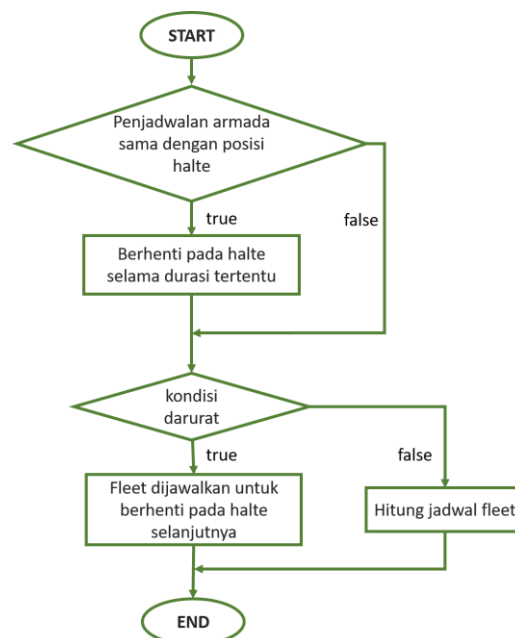
Tiap bus memiliki penjadwalan masing masing yang menjadi acuan apakah bus tersebut terlambat atau mendahului penjadwalan.

Algoritma penjadwalan juga dapat menangani kondisi darurat dimana jika salah satu fleet mengalami masalah sehingga tidak bisa melanjutkan perjalanan maka semua fleet akan diarahkan untuk berhenti pada halte berikutnya atau halte yang berada didepannya. Setelah fleet yang bermasalah diperbaiki atau dikeluarkan dari jalur maka penjadwalan akan menyesuaikan dirinya sendiri sehingga fleet mempunyai jarak yang sama dengan fleet yang lainnya.

Adapun rancangan algoritma penjadwalan adalah sebagai berikut.

- Prosedur penjadwalan

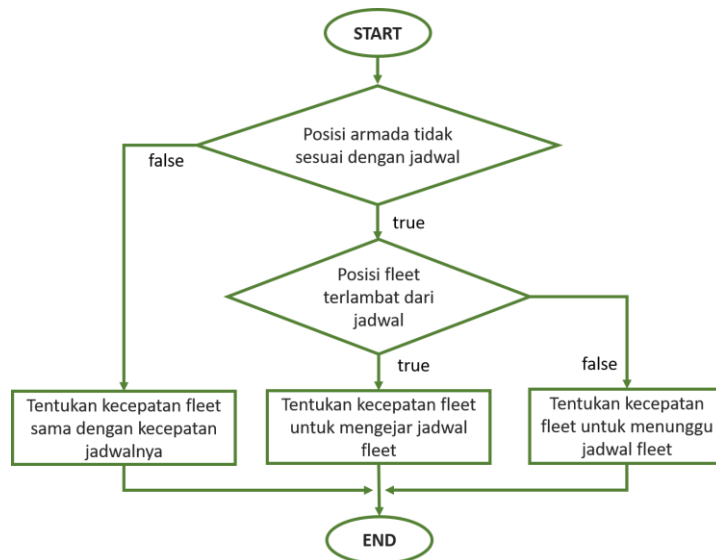
Prosedur ini akan mengatur jadwal dari armada tiap detik.



Gambar 9 flowchart prosedur penjadwalan

- Prosedur cek penjadwalan

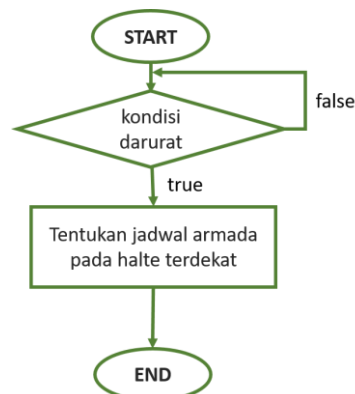
Prosedur ini berfungsi untuk memeriksa kondisi dari tiap armada terhadap jadwalnya.



Gambar 10 flowchart prosedur cek jadwal

- **Prosedur kondisi darurat**

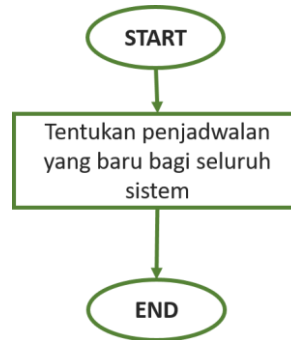
Prosedur ini dijalankan jika armada mengalami kondisi darurat dan pengemudi armada menekan tombol darurat pada fleet hardware. Prosedur ini berfungsi untuk menentukan halte terdekat sehingga dalam keadaan darurat armada berhenti pada halte tersebut



Gambar 11 flowchar prosedur kondisi darurat

- Prosedur hitung penjadwalan

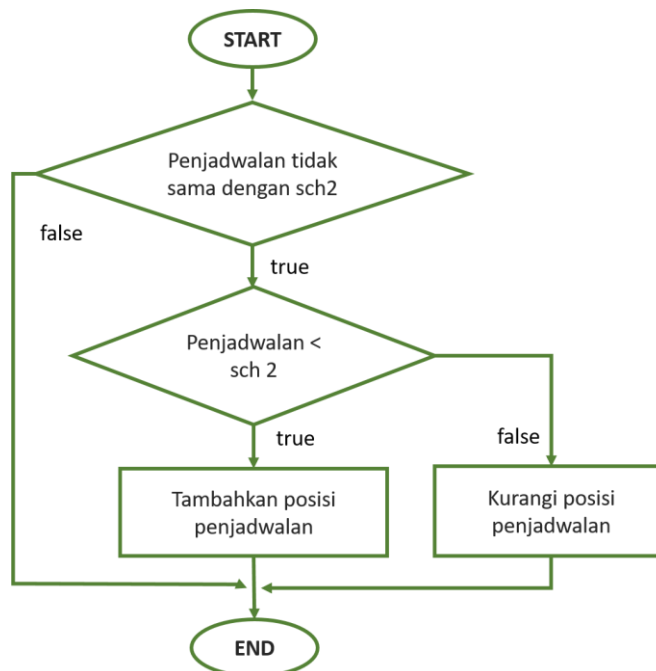
Prosedur ini dipanggil setelah situasi darurat dapat diatasi. Prosedur ini menentukan posisi array penjadwalan yang baru sesuai dengan jumlah armada yang masih dapat beroperasi.



Gambar 12 flowchart prosedur hitung penjadwalan

- Prosedur cek sch2

Sch2 berfungsi ketika kondisi darurat terjadi. Variable sch2 berisi posisi countersch ideal. Prosedur ini akan membuat counter sch agar sama dengan sch2 sehingga penjadwalan kembali normal setelah terjadi kondisi darurat.



Gambar 13 flowchart prosedur cek sch2

3.2.3 Perancangan modul dekripsi - enkripsi

Modul enkripsi dan dekripsi bertujuan untuk mengamankan data saat data di-publish melalui internet adapun rancangan dan implementasi modul enkripsi-dekripsi adalah sebagai berikut

Terdapat 3 algoritma enkripsi yang memiliki kelebihan dan kekurangan^[2] seperti tabel dibawah ini.

NAMA	<i>Avalance effect</i>	<i>throughput</i>
AES	93%	950 bits/sec
DES	75%	500 bits/sec
RSA	60%	100 bits/sec

tabel 5 tabel performa algoritma enkripsi dekripsi

Avalance effect merupakan perubahan hasil enkripsi yang terjadi ketika sebagian dari pesan diubah. Semakin besar nilai ini maka semakin baik

Throughput adalah umlah data yang diproses tiap detiknya. Semakin besar throughput maka semakin cepat algoritma melakukan enkripsi.

Dari tabel diatas maka enkripsi dan dekripsi yang dilakukan pada sistem FMCS ini menggunakan AES.

BAB IV IMPLEMENTASI DAN HASIL PENGUJIAN

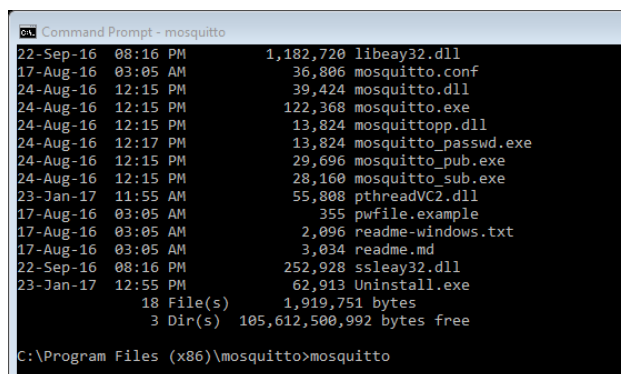
4.1 Implementasi

4.1.1 Implementasi Server

Terdapat beberapa langkah dalam meng-*install* server MQTT yaitu:

- download installer dan beberapa file tambahan yaitu libeay32.dll, ssleay32.dll dan pthreadVC2.dll (link file terlampir).
- Setelah itu *install* OpenSSL dan *copy* libeay32.dll dan ssleay32.dll dari *directory* openssl ke *directory* mosquito.
- *Copy* pthreadVC2.dll ke *directory* mosquito, pada tahap ini server telah selesai pasang.

Untuk menjalankan server akses mosquitto dari Command Prompt pada windows seperti gambar di bawah



```
Command Prompt - mosquitto
22-Sep-16 08:16 PM      1,182,720 libeay32.dll
17-Aug-16 03:05 AM       36,806 mosquitto.conf
24-Aug-16 12:15 PM       39,424 mosquitto.dll
24-Aug-16 12:15 PM      122,368 mosquitto.exe
24-Aug-16 12:15 PM       13,824 mosquitto_top.dll
24-Aug-16 12:17 PM       13,824 mosquitto_passwd.exe
24-Aug-16 12:15 PM       29,696 mosquitto_pub.exe
24-Aug-16 12:15 PM       28,160 mosquitto_sub.exe
23-Jan-17 11:55 AM       55,808 pthreadVC2.dll
17-Aug-16 03:05 AM         355 pwfile.example
17-Aug-16 03:05 AM         2,096 readme-windows.txt
17-Aug-16 03:05 AM          3,034 readme.md
22-Sep-16 08:16 PM      252,928 ssleay32.dll
23-Jan-17 12:55 PM        62,913 Uninstall.exe
      18 File(s)      1,919,751 bytes
       3 Dir(s)  105,612,500,992 bytes free

C:\Program Files (x86)\mosquitto>mosquitto
```

Gambar 14 mejalan server melalui Command Prompt

Untuk mengecek server ketik “netstat -an” pada Command Prompt apabila terdapat *port* 1883 yang terbuka maka service MQTT telah berjalan karena *default port* yang digunakan MQTT adalah 1883

Selanjutnya penulis melakukan *port forwarding* pada port 1883, sesuai dengan port MQTT, pada IP sesuai dengan IP komputer server yaitu 192.168.43.85. *Port forwarding* bertujuan agar port 1883 pada 192.168.43.85 dapat diakses melalui jaringan internet sehingga server dapat diakses oleh *client-client* melalui jaringan internet. Penulis menggunakan router dengan seri AN5506-04 FG. Adapun langkah melakukan *port forwarding* adalah sebagai berikut

- Buka *browser*, akses *default gateway* pada *router* yang digunakan yaitu 192.168.1.1. setelah itu akan muncul tampilan seperti gambar di bawah



Gambar 15 default gateway pada router

Masukan ID dan password

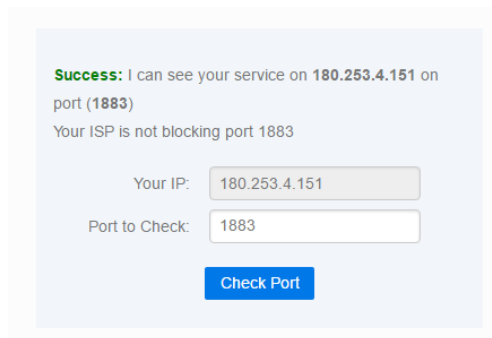
- Buka halaman *port forwarding* pada bagian tab *application*. Klik add lalu isi pengaturan sebagai berikut
 - IP = 192.168.1.7, sesuai dengan IP lokal computer, dapat dilihat pada Command Prompt dengan perintah “ipconfig” lihat pada bagian IPv4 “address”
 - Public Port = 1883-1883 port yang akan terlihat pada bagian internet
 - Private Port = 1883-1883 port yang digunakan MQTT

Lalu simpan pengaturan dan jalankan *profile* tersebut

WAN	WAN0 ▼	
Discription	mqtt	
Public Port	1883	- 1883
IP	192.168.1.7	
Private Port	1883	- 1883
Protocol	ALL ▼	
Enable	Enable ▼	
<input type="button" value="Apply"/> <input type="button" value="Cancel"/>		

Gambar 16 pengaturan port forwarding

- Cek apakah port sudah terbuka melaui website <http://www.canyouseeme.org/> atau menggunakan website port checker lainnya

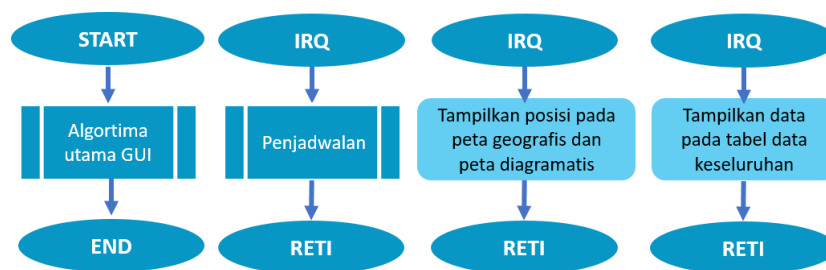


Gambar 17 hasil port checker

Isi IP dengan ip internet computer anda dan isi port dengan 1883, dapat dilihat pada gambar diatas port berhasil terbuka

4.1.2 Implementasi Penjadwalan

GUI memiliki susunan flowchart utama seperti gambar dibawah ini.



Gambar 18 flowchart utama GUI

Adapun implementasi algoritma penjadwalan dilakukan pada visual studio menggunakan Bahasa c#. dalam dokumen ini implementasi di dokumentasikan dalam bentuk *pseudocode* seperti dibawah ini.

- Prosedur penjadwalan

```

if posisi penjadwalan = posisi halte
    counter durasi halte++
    If counter durasi halte > durasi berhenti
        posisi penjadwalan ++
Else
    If kondisi darurat = true
        posisi penjadwalan ++ sampai pada halte terdekat di depan armada
    else
        posisi penjadwalan ++
  
```

Prosedur ini dipanggil tiap 200ms sehingga posisi penjadwalan bergerak dengan kecepatan 54km/s.

- Prosedur cekjadwal

```

    If posisi armada != posisi penjadwalan
      If posisi penjadwalan > posisi armada
        If |posisi penjadwalan - posisi armada| > 30 satuan array
          fleet speed = 86.4 km/h
        else
          fleet speed = 64.8 km/h
      else
        If |posisi penjadwalan - posisi armada| > 30 satuan array
          fleet speed = 21.6 km/h
        else
          fleet speed = 43.2 km/h
    else
      fleet speed = 54 km/h

```

Prosedur ini terdapat pada algoritma utama GUI. Prosedur ini dipanggil ketika pesan dari armada selesai di parsing.

Data posisi yang didapatkan dari fleet direpresetasikan pada array trayek yang dilakukan pada bagian GUI sebelumnya. Prosedur ini berfungsi untuk mengecek posisi *real fleet* dengan posisi *fleet* terjadwal. Apabila posisi *real fleet* mendahului posisi *fleet* terjadwal maka prosedur ini menghitung perbedaan jarak antara posisi *real fleet* dengan posisi *fleet* terjadwal dalam representasi array trayek. Perbedaan tersebut digunakan untuk menentukan kecepatan *fleet* untuk menyesuaikan dengan posisi *fleet* terjadwal. Semakin besar perbedaan antara posisi *real fleet* dengan posisi *fleet* terjadwal kecepatan semakin rendah. Jika perbedaan diatas 30 array maka kecepatan akan diturunkan menjadi 21.6 km/jam selain itu maka kecepatan akan diturunkan menjadi 43.2 km/jam. Jika posisi *real fleet* tertinggal dari posisi *fleet* terjadwal prosedur ini menghitung perbedaan jarak antara posisi *real fleet* dengan posisi *fleet* terjadwal dalam representasi array trayek. Sebaliknya, semakin besar perbedaan antara posisi *real fleet* dengan posisi *fleet* terjadwal kecepatan semakin tinggi. Jika perbedaan diatas 30 array maka kecepatan akan diturunkan menjadi 86.4 km/jam selain itu maka kecepatan akan diturunkan menjadi 64.8 km/jam Apabila posisi *real fleet* sama dengan posisi *fleet* terjadwal maka kecepatan akan sama dengan kecepatan normal fleet.

Kecepatan tersebut dipilih sedemikian rupa agar ketika di konversi dalam satuan array menjadi bilangan bulat sehingga tidak dilakukan pembulatan.

Hal tersebut bertujuan agar penjadwalan lebih presisi

Kecepatan(km/jam)	Kecepatan(satuan array/detik)
21.6	2
43.2	4
64.8	6
86.4	8

tabel 6 tabel kecepatan penjadwalan

- Prosedur kondisi darurat

```

If kondisi darurat = true
    for i=0 until i<jumlah halte
        if posisi armada < halte[i]
            halte terdekat didepan armada = halte[i]

```

Prosedur ini akan mejadwalakan armada untuk berhenti pada halte terdekat ketika terjadi keadaan darurat. Prosedur ini hanya dipanggil ketika keadaan darurat berlangsung

Adapun posisi penjadwalan seharusnya didapatkan dari perhitungan jarak total rute dibagi jumlah fleet yang masih dapat beroperasi.

- Prosedur hitung penjadwalan

```

for i=1 until i<fleetsch2.length
    fleetsch2[i] += fleethsch2[i-1] + jarak

```

Prosedur ini dipanggil setelah situasi darurat dapat diatasi. Prosedur ini menentukan posisi array penjadwalan yang baru sesuai dengan jumlah armada yang masih dapat beroperasi.

- Prosedur cek sch2

```

If sch2 != posisi penjadwalan
    If sch2 > posisi penjadwalan
        posisi penjadwalan++
    else
        posisi penjadwalan--

```

Sch2 berfungsi ketika kondisi darurat terjadi. Variable sch2 berisi posisi countersch ideal. Prosedur ini akan membuat counter sch agar sama dengan sch2 sehingga penjadwalan kembali normal setelah terjadi kondisi darurat.

4.1.3 Implementasi Modul Enkripsi Dekripsi

Implementasi AES dilakukan pada fleet hardware dan juga GUI menggunakan AES-128 bit. Pada fleet hardware digunakan library AES.h. Enkripsi dilakukan dengan menggunakan prosedur dibawah ini.

```
do_aes_encrypt(plain, length, ciphertext, key, bits, iv);
```

Adapun penjelasan dari parameter prosedur tersebut adalah sebagai berikut

- *plaintext*, variable yang berisi pesan yang akan di enkripsi.
- *length*, merupakan panjang pesan.
- *ciphertext* adalah variable hasil enkripsi, isi dari variable ini akan dikirim ke server.
- *key* merupakan *string* tertentu digunakan sebagai “kunci” untuk melakukan enkripsi dan dekripsi dari suatu pesan.
- *bits* adalah jumlah bit dari key, pada implementasi ini digunakan 128 bit key.
- *iv* merupakan *initial vector* yang digunakan dalam proses enkripsi.

Setelah pesan dienkripsi pada fleet hardware. Pesan dikirimkan ke server lalu server meneruskan pesan tersebut pada GUI. Dekripsi pesan pada GUI menggunakan fungsi sebagai berikut.

```
DecryptStringFromBytes_Aes(ciphertext, key, iv);
```

Fungsi tersebut akan menghasilkan string berisi pesan yang sudah terdekripsi. Untuk mendekripsi pesan harus menggunakan variable key dan iv yang sama yang digunakan pada fleet hardware. Pesan tersebut akan diproses oleh GUI untuk menentukan command yang akan dikirim pada fleet hardware. Command tersebut harus melalui proses enkripsi pada GUI dengan menggunakan fungsi sebagai berikut.

```
EncryptStringToBytes_Aes(plaintext, key, iv);
```

Fungsi tersebut akan mengembalikan variable byte berisi pesan yang sudah terenkripsi. Pesan tersebut harus didekripsi pada fleet hardware menggunakan prosedur sebagai berikut.

```
do_aes_decrypt(ciphertext, length, message, key, bits, iv);
```

Prosedur tersebut akan menyimpan pesan hasil dekripsi pada variable message.

4.2 Prosedur dan Hasil Pengujian

4.2.1 Server

4.2.1.1 Parameter Pengujian Server

Pada pengujian kemampuan server, lingkup spesifikasi yang akan diuji adalah server dapat menerima 40 koneksi klien dengan tiap klien mengirimkan data sebanyak 32 byte tiap 0.7 detik dan server latency server dibawah 10 ms.

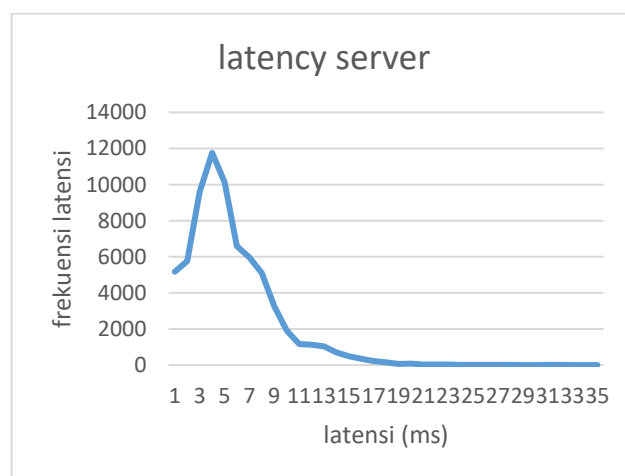
4.2.1.2 Prosedur Pengujian Server

Asumsi yang digunakan dalam pengujian server adalah *latency* yang ditimbulkan murni dari server. Adapun prosedur pengujian server adalah:

- Menjalankan server dan hubungkan dengan jaringan lokal.
- Menjalankan aplikasi dummy data dan hubungkan aplikasi dengan server pada jaringan lokal
- Sistem dibiarkan menyala selama 95 detik dan diambil datanya.

4.2.1.3 Hasil Pengujian Server

Pada pengujian ini dilakukan selama 20 menit didapatkan rata-rata latensi adalah 4.81 ms dengan persebaran data seperti grafik dibawah.



Gambar 19 Grafik persebaran latensi

Dengan menganggap grafik tersebut mendekati persebaran *Gaussian* maka dapat ditentukan confidence interval menggunakan rumus sebagai berikut

$$\bar{x} \pm z^* \frac{\sigma}{\sqrt{n}}$$

Rata-Rata	Standar Deviasi	Margin of Error
4.81	2.56	2.01 ms

Maka didapat rata rata persebaran 4.81 ± 2.01 ms dengan selang kepercayaan 95%

Hasil pengujian menunjukan server memiliki latensi rata rata dibawah 5 ms.

4.2.2 Algortima Penjadwalan

4.2.2.1 Parameter Pengujian Algortima Penjadwalan

Pada pengujian algoritma penjadwalan, lingkup spesifikasi yang akan diuji adalah spesifikasi nomor 5, yaitu ada algoritma yang dapat mengatur penjadwalan operasi tiap armada.

4.2.2.2 Prosedur Pengujian Algortima Penjadwalan

Asumsi yang digunakan pada pengujian ini adalah pengemudi *guided bus* selalu mengikuti perintah dari *control station* sehingga program *dummy data* merepresentasikan keadaan armada dengan tepat. Adapun prosedur pengujian adalah

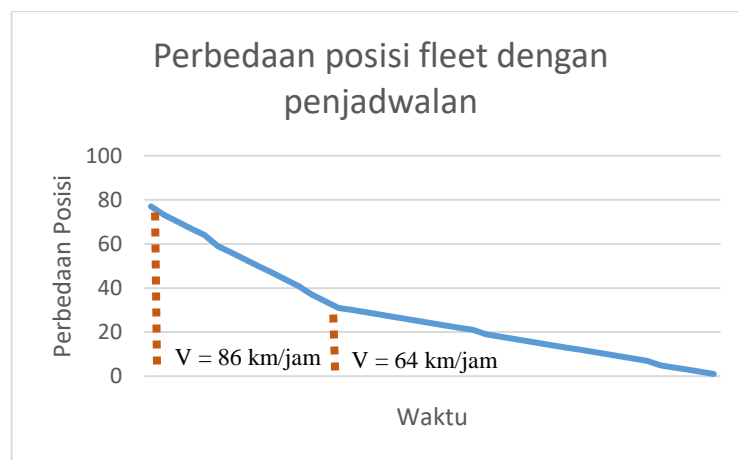
- Menjalankan server dan hubungkan pada jaringan internet.
- Menjalankan GUI dan hubungkan pada server melalui jaringan local.
- Atur posisi *fleet* agar lebih cepat dari penjadwalan pada aplikasi *Dummy Data*.
- Menjalankan aplikasi *Dummy Data* dan hubungkan pada server melalui internet.
- Merekam hasil *screenshot* dan perbedaan waktu yang dibutuhkan *fleet* untuk sesuai dengan penjadwalan.
- Atur posisi *fleet* agar lebih lambat dari penjadwalan pada aplikasi *Dummy Data*.
- Menjalankan aplikasi *Dummy Data* dan hubungkan pada server melalui internet.
- Merekam hasil *screenshot* dan perbedaan waktu yang dibutuhkan *fleet* untuk sesuai dengan penjadwalan.
- Mengatur satu *fleet* menjadi non-aktif pada aplikasi *Dummy Data*.
- Menjalankan aplikasi *Dummy Data* dan hubungkan pada server melalui internet.
- Merekam hasil *screenshot* dan perbedaan waktu yang dibutuhkan *fleet* untuk sesuai dengan penjadwalan.

4.2.2.3 Hasil Pengujian Algoritma Penjadwalan

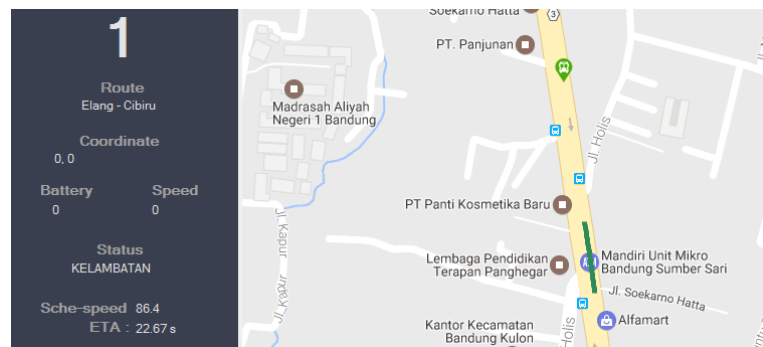
Hasil pengujian ini dibagi menjadi tiga, yaitu hasil pengujian untuk *fleet* mendahului penjadwalan, *fleet* terlambat dari penjadwalan dan *fleet* sesuai dengan penjadwalan.

- *Fleet* Terlambat dari Penjadwalan

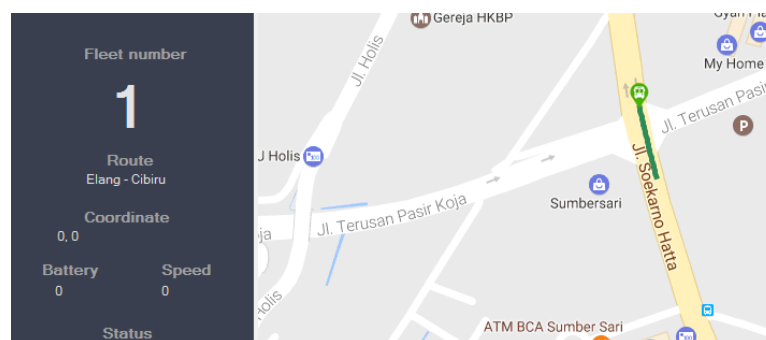
Untuk pengujian ini, *fleet* diatur mulai dari segmen ke-135 pada Jalan Soekarno-Hatta dengan posisi penjadwalan di segmen ke-230 pada Jalan Soekarno-Hatta. *Fleet* telah berhasil mencapai penjadwalannya dengan mengatur kecepatan untuk menyusul ketertinggalan. Berikut grafik yang menunjukkan perbedaan posisi *fleet* dengan posisi penjadwalannya terhadap waktu.



Gambar 20 Perbedaan posisi fleet dengan penjadwalan



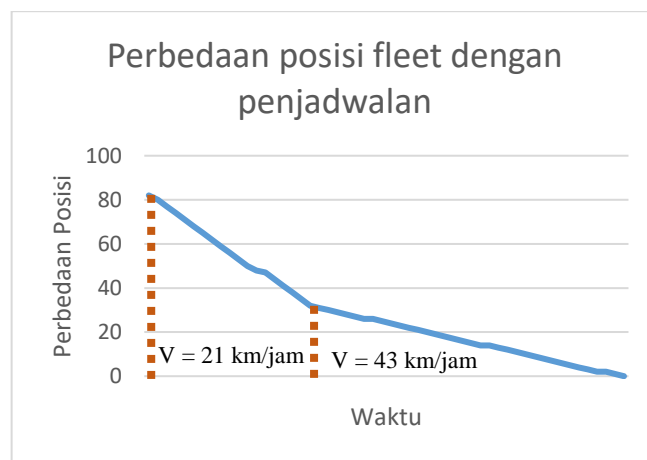
Gambar 21 Tampilan GUI saat fleet (pin hijau muda) tertinggal dari penjadwalan (garis hijau)



Gambar 22 Tampilan GUI saat fleet (pin hijau muda) tepat dengan penjadwalan (garis hijau)

- *Fleet* Mendahului dari Penjadwalan

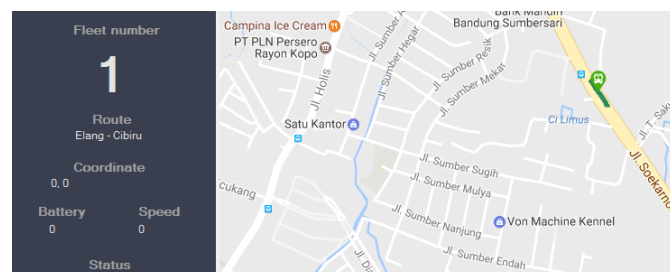
Untuk pengujian ini, *fleet* diatur mulai dari segmen ke-325 pada Jalan Soekarno-Hatta dengan posisi penjadwalan di segmen ke-230 pada Jalan Soekarno-Hatta. *Fleet* telah berhasil mencapai penjadwalannya dengan mengatur kecepatan untuk kembali ke penjadwalannya. Berikut grafik yang menunjukkan perbedaan posisi *fleet* dengan posisi penjadwalannya terhadap waktu.



Gambar 23 Perbedaan posisi fleet dengan penjadwalan



Gambar 24 Tampilan GUI saat fleet (pin hijau muda) mendahului dari penjadwalan (garis hijau)



Gambar 25 tampilan GUI saat armada sesuai dengan penjadwalan

- *Fleet* Sesuai dengan Penjadwalan

Untuk pengujian ini, *fleet* diatur mulai dari segmen ke-230 pada Jalan Soekarno-Hatta dengan posisi penjadwalan di segmen ke-230 pada Jalan Soekarno-Hatta. *Fleet* telah berhasil mencapai penjadwalannya dengan mengatur kecepatan untuk kembali ke penjadwalannya. Berikut grafik yang menunjukkan perbedaan posisi *fleet* dengan posisi penjadwalannya terhadap waktu.



Gambar 26 grafik perbedaan posisi *fleet* dengan penjadwalan



Gambar 27 tampilan GUI saat armada sesuai dengan jadwalnya

4.2.3 Modul Enkripsi Dekripsi

4.2.3.1 Parameter Pengujian Modul Enkripsi-Dekripsi

Pada pengujian ini akan di test spesifikasi modul enkripsi dan dekripsi yaitu a.Data pesan antara armada dan control station hanya dapat dibaca oleh armada dan control

station saja sehingga pihak lain tidak dapat mengetahui isi pesan antara fleet hardware dengan GUI

4.2.3.2 Prosedur Pengujian Modul Enkripsi-Dekripsi

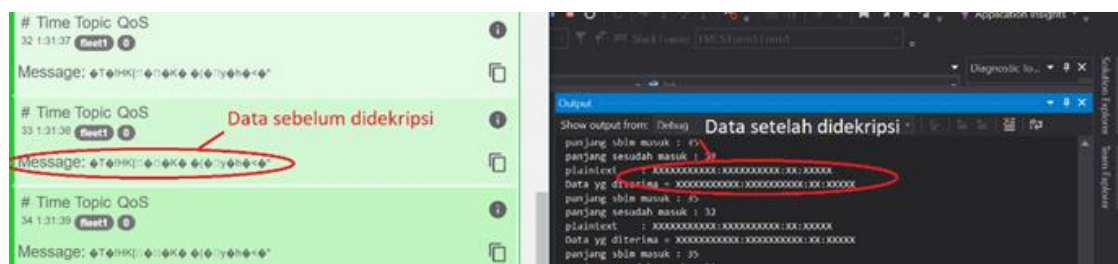
Pengujian ini dilakukan dengan langkah langkah sebagai berikut:

- Menjalankan server dan hubungkan pada jaringan internet.
- Menjalankan GUI dan hubungkan pada server melalui jaringan lokal.
- Menyalakan fleet hardware.
- Mengamati hasil data yang diterima oleh GUI dan client.

Dalam pengujian ini fleet hardware melukan enkripsi dan GUI melakukan dekripsi.

4.2.3.3 Hasil Pengujian Modul Enkripsi-Dekripsi

Fleet hardware mengirimkan pesan “xxxxxxxxxxxx:xxxxxxxxxx:xx:xxxxxx” yang sebelumnya telah di enkripsi terlebih dahulu.



Gambar 28 hasil pengujian Modul enkripsi dekripsi

Hasil pengujian menunjukkan pesan yang didekripsi GUI sama dengan pesan sebelum dienkripsi oleh fleet hardware jika menggunakan key dan iv yang sama. Dari hasil pengujian juga menunjukkan ketika menggunakan iv atau key yang berbeda maka pesan hasil dekripsi tidak sesuai dengan pesan sebelum dienkripsi.

BAB V

KESIMPULAN

5. 1. Kesimpulan

Sistem FMCS khususnya bagian control station telah berjalan dengan baik. Server telah memenuhi spesifikasi dengan memiliki rata rata latensi 4.81 ms dan dapat meneruskan pesan sesuai topiknya. Pesan tersebut yang sebelumnya dienkripsi dapat didekripsi dengan menggunakan suatu *key* sehingga hanya hardware dan GUI yang dapat melakukan dekripsi terhadap pesan tersebut sehingga spesifikasi modul enkripsi dan dekripsi tercapai. Selain itu algoritma penjadwalan dapat mengarahkan armada agar sesuai dengan jadwalnya sehingga algoritma penjadwalan telah memenuhi spesifikasi.

5. 2. Saran

Untuk mengurangi latensi yang di alami server, *Control Station* harus menggunakan networking device yang memiliki latensi rendah seperti switch.

DAFTAR PUSTAKA

Razali Ritonga, “Kebutuhan Data Ketenagakerjaan Untuk Pembangunan Berkelanjutan”, 2014, BPJS.

scalagent, “Benchmark of MQTT servers”, 2015.

[http://www.scalagent.com/IMG/pdf/Benchmark MQTT servers-v1-1.pdf](http://www.scalagent.com/IMG/pdf/Benchmark_MQTT_servers-v1-1.pdf)

S.N.Ghosh, “ Performance Analysis of AES, DES, RSA And AES-DES-RSA Hybrid Algorithm for Data Security”.,2015:

[https://www.academia.edu/15749187/Performance_Analysis_of_AES_DES_RSA And AES-DES-RSA Hybrid Algorithm for Data Security](https://www.academia.edu/15749187/Performance_Analysis_of_AES_DES_RSA_And_AES-DES-RSA_Hybrid_Algorithm_for_Data_Security)

Isode, “M-Link & XMPP Performance Measurements over Satcom and Constrained IP Networks”, 2014. <http://www.isode.com/whitepapers/xmpp-performance-constrained.html>.

Djarot Saiful Hidayat , “Sepanjang 2015, Jumlah Motor di Jakarta Capai 13 Juta Unit”, 2015 <https://kumparan.com/ananda-wardhiati-teresia/sepanjang-2015-jumlah-sepeda-motor-capai-13-juta-unit-di-dki-jakarta>