SUBSISTEM ELEKTRONIK UNTUK AKUISISI DATA ARMADA PADA FLEET MONITORING AND CONTROL SYSTEM UNTUK GUIDED BUS

TUGAS AKHIR

Oleh

ALI ZAENAL ABIDIN

NIM : 13213106

PROGRAM STUDI TEKNIK ELEKTRO

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

2017

SUBSISTEM ELEKTRONIK UNTUK AKUISISI DATA ARMADA PADA FLEET MONITORING AND CONTROL SYSTEM UNTUK GUIDED BUS

Oleh :

Ali Zaenal Abidin

Tugas akhir ini telah diterima dan disahkan
Sebagai persyaratan untuk meperoleh gelar
SARJANA TEKNIK
di
PROGRAM STUDI TEKNIK ELEKTRO
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

Bandung, Mei 2017

Disetujui oleh :

Pembimbing I,

Ir. Arief Syaichu
Rohman, MEngSc, Ph.D

Pembimbing II,

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

Pembimbing III,

Ir. Agung Darmawan

ABSTRAK

SUBSISTEM ELEKTRONIK UNTUK AKUISISI DATA ARMADA PADA FLEET MONITORING AND CONTROL SYSTEM UNTUK GUIDED BUS

Oleh:
Ali Zaenal Abidin
NIM : 13213106
PROGRAM STUDI TEKNIK ELEKTRO

Kemacetan dan kurang populernya kendaraan umum sebagai alat transportasi utama selalu menjadi masalah yang dihadapi oleh kota Bandung. Guided bus sebagai kendaraan umum merupakan solusi yang tepat untuk mengatasi permasalahan tersebut. Dengan guided bus yang berjalan di jalur khusus, kenyamanan penumpang dan ketepatan waktu berangkat akan terjaga. Pada implementasinya, guided bus terdiri dari banyak subsistem. Dalam tugas akhir ini, dibuat sebuah prototipe salah satu sistem penyusun *guided bus*, yaitu *fleet monitoring and control system* (FMCS). FMCS adalah system komunikasi antara armada dengan control station untuk mengatur penjadwalan keberangkatan armada dengan masukan berupa posisi dan kondisi armada. Sistem ini terdiri dari subsistem elektronik, server dan GUI. Buku ini berisi penjelasan mengenai rancangan, implementasi dan pengujian subsistem elektronik yang berfungsi untuk mengirimkan data baterai, fault, rpm dan posisi armada ke server untuk diteruskan ke GUI pada *control station* serta menampilkan pesan dari GUI pada *control station*. Pada implementasinya, subsistem elektronik yang dirancang sudah dapat melakukan fungsi-fungsinya. Pada hasil pengujian, subsistem elektronik telah berhasil memenuhi spesifikasi yang diperlukan dengan 88.48% *error* GPS tidak mencapai 6 meter dan latensi jaringan GSM sebesar sebesar 76.1 ± 0.3 ms dengan 1.3% data memiliki latensi melebihi spesifikasi. Untuk latensi GSM yang lebih rendah, dapat digunakan SIM5215A atau Telit UC864-E yang bekerja di jaringan 3G untuk menggantikan SIM900 yang digunakan pada implementasi ini. Untuk data GPS yang lebih baik, dapat digunakan modul GPS dengan akurasi lebih tinggi.

Kata kunci— ECU, FMCS, CAN.

ABSTRACT

ELECTRONIC SUBSYSTEM FOR FLEET DATA ACQUISITION AT FLEET MONITORING AND CONTROL SYSTEM FOR GUIDED BUS

By:
Ali Zaenal Abidin
NIM : 13213106
ELECTRICAL ENGINEERING STUDY PROGRAM

Traffic jams and less-popular public transport as main transportation are the problems faced by Bandung. Guided Bus as a mass public transportation can be a solution for the problem. Guided bus running on special lane isolated from other vehicles will be an on-time, pleasant mass public transportation for the passengers. Guided bus consists of many subsystem. In this final project, one of the subsystem, fleet monitoring and control system (FMCS), was made as a prototype. FMCS is a communicarion system for fleets and control station to manage fleet schedulings with fleet positions and conditions. This system consists of electronic, server and GUI subsystems. This book contains informations about design, implementation and verification of electronic subsystem in FMCS for guided bus. Electronic subsystem is a part of FMCS for sending rpm, battery, fault and position data from fleet to control station, and display message from GUI at control station. During implementation process, electronic subsystem can do its functions. Testing results show that electronic subsystem can successfully satisfy the specifications with 88.48% of GPS error has less than 6 meters and GSM latency at 76.1 ± 0.3 ms with 1.3% of the data have higher latency than the specification. For better latency performance, SIM5215A or Telit UC864-E are both capable to work in 3G network and can be used over SIM900. For better GPS data, other GPS modules with better accuracy can be used.

Keywords— ECU, FMCS, CAN.

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 "*Subsistem Elektronik untuk Akuisisi Data Armada pada Fleet Monitoring and Control System untuk Guided Bus*" 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 dalam pengembangan sub-sistem ini, antara lain sebagai berikut.

1. Ayah, Ibu, dan Adik yang telah memberikan dukungan yang sangat besar kepada penulis selama menuntut ilmu di ITB.
2. Bapak Ir. Agung Darmawan selaku pembimbing dan pemilik proyek FMCS.
3. Bapak Ir. Arief Syaichu Rohman, MEngSc, Ph.D., selaku dosen pembimbing penulis yang telah memberikan bimbingan, saran dan masukannya kepada penulis.
4. Bapak Arif Sasongko, S.T., M.Sc.,Ph.D., selaku dosen pembimbing sekaligus ketua program studi Teknik Elektro ITB beserta Tim Tugas Akhir Teknik Elektro ITB yang telah memberikan arahan selama pengerjaan tugas akhir.
5. Bapak Firdaus, Bapak Ahmad Husnan, Mas Ilmi dan Mas Wahid selaku pembimbing selama pengerjaan di PT. LEN Industri.
6. Shah Dehan Lazuardi dan Aulia Hening Darmasti selaku rekan satu kelompok pengerjaan tugas akhir yang telah bersedia bekerjasama dengan penulis selama kegiatan tugas akhir ini dilakukan.

7. Teman-teman di ruang riset mandiri, Fikri, Dinan, Duhan, Iftikar, Ridhan, Yusri, Kevin Irawan, Syifa, Billie, Rama, Fadhil, Irham dan Audi yang telah bersama-sama melaksanakan kegiatan tugas akhir selama ini.
8. Teman-teman di lab komputer, Kevin, Deo, Bramantio, Hasbi, Brian, Saqfi, Aldo dan Rendra yang telah bersama-sama melaksanakan kegiatan tugas akhir selama ini.
9. Teman-teman di Teknik Elektro ITB yang telah menempuh kegiatan perkuliahan bersama-sama hingga saat ini.
10. Semua pihak yang tidak bisa disebutkan satu-persatu yang telah membantu kami dalam penyelesaian laporan 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, 12 Mei 2017

Penulis

DAFTAR ISI

DAFTAR GAMBAR

BAB I

PENDAHULUAN

Bagian ini menjelaskan latar belakang, rumusan masalah, tujuan tugas akhir, lingkup tugas akhir, metodologi penulisan dan sistematika penulisan.

1.1 Latar Belakang

Kemacetan merupakan salah satu permasalahan yang semakin pelik di Kota Bandung. Permasalahan ini diakibatkan oleh jumlah kendaraan yang turun ke jalan raya sangat banyak. Akibat yang ditimbulkan oleh permasalahan kemacetan ini bukan hanya ketidaknyamanan, tetapi juga kerugian materi. Pada tahun 2014, kerugian yang dialami akibat kemacetan di Kota Bandung mencapai 4.36 triliun per tahun. Kerugian ini berasal dari bahan bakar minyak (BBM) yang terbuang sia-sia selama kendaraan berhenti.

Salah satu solusi yang dapat ditawarkan yaitu penggunaan *guided bus* listrik sebagai transportasi umum massal di Kota Bandung. *Guided bus* merupakan transportasi umum massal berupa bus yang bergerak pada jalur khusus yang hanya dapat dilalui oleh *guided bus* sehingga bebas dari kemacetan. Bus ini memiliki kapasitas yang jauh lebih besar daripada transportasi umum massal yang biasa beroperasi di Kota Bandung. Karena menggunakan listrik sebagai sumber energi, *guided bus* lebih hemat energi dibandingkan dengan kendaraan yang menggunakan BBM.

Untuk mengimplementasikan *guided bus* yang dapat menyelesaikan permasalahan-permasalahan yang telah dijelaskan sebelumnya, diperlukan suatu sistem untuk mengatur seluruh armada *guided bus*. Pengaturan tersebut diatur oleh *control station* dan meliputi penjadwalan dan pengecekan kondisi armada. Oleh karena itu, diperlukan sebuah perangkat yang dapat menghubungkan armada dengan *control station* untuk keperluan pengaturan tersebut.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan pada bagian sebelumnya, rumusan masalah dalam tugas akhir ini yaitu sebagai berikut.

- Bagaimana cara mengakuisisi data dari ECU yang ada pada *guided bus*?
- Bagaimana cara mengambil data posisi armada *guided bus*?

- Bagaimana cara mengirimkan data-data yang telah diperoleh ke server?
- Bagaimana cara menerima dan menampilkan data dari server?

1.3 Tujuan Tugas Akhir

Tujuan dari tugas akhir ini adalah sebagai berikut.

- Merancang dan implementasi perangkat keras (subsistem elektronik) untuk akuisisi data posisi, baterai, RPM dan fault yang akan dipasang pada tiap armada *guided bus*.
- Merancang dan implementasi komunikasi antara subsistem elektronik dengan ECU lain pada *guided bus*.
- Merancang dan implementasi komunikasi antara subsistem elektronik dengan server.

1.4 Lingkup Permasalahan

Tugas akhir ini dibuat dengan asumsi dan batasan sebagai berikut.

- Subsistem elektronik yang dibuat berupa prototipe.
- Pada pengujian, digunakan kendaraan bermotor biasa untuk melakukan pengujian data posisi dan mini AGT pada PT. LEN Industri untuk melakukan pengujian data dari CAN Bus.
- Implementasi subsistem elektronik yang dibuat hanya satu buah.
- Digunakan kartu SIM dari Indosat untuk melakukan komunikasi internet antara subsistem elektronik dengan GUI.

1.5 Metodologi Penulisan

Seluruh dokumentasi pada buku tugas akhir ini merupakan seluruh data-data kegiatan selama pengerjaan tugas akhir. Metode yang digunakan untuk penyelesaian tugas akhir ini adalah sebagai berikut.

- Penentuan Topik, yaitu penentuan topik dilakukan dengan memilih topik yang telah ditentukan oleh tim Tugas Akhir berdasarkan pilihan penulis.
- Penentuan Spesifikasi, yaitu tahap menentukan spesifikasi dari keluaran yang diharapkan.
- Studi Literatur, yaitu mempelajari desain dari sistem dan algoritma yang sudah ada sebagai dasar perancangan untuk memenuhi spesifikasi yang ingin dicapai.

- Perancangan dan Implementasi Sistem, yaitu analisis yang akan dibuat dirancang memperhatikan masukan, kebutuhan untuk melakukan proses, dan keluaran yang diharapkan. Kemudian analisis diimplementasikan menggunakan bahasa pemrograman yang sesuai.
- Pengujian dan Perbaikan Sistem, yaitu pengujian sistem dilakukan untuk melihat performansi dan kinerja dari sistem dan melakukan perbaikan terhadap *failure/bug* yang dihasilkan.
- Penarikan Kesimpulan, yaitu analisis berhasil atau tidaknya hasil pengujian sistem dan ditarik kesimpulannya.

1.6 Sistematika Penulisan

Penulis membagi buku tugas akhir ini menjadi lima bab, yaitu sebagai berikut.

- **BAB I PENDAHULUAN**

  Penulis mengemukakan latar belakang, rumusan masalah, tujuan tugas akhir, lingkup permasalahan, metodologi yang digunakan dalam pengerjaan tugas akhir, dan sistematika penulisan.
- **BAB II TINJAUAN PUSTAKA**

  Penulis menguraikan beberapa teori yang mendukung pengerjaan tugas akhir ini. Teori-teori tersebut berkaitan dengan perancangan analisis perilaku mengemudi supir serta aplikasi yang digunakan untuk mengimplementasikan rancangan tersebut.
- **BAB III SPESIFIKASI DAN PERANCANGAN**

  Penulis mengemukakan spesifikasi dan desain sistem.
- **BAB IV IMPLEMENTASI**

  Penulis memaparkan metode dan hasil implementasi sistem.
- **BAB V HASIL PENGUJIAN**

  Penulis melaporkan hasil verifikasi terhadap spesifikasi yang telah dibuat sebelumnya.
- **BAB VI KESIMPULAN**

  Penulis mengutarakan kesimpulan dari pengerjaan tugas akhir ini dan saran untuk para pembaca maupun pengembang topik tahap selanjutnya

BAB II

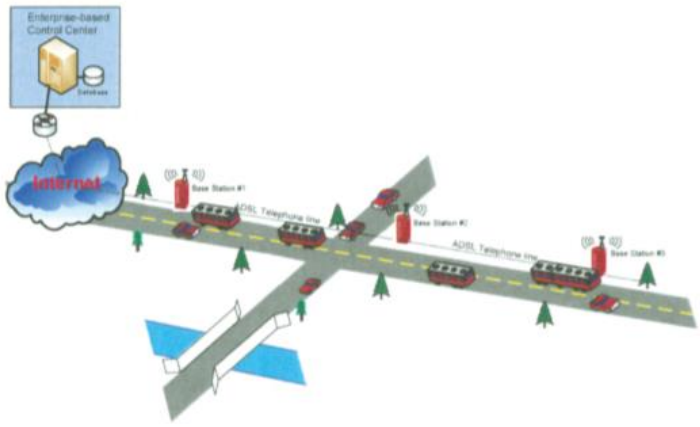
PEMANTAUAN DAN MANAJEMEN KENDARAAN LISTRIK

Pada bagian ini akan dijelaskan tinjauan pustaka untuk dapat mengimplementasikan subsistem elektronik. Tinjauan pustaka ini terdiri dari studi literatur makalah, protokol MQTT, CAN Bus dan NMEA sentences. Protokol MQTT digunakan Karena server yang digunakan pada FMCS ini adalah server MQTT. CAN Bus merupakan bus standar yang digunakan pada kendaraan listrik, sehingga *guided bus* juga menggunakan CAN Bus sebagai protokol komunikasi antar mesinnya. NMEA *sentences* adalah data yang dihasilkan oleh sensor posisi GPS yang berisi data posisi, ketinggian, waktu, dan lain-lain. Untuk keperluan FMCS, data yang akan diambil dari GPS adalah data posisi.

2.1 Pemantauan Posisi Armada Kendaraan

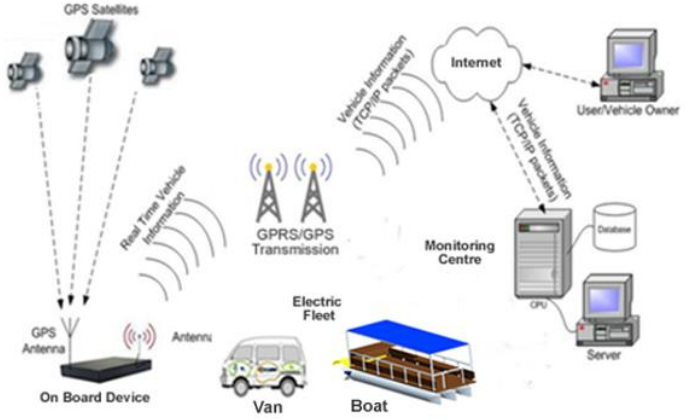
Kegiatan pemantauan posisi armada kendaraan dapat dilakukan dengan beberapa metode. Contoh metode yang dapat digunakan yaitu dengan menggunakan GPS dan RFID.

Pada penggunaan RFID, digunakan sebuah tag dalam kendaraan dan reader di sekitar jalan raya. Kelebihan metode ini yaitu sangat presisi. Namun, metode ini memiliki akurasi yang buruk dan membutuhkan dukungan infrastruktur untuk reader di sekitar jalan raya, dan pada kondisi jalan yang ramai, performa frekuensi radio 433 MHz mudah terinterferensi sehingga seringkali terjadi kegagalan dalam komunikasi tag dan reader.



Gambar 2. 1 Arsitektur pemantauan posisi dengan RFID

Pada penggunaan GPS, digunakan sebuah GPS receiver dan antenna. Kelebihan metode ini yaitu lebih akurat dari metode RFID. Selain itu, metode ini mudah diimplementasikan Karena tidak membutuhkan infrastruktur tambahan. Namun, modul GPS memiliki presisi yang lebih buruk dari RFID dan membutuhkan waktu sebelum dapat membaca data posisi sehingga armada harus menunggu sampai GPS siap digunakan sebelum berangkat.

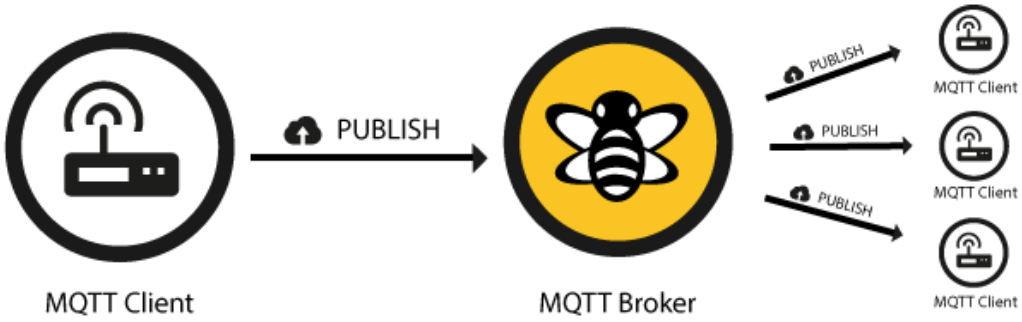


Gambar 2. 2 Arsitektur pemantauan posisi dengan GPS

2.2 *Message Queuing Telemetry Transport* (MQTT)

MQTT diciptakan pada tahun 1999 oleh Dr Andy Stanford-Clark dari IBM dan Arlen Nipper dari Arcom (sekarang Eurotech). MQTT adalah standar ISO (ISO/IEC PRF 20922) sebagai protokol pengiriman pesan yang ringan berbasis metode publish/subscribe, didesain untuk perangkat dengan komunikasi bandwidth rendah, latensi tinggi atau berada di dalam jaringan yang tidak handal. Protokol ini cocok untuk komunikasi machine to machine atau Internet of Things.

Protokol MQTT menggunakan metode *publish/subscribe*. Klien yang menghasilkan dan mengirimkan informasi tertentu disebut *publisher*. Klien yang mendapatkan informasi tersebut disebut *subscriber*. Selain *publisher* dan *subscriber* ada juga *broker* yang menjamin *subscriber* mendapatkan info yang diinginkan dari *publisher*. MQTT *broker* bertanggung jawab untuk mendistribusikan pesan antar klien berdasarkan topik dari pesan tersebut. Keuntungan dari sistem *publish/subscribe* adalah *space decoupling* dan *time decoupling*, dimana pengirim dan penerima tidak saling mengetahui karena ada *broker* diantara mereka dan tidak perlu terkoneksi secara bersamaan (Kapitu, 2015).



Gambar 2. 3 Aliran data MQTT

2.3 *Global Positioning System* (GPS) dan *National Marine Electronics Association* (NMEA) *Sentence*

NMEA membentuk spesifikasi standar untuk komunikasi antara mesin-mesin elektrik yang biasa dipakai di kelautan. Pada implementasinya, modul GPS menggunakan standar NMEA untuk mengomunikasikan data GPS yang terbaca, seperti kecepatan, posisi dan waktu. NMEA mengirimkan beberapa "kalimat" (sentence) berupa string yang berisi data-data tertentu sesuai dengan bagiannya masing-masing (berbeda untuk setiap sentence tersebut). Terdapat beberapa jenis sentence dan dikodekan dengan tiga buah huruf, dan dua buah huruf lain sebagai prefix yang mendefinisikan asal data tersebut (untuk GPS, prefix-nya adalah GP).

Setiap sentence diawali dengan karakter '$' dan diakhiri dengan karakter '\r\n'. Setiap data pada sebuah sentence dipisahkan dengan sebuah karakter ','. Sebuah sentence terdiri dari maksimal 80 karakter.

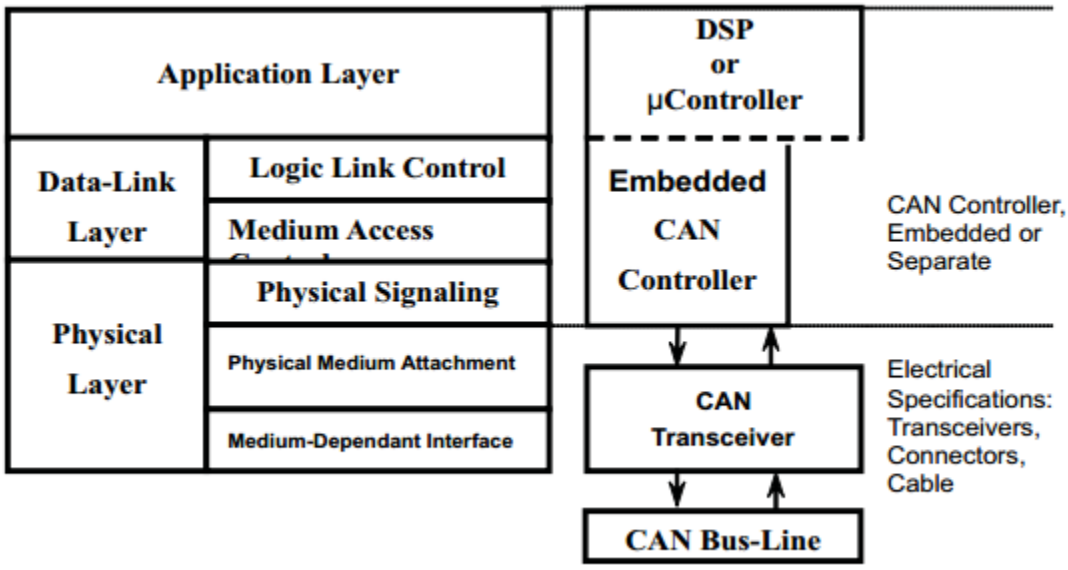
Tabel 2. 1 Jenis NMEA sentence

Message	Description
GGA	Time, position and fix type data
GLL	Latitude, longitude, UTC time of position fix and status
GSA	GPS receiver operating mode, satellites used in the position solution, and DOP values
GSV	Number of GPS satellites in view satellite ID numbers, elevation, azimuth, & SNR values
MSS	Signal-to-noise ratio, signal strength, frequency, and bit rate from a radio-beacon receiver
RMC	Time, date, position, course and speed data
VTG	Course and speed information relative to the ground
ZDA	PPS timing message (synchronized to PPS)
150	OK to send message
151	GPS Data and Extended Ephemeris Mask
152	Extended Ephemeris Integrity
154	Extended Ephemeris ACK

2.4 Controller Area Network (CAN)

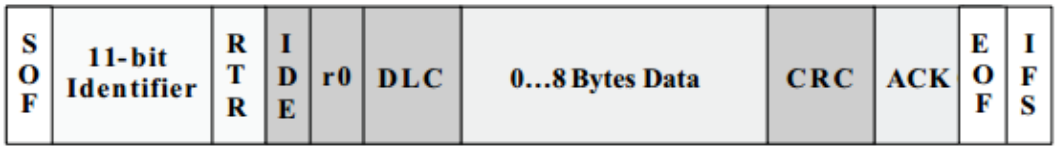
CAN Bus pertama kali dikembangkan oleh BOSCH sebagai *multi-master message broadcast system* yang dapat mencapai kecepatan 1 megabit per detik (1 Mbps). Dalam jaringan CAN, pesan-pesan singkat seperti data kecepatan dan temperature disebarkan ke seluruh jaringan. CAN juga merupakan sistem jaringan yang dapat meminimalisir kesalahan akibat *noise*.

Protokol komunikasi CAN dijelaskan dalam standar internasional ISO-11898. Dalam standar ini komunikasi CAN dijelaskan dalam *Open System Interconnection* (OSI) *model* dan *layer-layer*.



Gambar 2. 4 Arsitektur CAN pada ISO 11898

Pesan pada CAN Bus terdiri dari susunan 11-bit yang standar seperti pada gambar berikut.

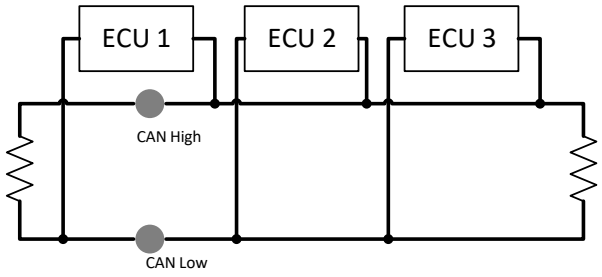


Gambar 2. 5 11-bit Identifier CAN

- SOF merupakan penanda awal pesan. SOF digunakan untuk singkronisasi node-node pada seluruh jaringan CAN.
- Identifier digunakan untuk menentukan prioritas pesan yang dikirimkan.

- *Remote Transmission Request* (RTR) digunakan saat sebuah informasi diperlukan oleh node lain pada jaringan.
- IDE menandakan identifier yang digunakan adalah standar, tanpa *extension*.
- r0 merupakan *reserved bit* yang sekarang belum dipakai, tetapi ada kemungkinan dipakai untuk fitur lain di masa depan.
- *Data Length Code* (DLC) berisi jumlah byte data yang dikirim.
- Data yaitu isi informasi yang akan dikirimkan.
- *Cyclic Redundancy Check* (CRC) berfungsi sebagai *checksum* jumlah data yang dikirimkan.
- ACK bernilai sesuai dengan kevalidan informasi yang diterima oleh node-node. Jika sebuah node memberikan nilai ACK yang tidak sesuai, pesan tersebut akan dibuang, kemudian node yang mengirimkan data tersebut akan mengirimkan ulang datanya.
- *End of Frame* (EOF) menandakan akhir *frame* CAN.
- *Interframe Space* (IFS) berisi waktu yang diperlukan oleh kontroller untuk mengirimkan informasi.

CAN menjadi standar jaringan yang digunakan untuk komunikasi dalam kendaraan elektrik. Konfigurasi CAN Bus secara garis besar adalah seperti gambar berikut.



Gambar 2. 6 CAN Bus

CAN Bus terdiri dari dua buah data yaitu CAN High dan CAN Low, menggunakan kawat *twisted differential* dan dua buah resistor 120 Ω pada kedua ujungnya.

BAB III

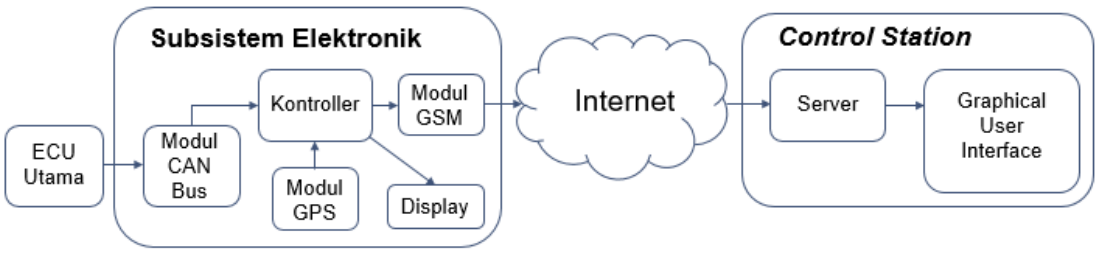
SPESIFIKASI, DESAIN DAN IMPLEMENTASI

Pada bagian ini akan dijelaskan mengenai FMCS pada *guided bus* secara keseluruhan. Kemudian dijelaskan juga spesifikasi, aliran data dan *flowchart* program subsistem elektronik.

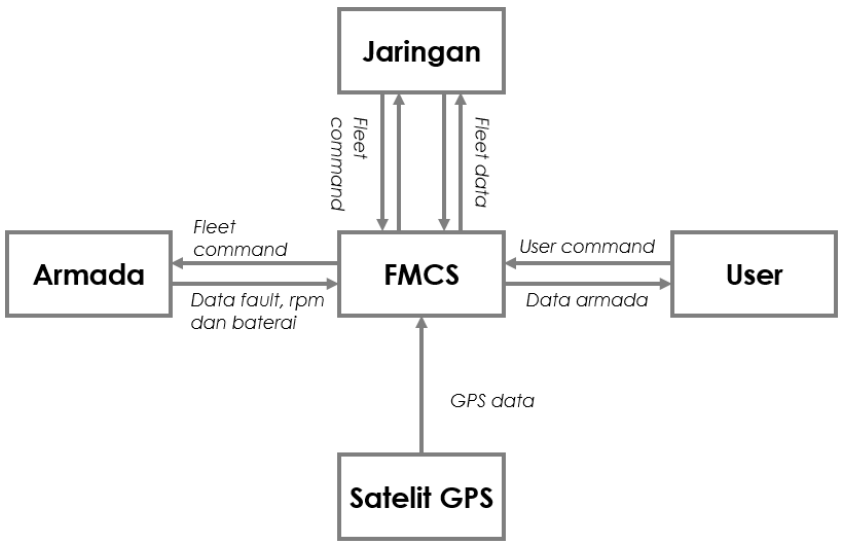
Desain dan implementasi yang akan dijelaskan pada bagian ini adalah implementasi prototipe subsistem elektronik pada FMCS pada *guided bus*. Implementasi ini meliputi prosesor, akuisisi data dari CAN Bus, akuisisi data posisi, pengiriman data ke server dan penerimaan data dari server.

3.1 FMCS pada *Guided Bus*

FMCS pada *guided bus* merupakan sistem komunikasi antara tiap armada dengan sebuah *control station* untuk melakukan penjadwalan dan pemantauan kondisi tiap armada. Arsitektur FMCS yang dibuat dan data flow diagram FMCS yaitu sebagai berikut.



Gambar 3. 1 Arsitektur FMCS



Gambar 3. 2 Data flow diagram FMCS level 0

Data yang mengalir dalam FMCS yaitu kondisi armada, posisi armada dan perintah dari *control station* untuk armada. Data kondisi dan posisi armada berasal dari sensor-sensor yang ada pada *guided bus* dan subsistem elektronik FMCS. Data-data ini kemudian disusun dan dikirim ke GUI pada *control station* melalui server dan ditampilkan. Setelah itu, GUI akan merespon dengan mengirim perintah ke tiap armada berupa kecepatan. Perintah ini bergantung pada masukan data dan algoritma penjadwalan yang sedang berjalan. Subsistem elektronik kemudian menampilkan perintah tersebut untuk dilihat oleh pengemudi *guided bus*.

3.2 Definisi, Fungsi dan Spesifikasi Subsistem Elektronik

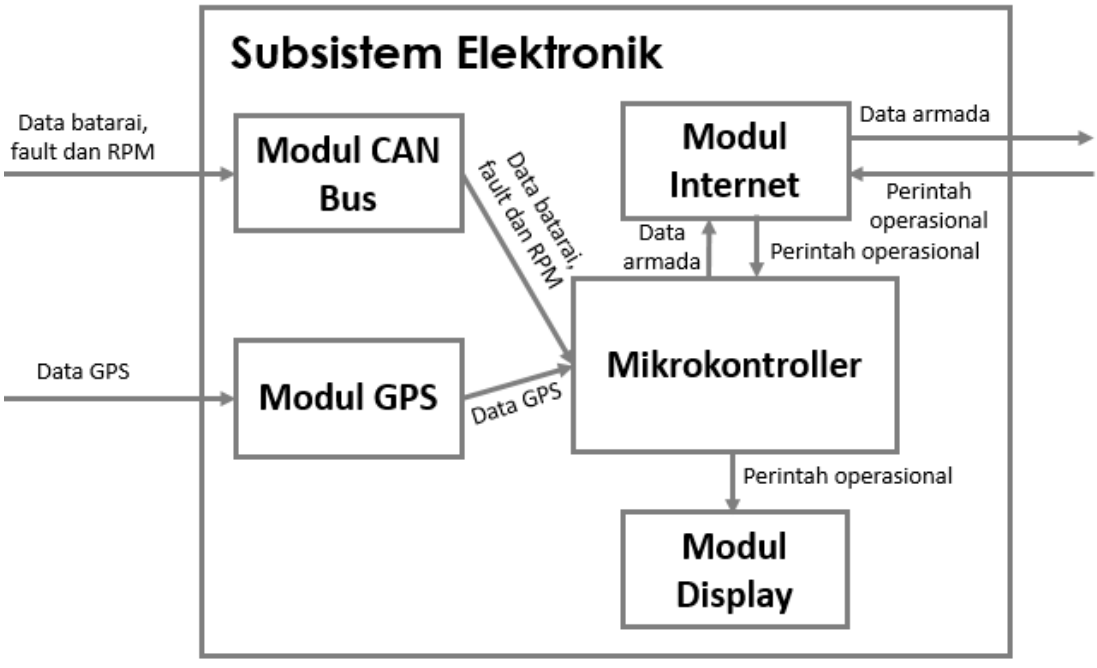
Subsistem elektronik pada FMCS merupakan perangkat keras yang diletakkan di setiap armada *guided bus* dan dihubungkan dengan ECU utama pada *guided bus* untuk berkomunikasi di jaringan CAN Bus. Subsistem elektronik bertugas untuk mengakuisisi data pada CAN Bus, mengakuisisi data posisi armada dan mengirimkan data tersebut ke server melalui jaringan internet GSM. Selain itu, subsistem elektronik juga menerima dan menampilkan data perintah dari *control station* berupa kecepatan.
Untuk memperoleh spesifikasi *error* sensor posisi pada FMCS, tampilan jalur pada GUI disegmentasi tiap 12 meter sesuai dengan panjang bus. Dari panjang segmen ini, dapat ditentukan *error* maksimal sensor posisi yaitu 6 meter, supaya tidak ada 2 armada yang ditampilkan dalam 1 segmen yang sama.
Untuk meperoleh spesifikasi waktu pengiriman data, digunakan asumsi kecepatan armada pada jalur khusus yaitu pada 15 meter/detik. Karena segmentasi yang dibuat sebesar 12 meter, maka waktu pengiriman data harus memperhatikan nilai tersebut supaya bus tetap terpantau setiap melewati segmen-segmen. Sehingga, data harus dikirimkan setiap 12/15 detik, yaitu 0.8 detik.
Berikut spesifikasi FMCS yang terkait dengan subsistem elektronik.

Tabel 3. 1 Fungsi dan spesifikasi subsistem elektronik

Fungsi	Spesifikasi
Memantau posisi tiap armada.	Posisi yang dibaca oleh operator pada *control station* memiliki *error* maksimal sebesar 6 meter.
Memantau rpm kendaraan, kondisi fault, dan level energi baterai.	Dapat membaca data rpm kendaraan, kondisi fault, dan level energi baterai

	sesuai spesifikasi ECU yang sudah terdapat di guided bus.
Mengirimkan data ke *control station* melalui server.	Data yang dikirimkan adalah: • Posisi • rpm kendaraan • Kondisi fault • Level energi baterai Data yang dikirim harus dapat dibaca pada GUI di *control* station maksimal setiap 0.8 detik.
Mengirimkan perintah dari *control station* untuk pengemudi melalui server.	*Control station* dapat mengirimkan perintah operasional ke armada setiap terjadi perubahan perintah.
Menunjukkan kondisi armada (normal atau keadaan emergency) kepada *control station*.	Subsistem elektronik dapat mengirimkan tanda ke *control station* saat armada tidak dapat beroperasi.

3.3 Data Flow Diagram



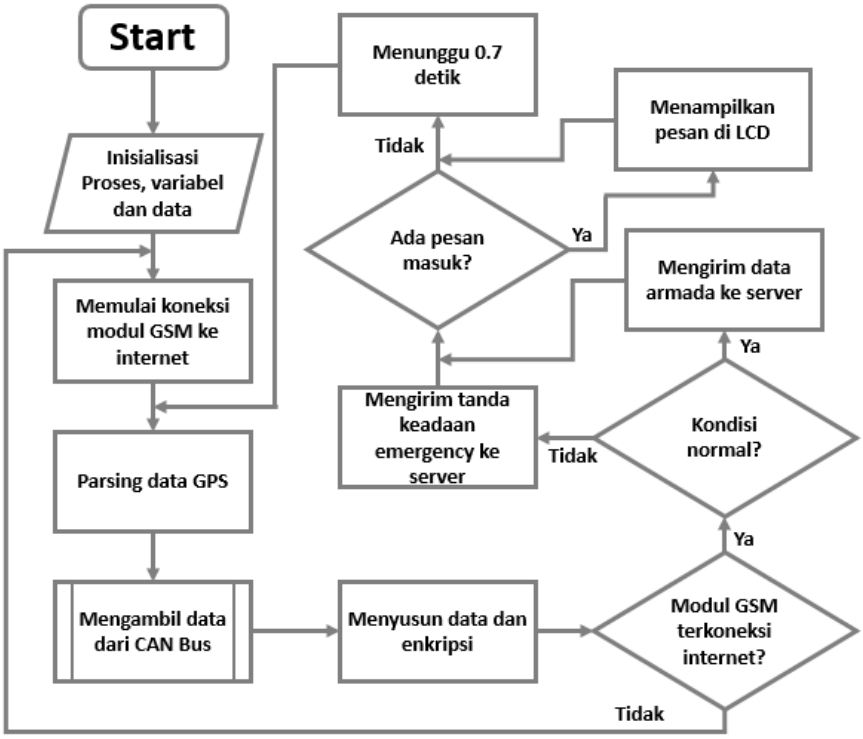
Gambar 3. 3 Data flow diagram subsistem elektronik

Gambar 3.2 menunjukkan *data flow diagram* dari subsistem elektronik. Secara garis besar, subsistem elektronik menerima data-data dari sensor, kemudian dikirimkan ke server. Terdapat dua buah data yang masuk ke subsistem elektronik, yaitu data baterai

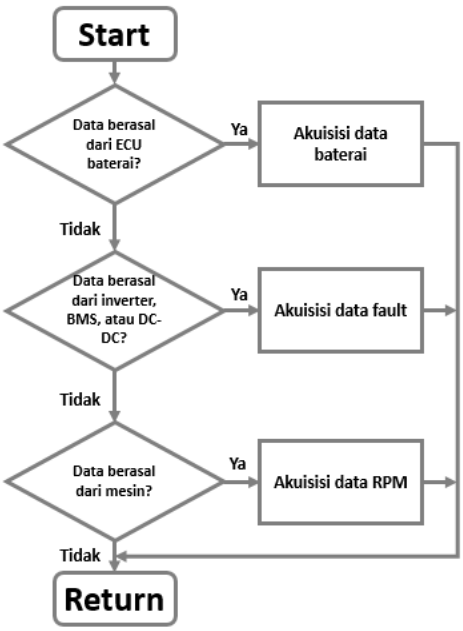
dan GPS. Kedua data ini kemudian diproses dalam mikrokontroller untuk kemudian dijadikan sebuah kesatuan data armada. Setelah itu, data armada diarahkan ke modul komunikasi untuk diteruskan ke server sebagai *fleet data*. Selain itu, subsistem elektronik juga menerima data dari server berupa *fleet command*. Data tersebut pertama masuk ke modul komunikasi, kemudian diarahkan ke mikrokontroller untuk kemudian diproses untuk ditampilkan di modul *display*.

3.4 *Flowchart* Program

Pada prosesor akan dijalankan sebuah program untuk tugas-tugas sesuai fungsi subsistem elektronik, yaitu mengakuisisi data posisi, baterai, RPM dan fault pada armada, dan mengirimkannya ke server. Selain itu juga menampilkan data yang dikirimkan dari GUI melalui server. Berikut flowchart program yang digunakan.



Gambar 3. 4 Flowchart program utama



Gambar 3. 5 Flowchart sub-program akuisisi data dari CAN Bus

3.5 Desain dan Implementasi

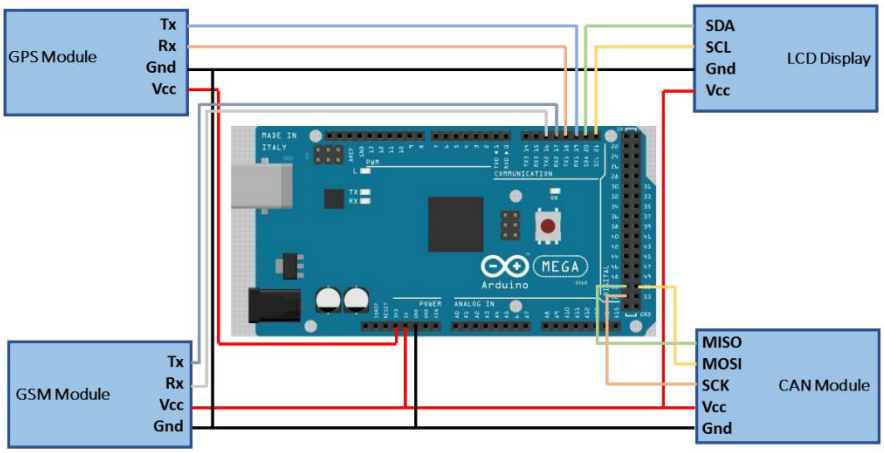
Desain dan implementasi subsistem elektronik dibagi menjadi 8 bagian, yaitu kontroller, Akuisisi Data Armada melalui CAN Bus, komunikasi Subsistem Elektronik dengan GUI melalui Internet, Akuisisi Data Posisi melalui GPS, Penentuan Kondisi Normal dan Emergency, packaging, display dan PCB shield.

3.5.1 Kontroller

Untuk menjalankan program seperti flowchart di atas, diperlukan sebuah mesin sekuensial. Karena tidak ada proses berat seperti pengolahan citra, tidak diperlukan prosesor dengan kecepatan tinggi sehingga cukup digunakan prosesor berbasis AVR. Pada perancangan ini, dipilih Arduino Mega berbasis mikroprosesor ATMega2560 dengan *clock* 16 MHz sebagai kontroller. Arduino Mega 2560 memiliki spesifikasi teknis yang memenuhi spesfikasi desain yang diperlukan, antara lain jumlah UART (4 buah), memori yang cukup (8 KB SRAM, 4 KB EEPROM, 256 KB FLASH) dan kecepatan proses Arduino Mega 2560 (16 MHz) yang sudah mencukupi karena tidak ada proses berat seperti pengolahan citra pada subsistem elektronik ini.

Pada implementasinya, digunakan Arduino Mega 2560 yang sudah dijual di pasaran (bukan *custom* PCB dan *bootload* sendiri) agar kompatibel dengan modul-modul yang ada. Perangkat lunak yang digunakan adalah Arduino IDE untuk mengedit dan mengunggah program ke Arduino Mega 2560.

Untuk selanjutnya, seluruh modul pada subsistem elektronik akan diuji dan dikendalikan menggunakan kontroller ini. Modul-modul yang digunakan akan memanfaatkan komunikasi SPI, I^2C dan serial pada Arduino.



Gambar 3. 6 Skematik implementasi subsistem elektronik

Gambar 3.6 menunjukkan skematik implementasi integrasi subsistem elektronik. Keempat sub-modul dihubungkan ke pin-pin tertentu pada kontroller Arduino Mega. Sub-modul GSM dan GPS menggunanakan komunikasi serial. Untuk diimplementasikan ke Arduino Mega, kedua modul ini menggunakan pin serial yang berbeda pada Arduino Mega, yaitu pin Serial0 untuk modul GSM dan pin Serial1 untuk modul GPS. Sub-modul CAN menggunakan komunikasi SPI. Pada implementasinya, SPI terdiri dari beberapa bagian yang ada pada pin-pin tertentu, yaitu MISO (pin 50), MOSI (pin 51) dan SCK (pin 52). LCD Display menggunakan komunikasi I^2C. Pada implementasinya, komunikasi I^2C pada Arduino Mega menggunakan dua pin yaitu SDA dan SCL.

3.5.2 Akuisisi Data Armada melalui CAN Bus

Untuk berkomunikasi dengan ECU lain pada *guided bus*, dibutuhkan sebuah komponen komunikasi CAN Bus. Salah satu komponen yang dapat melakukan fungsi ini adalah MCP2515. Berikut spesifikasi MCP2515.

Tabel 3. 2 Spesfikasi MCP2515

Parameter	Nilai
Version Supported	2.0B Active

Tx Buffers	3
Rx Buffers	2
Speed	Up to 1000 kbps
Interrupt Output	1
Filters	6
Temperature Range	-40 to +125 ^{o}C
Operating Voltage Range	2.7 to 5.5 V
Communication	Serial, SPI

Dari tabel spesifikasi di atas, MCP2515 memiliki kecepatan data 1000 kbps. Nilai ini sesuai dengan kecepatan CAN Bus yang ada di PT LEN Industri. Selain itu, *operating voltage* dan komunikasi dengan kontrollernya sesuai dengan Arduino, yaitu 2.7-5.5 volt dan komunikasi SPI.

Untuk implementasi ini, digunakan MCP2515 sebagai chip untuk komunikasi subsistem elektronik dengan CAN Bus pada fleet. Chip ini dapat berkomunikasi dengan Arduino melalui komunikasi SPI. Untuk metode implementasinya, digunakan CAN Bus shield dari DFRobot.

Modul CAN ini dapat menggunakan komunikasi serial maupun SPI. Pada implementasi subsistem elektronik ini, digunakan komunikasi SPI.

Pada implementasinya, modul CAN ini akan digunakan untuk membaca data yang dibutuhkan saja dari CAN Bus, tidak semua data yang akan diolah. Berikut daftar CAN ID dan data yang ada pada ID tersebut.

ID	Equipment	Remarks								
		Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	
1	INVERTER	ID 1	102 (FAILUREIW) 103 (FAILUREVDC) 104(FAILUREZERODE	101(FAILUREIU) 102 (FAILUREIW) 103 (FAILUREVDC) 104 (FAILUREZERODEG)	0	0	0	0	0	
	BMS	ID 2	FAULT_REG	0	0	0	0	0	0	
	DC-DC	ID 3	1: Input UV LIRE 2: Input UV FON 3: Input OV LIRE 4: Input OV NAON 5: Output UV LIRE 6: Output UV NAON 7: Output OV LIRE 8: Output OV NAON 9: Output OC LIRE 10: Output OC NAON 11: Powermodule NOK LIRE 12: Powermodule NOK NAON 13: Commu Lost LINORE	0	0	0	0	0	0	
	BCU	ID 4								

2	ECU to Inv	Mode 0: SensorInit 2: ZeroDeg 1: Neutral 4: Forward 8: Reverse	Throttle 0 - 100	Brake 0 - 100	0	0	0	0	0	
3	Inv to ECU	Mode Status 1: ZeroDeg 2: Neutral 3: Forward 4: Reverse	Applied Throttle 0 - 100	Applied Brake 0 - 100	RPM Byte0 (signed int) (Data)	RPM Byte1 (signed int) (Data)	Motor Temp (Data)	IGBT Temp (Data)	0	
4	ECU to BMS	Mode 1: Close Contactor 0: Open Contactor	Mode 1. Charge 0. Idle	0	0	0	0	0	0	
5	BMS to ECU	1	V_BAT Byte0	V_BAT Byte 1	I_BAT Byte 0	I_BAT Byte 1	V_HVDC Byte 0	V_HVDC Byte 1	0	
		2	SOC	g	BMS_STATE	0	0	0	0	
16	ECU to DC-DC 1	Status 1: Comm OK	0	0	0	0	0	0	0	
26	ECU to DC-DC 2	Status 1: Comm OK	0	0	0	0	0	0	0	
7	DC-DC 1 to ECU	Status 1: OK 2: Auto Restart	1	0	0	0	0	0	0	
7	DC-DC 2 to ECU	Status 1: OK 2: Auto Restart	2	0	0	0	0	0	0	
8	ECU to BCU									
9	BCU to ECU									

Gambar 3. 7 Isi data CAN mini AGT LEN

```
/* deklarasi */
#include <SPI.h>
#include "mcp_can.h"
const int SPI_CS_PIN = 10;
MCP_CAN CAN(SPI_CS_PIN);

/* setup */
while (CAN_OK != CAN.begin(CAN_1000KBPS))
  {
    lcd.setCursor(0, 2);
    lcd.print("CAN INIT FAILED");
    delay(100);
  }
  lcd.setCursor(0, 2);
  lcd.print("CAN INIT OK");

/* program utama */
  data[25] = (byte)':';
  if(CAN_MSGAVAIL == CAN.checkReceive())
  {
    CAN.readMsgBuf(&len, buf);
    unsigned char canId = CAN.getCanId();
    lcd.setCursor(11, 5);
    lcd.print(canId);

    /* data baterai */
    if (canId == 5)
    {
      if (buf[0] == 1)
      {
        data[23] = (byte)buf[2];
        data[24] = (byte)buf[1];
      }
    }
    /* data fault */
    if (canId == 1)
    {
      data[26] = (byte)buf[0];
      data[27] = (byte)buf[1];
      data[28] = (byte)buf[2];
    }
    /* data rpm */
    else if (canId == 3)
    {
      data[30] = (byte)buf[3];
      data[29] = (byte)buf[4];
    }
  }
```

Kode di atas merupakan implementasi dari modul CAN Bus pada Arduino Mega 2560. Isi dari fungsi-fungsi yang digunakan terlampir.

Pada bagian awal program dideklarasikan library yang dibutuhkan dengan fungsi #include. Library yang dibutuhkan yaitu untuk komunikasi SPI dan library CAN Bus yang berisi fungsi-fungsi untuk membaca data dari CAN Bus. Konstanta SPI_CS_PIN berisi pin Arduino yang terkoneksi untuk komunikasi SPI, bergantung pada jenis CAN Bus yang digunakan, dalam implementasi ini pin 10. MCP_CAN CAN(SPI_CS_PIN) merupakan deklarasi class CAN yang akan digunakan pada program utama.

Pada bagian setup, while (CAN_OK!=CAN.begin(CAN_1000KBPS)) berfungsi untuk memastikan modul CAN Bus dapat digunakan, dan memulai komunikasi dengan CAN Bus lain dengan baudrate 1000kbps. Fungsi readMsgBuf digunakan untuk membaca setiap data yang masuk ke modul CAN Bus. Fungsi getCanId digunakan untuk mengetahui ID dari pesan yang diperoleh modul CAN Bus. Data yang diperlukan dari CAN Bus terletak pada id=1 dan pada byte ke-3 hingga ke-6, sehingga variabel data (dijelaskan pada bagian persiapan data) diisikan dengan data-data tersebut.

3.5.3 Komunikasi Subsistem Elektronik dengan GUI melalui Internet

Komunikasi antara subsistem elektronik dengan GUI ini akan dilakukan melalui jaringan internet GSM. Jaringan GSM dipilih karena armada akan terus bergerak sehingga diperlukan koneksi internet yang dapat bergerak pula.

Perangkat keras yang akan digunakan untuk implementasi modul internet GSM ini adalah modul GSM SIM900. Modul GSM ini memiliki fitur-fitur yang mencukupi untuk digunakan pada sub-sistem ini, yaitu frekuensi kerja 2G-3G (Quad-Band 850/900/1800/1900 MHz), kebutuhan daya yang cukup kecil (1 hingga 2 Watt) dan dapat dikendalikan melalui komunikasi serial dengan AT *Command*.

Tabel 3. 3 Spesifikasi modul komunikasi

Parameter	Nilai
Frekuensi Kerja	Quad-Band 850/900/1800/1900 MHz
Power Consumption	Class 4 (2 W @850/900 MHz) Class 1 (1 W @1800/1900 MHz)

Supply Voltage	3.2-4.8 Volts
Command	GSM 07.07, 07.05 and SIMCOM enhanced AT Commands
Operation Temperature	-40 to 85 °C
Data Transfer	GPRS class 10: max. 85.6 kbps (downlink) CSD USSD

Pada keadaan sinyal paling buruk, data rate dari SIM900 dapat mencapai hingga 9.05 kbps. Karena data yang dikirim berjumlah 32 byte, maka dapat dihitung waktu pengiriman data ini menggunakan SIM900 yaitu sekitar 19.1 ms, masih dapat ditoleransi.

Data yang akan dikirimkan memiliki susunan seperti berikut.

10 byte		**11 byte**		**2 byte**		**2 byte**	**3 byte**
longitude	**:**	latitude	**:**	Data baterai	**:**	RPM	Data fault

Implementasi modul internet ini menggunakan *shield* SIM900 dari EFCom. Kelebihan dari *shield* ini dibandingkan modul SIM900 *standalone* adalah manajemen *power* yang lebih baik, *built-in* antenna dan penggunaannya yang mudah. Kelebihan *shield* ini dibandingkan dengan *shield* SIM900 lain adalah fleksibilitas untuk menggunakan pin lain pada Arduino untuk komunikasi *software serial* sehingga *debugging* dan *troubleshooting* dapat dilakukan dengan lebih mudah.

Untuk menggunakan modul ini dengan Arduino Mega 2560, Serial pada Arduino Mega dihubungkan dengan Rx dan Tx modul GSM. Berikut implementasi kode pada Arduino.

```
/* setup */
MQTT.begin();

/* deklarasi */
#include "GSM_MQTT.h"
#include <SoftwareSerial.h>
String MQTT_HOST = "broker.hivemq.com";
String MQTT_PORT = "1883";
bool mqttReady = false;
SoftwareSerial cobac(3, 4);
void GSM_MQTT::AutoConnect(void)
{
  connect("TA16170194", 0, 0, "", "", 1, 0, 0, 0, "", "");
}
void GSM_MQTT::OnConnect(void)
{
  subscribe(0, _generateMessageID(), "fleet1sub", 1);
```

```
  publish(1, 1, 0, _generateMessageID(), "fleet1", "connected");
}
void GSM_MQTT::OnMessage(char *Topic, int TopicLength, char *Message, int
MessageLength)
{
  lcd.setCursor(11, 3);
  lcd.print(Message);
}
GSM_MQTT MQTT(20);

/* program utama */
  if (MQTT.available())
  {
    mqttReady = true;
    MQTT.publish(1, 1, 0, MQTT._generateMessageID(), "fleet1", packetSend);
  }
  MQTT.processing();
```

Kode di atas merupakan implementasi kode yang digunakan untuk mengirim data dengan modul GSM SIM900 dan Arduino Mega 2560. Isi dari fungsi-fungsi yang digunakan pada implementasi ini terlampir.

Pada bagian awal kode dideklarasikan #include untuk memasukkan library yang dibutuhkan, dalam hal ini library untuk MQTT. MQTT_HOST merupakan alamat server yang akan dituju, sedangkan MQTT_PORT merupakan port yang digunakan untuk menuju alamat tersebut.

MQTT.begin() dipanggil pada saat subsistem elektronik pertama kali dinyalakan. Fungsi ini akan melakukan pengecekan komunikasi dengan modul GSM SIM900. Setelah komunikasi ini berhasil, fungsi ini akan melakukan aktivasi kartu SIM yang terpasang (jika belum diaktivasi) dan membuka koneksi TCP ke host MQTT melalui port yang telah dideklarasikan sebelumnya. Fungsi ini selesai dipanggil jika koneksi TCP ke server MQTT sudah berhasil dibuka.

Fungsi AutoConnect merupakan fungsi yang dipanggil secara otomatis saat koneksi TCP ke server MQTT berhasil dibuka. Prosedur ini berisi fungsi untuk melakukan koneksi untuk memberitahukan server bahwa ID yang akan digunakan menginginkan akses untuk dapat melakukan publish dan subscribe pada topik tertentu. Server MQTT akan merespon dan program akan keluar dari fungsi ini.

Fungsi berikutnya yaitu OnConnect, yaitu fungsi yang dipanggil saat ID yang digunakan sudah terkoneksi dengan server MQTT. Fungsi ini akan melakukan publish ke topik “fleet1” untuk mengirimkan data “connected” ke GUI agar operator pada GCS dapat mengetahui bahwa armada telah siap dijalankan. Setelah itu, program akan

melakukan subscribe ke topik “fleet1sub” untuk bersiap menerima komando dari GUI pada GCS.

Fungsi terakhir yaitu OnMessage, yaitu fungsi yang dipanggil saat ada pesan masuk ke ID yang digunakan. Pada implementasi ini, topik yang di-subscribe hanya “fleet1sub” sehingga fungsi ini akan dipanggil setiap ada komando dari GUI pada GCS. Setelah menerima pesan, pesan ini akan ditampilkan pada LCD.

GSM_MQTT MQTT(20) merupakan deklarasi nama class MQTT untuk digunakan pada program utama. Angka 20 menunjukkan keep alive duration dalam detik, sehingga subsistem elektronik dapat tetap terkoneksi dengan server meskipun tidak ada komunikasi sampai 20 detik.

Pada program utama, kode yang digunakan yaitu untuk mengecek kondisi koneksi ke server MQTT dengan fungsi MQTT.available(). Jika koneksi masih terbuka, program akan melakukan publish ke server MQTT dengan topik “fleet1” dan isi pesannya merupakan integrasi dari data-data yang diperlukan (dijelaskan pada bagian persiapan data). Fungsi dari variabel counter adalah agar pengiriman data tidak selalu dilakukan pada setiap loop program, tetapi masih dalam rentang waktu sesuai dengan spesifikasi.

3.5.4 Akuisisi Data Posisi melalui GPS

Untuk memperoleh data posisi armada, diperlukan sebuah sensor posisi. Dari arsitektur FMCS yang dipilih, sensor posisi yang cocok adalah sensor GPS. Sensor GPS akan menghasilkan NMEA sentence yang berisi banyak data, termasuk koordinat armada.

Terdapat beberapa jenis sensor GPS yang ada di pasaran. GPS Ublox Neo-M8N merupakan salah satu sensor terbaru yang diproduksi saat dokumen ini dibuat. Berikut spesifikasi GPS Ublox Neo-M8N.

Tabel 3. 4 Spesifikasi modul GPS

Jenis	Neo-8M
Input Voltage	1.65-3.6 Volts
Interface	UART, USB, SPI and DDC
Receiver Type	72-channel u-blox M8 engine GPS L1C/A SBAS L1C/A

	QZSS L1C/A GLONASS L1OF BeiDou B1 Galileo E1B/C2
Update Rate	Up to 10 Hz (GPS), Up to 5 Hz (GPS & GLONASS)
Accuracy	2.5 meters
Start Time	27 s cold start (GPS & GLONASS), 30 s cold start (GPS), 1 s hot start
Supported Antenna	Active and Passive
Operating Temperature	40 to 105 ºC

Dari tabel di atas, dapat dilihat modul GPS Ublox Neo-M8N dapat mendeteksi satelit GLONASS. Satelit ini merupakan satelit yang paling berpengaruh pada presisi data posisi di Indonesia. Selain itu, pada satelit GLONASS, modul ini memiliki *update rate* mencapai 5 Hz dan akurasi 2.5 meter.

Untuk melengkapi modul GPS ini, digunakan sebuah antenna aktif untuk menangkap sinyal GPS dengan lebih baik. Antenna aktif ini dihubungkan dengan modul GPS melalui kabel panjang agar antenna dapat diletakkan di luar kendaraan untuk kualitas sinyal yang lebih baik.

Pada implementasinya, modul GPS Ublox Neo-M8N akan mengirimkan NMEA sentence. Data ini diterima oleh Arduino dan diproses untuk hanya mengambil koordinatnya. Proses (parsing) ini memanfaatkan library Ublox.

Koneksi hardware yang digunakan pada implementasi ini yaitu pin Rx pada GPS dihubungkan ke pin Tx1 Arduino, sedangkan pin Tx pada GPS dihubungkan ke pin Rx1 Arduino. Agar Serial1 dapat menampung data GPS yang diterima, perlu dilakukan penambahan buffer size Arduino Mega menjadi 256 byte.

Berikutnya, diperlukan pengiriman perintah dari kontroller ke modul GPS untuk mengubah *refresh rate* dan *baud rate* modul GPS. Perintah ini merupakan urutan *bytes* sebagai berikut.

```
  byte Update5Hz[22] = { 0xB5, 0x62, 0x06, 0x08, 0x06, 0x00, 0xC8, 0x00, 0x01,
0x00,  0x01, 0x00, 0xDE, 0x6A, 0xB5, 0x62, 0x06, 0x08, 0x00, 0x00, 0x0E, 0x30 };
  byte BaudRate57600[28] = { 0xB5, 0x62, 0x06, 0x00, 0x14, 0x00, 0x01, 0x00, 0x00,
0x00,  0xD0, 0x08, 0x00, 0x00, 0x00, 0xE1, 0x00, 0x00, 0x07, 0x00, 0x03, 0x00, 0x00,
0x00, 0x00, 0x00, 0xDE, 0xC9 };
```

Berikut kode yang digunakan dalam implementasi modul GPS pada Arduino Mega 2560.

```
/* deklarasi */
#include "Ublox.h"
#define GPS_BAUD 57600
#define N_FLOATS 4

char latitude_chars[10];
char longitude_chars[11];

Ublox M8_Gps;
// Altitude - Latitude - Longitude - N Satellites
float gpsArray[N_FLOATS] = {0, 0, 0, 0};

/* setup */
Serial1.begin(9600);
byte Update5Hz[22] = { 0xB5, 0x62, 0x06, 0x08, 0x06, 0x00, 0xC8, 0x00, 0x01, 0x00,
0x01, 0x00, 0xDE, 0x6A, 0xB5, 0x62, 0x06, 0x08, 0x00, 0x00, 0x0E, 0x30 };
byte BaudRate57600[28] = { 0xB5, 0x62, 0x06, 0x00, 0x14, 0x00, 0x01, 0x00, 0x00,
0x00,  0xD0, 0x08, 0x00, 0x00, 0x00, 0xE1, 0x00, 0x00, 0x07, 0x00, 0x03, 0x00,
0x00, 0x00, 0x00, 0x00, 0xDE, 0xC9 };
Serial1.write(Update5Hz, 22);
delay(1000);
Serial1.write(BaudRate57600, 28);
delay(1);
Serial1.end();
delay(10);

Serial1.begin(GPS_BAUD);

/* program utama */
if(!Serial1.available())
    return;
  while(Serial1.available()){
        char c = Serial1.read();
         if (M8_Gps.encode(c)) {
          gpsArray[0] = M8_Gps.altitude;
          gpsArray[1] = M8_Gps.latitude;
          gpsArray[2] = M8_Gps.longitude;
          gpsArray[3] = M8_Gps.sats_in_use;
        }
  }
  dtostrf(gpsArray[1], 4, 7, latitude_chars);
  dtostrf(gpsArray[2], 4, 7, longitude_chars);
  for (i=0; i<11; i++)
    data[i] = (byte)longitude_chars[i];
  data[11] = (byte)':';
  for (i=0; i<10; i++)
    data[i+12] = (byte)latitude_chars[i];
  data[22] = (byte)':';
  data[12] = (byte)'-';

  if (data[0] != (byte)'1')
  {
    data[11] = (byte)':';
    data[22] = (byte)':';
    for (i=0; i<11; i++)
    {
      data[i] = (byte)'X';
    }
    for (i=0; i<10; i++)
    {
      data[i+12] = (byte)'X';
    }
  }
```

Pada bagian awal kode dideklarasikan penggunaan library Ublox.h dengan menggunakan fungsi #include. Selain itu ditentukan juga baudrate yang digunakan pada konstanta GPS_BAUD. Dua buah array of chars longitude_chars dan latitude_chars digunakan untuk menampung sementara nilai koordinat yang diperoleh dari modul GPS.

Untuk menggunakan fungsi-fungsi pada library Ublox.h, dibuat sebuah kelas Ublox bernama M8_Gps. Kelas ini kemudian akan digunakan untuk memanggil fungsi untuk mengambil data koordinat dari GPS.

Karena GPS dihubungkan dengan Serial1 Arduino, maka dideklarasikan Serial1.begin untuk memulai komunikasi melalui Serial1. Dalam program utama, dilakukan pengecekan kevalidan data GPS dan pengambilan data koordinat dari GPS, setelah itu disusun dalam variabel data. Jika data tidak valid, variabel data diisi dengan huruf 'X'.

3.5.5 Penentuan Kondisi Normal dan Emergency

Untuk menentukan kondisi normal dan emergency, dibuat sebuah *switch* yang dikendalikan manual oleh pengemudi *guided bus*. Pada saat keadaan normal, *switch* diatur untuk berada pada kondisi normal. Pada kondisi ini, subsistem elektronik akan mengirimkan data armada ke *control station*. Pada saat keadaan armada tidak dapat beroperasi, *switch* diatur untuk berada pada kondisi emergency. Pada kondisi ini, subsistem elektronik akan mengirimkan pesan emergency ke *control station*.

3.5.6 Modul Display

Pada implementasinya, modul LCD display 20x4 digunakan untuk menampilkan pesan dari GUI pada GCS. Berikut kode yang digunakan untuk implementasi display menggunakan LCD display 20x4.

```
/* deklarasi */
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x3F, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);

/* setup */
lcd.begin(20,4);
lcd.backlight();
lcd.setCursor(0, 3);
```

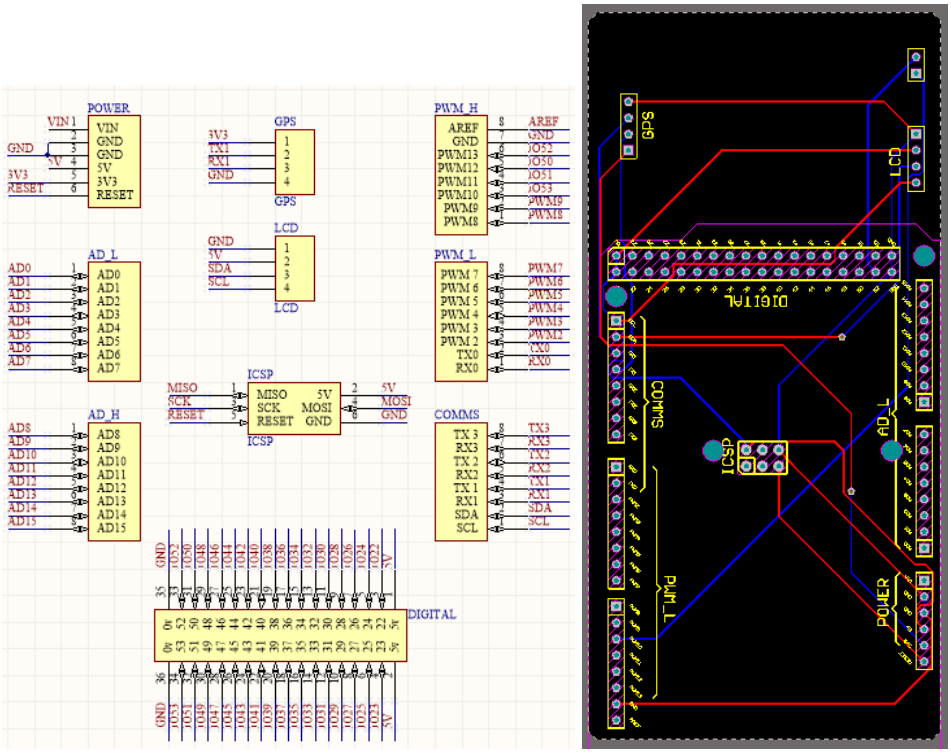
Pada awal program dideklarasikan library yang diperlukan yaitu Wire.h dan LiquidCrystal_I2C.h menggunakan fungsi #include. Berikutnya dideklarasikan juga LiquidCrystal_I2C lcd untuk membuat class lcd yang akan digunakan pada program utama.

Pada bagian setup dipanggil fungsi begin untuk memulai komunikasi dengan LCD dan parameter 20,4 yang menunjukkan ukuran LCD yang digunakan. Fungsi backlight() digunakan untuk menyalakan LED yang menjadi background LCD. Fungsi setCursor digunakan untuk memindahkan kursor ke tempat yang diinginkan. Terdapat sebuah fungsi lagi yang sering dipanggil pada kode implementasi modul lain yaitu print, yang digunakan untuk mencetak karakter yang diinginkan pada posisi kursor.

3.5.7 PCB Shield

Pada integrasi subsistem elektronik dilakukan pembuatan sebuah shield untuk menghubungkan LCD dan GPS dengan Arduino sehingga bisa digabungkan dengan modul-modul lainnya dengan rapi. Untuk membuat shield ini perlu dimulai dengan membuat skematiknya terlebih dahulu. Kemudian setelah skematik selesai, dibuat imeplementasi desain PCBnya. Implementasi PCB ini harus menyesuaikan dengan dimensi Arduino Mega 2560.

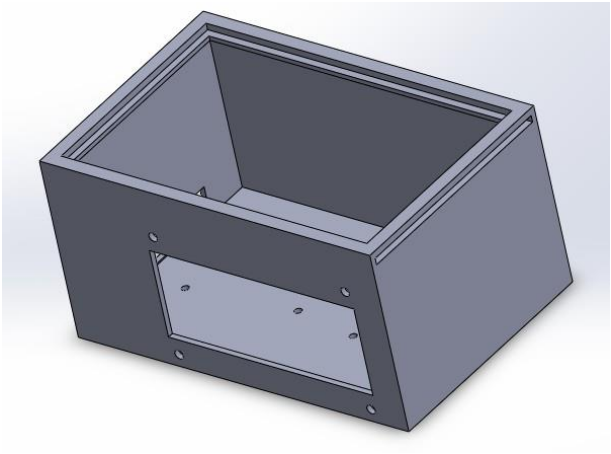
Berikut skematik dan desain PCB yang digunakan untuk membuat shield Arduino Mega 2560.



Gambar 3. 8 Skematik dan PCB shield

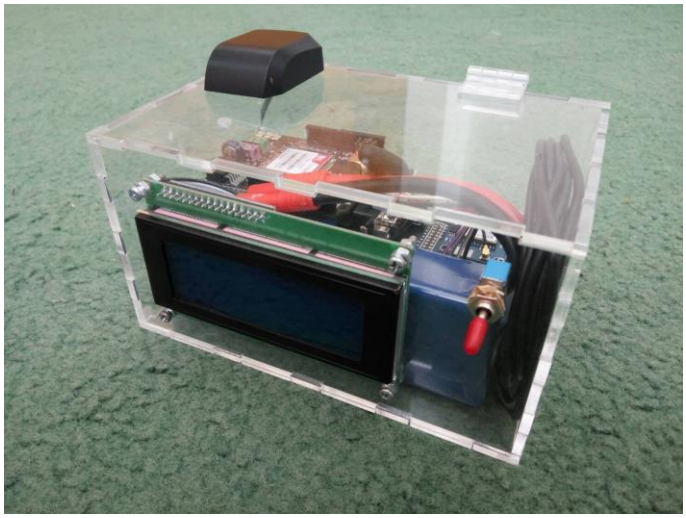
3.5.8 Packaging

Untuk kebutuhan pengujian dan mengemas seluruh komponen pada subsistem elektronik, didesain sebuah kotak *packaging* yang akan diisi oleh seluruh komponen yang digunakan. Berikut gambar desain *packaging*.



Gambar 3. 9 Desain packaging

Pada implementasinya, *packaging* dibuat dengan menggunakan bahan akrilik 3 mm. Penggunaan bahan ini dikarenakan tujuan *packaging* sebatas untuk keperluan pengujian, bukan untuk penggunaan pada *guided bus*.



Gambar 3. 10 Prototipe subsistem elektronik

BAB IV

HASIL PENGUJIAN

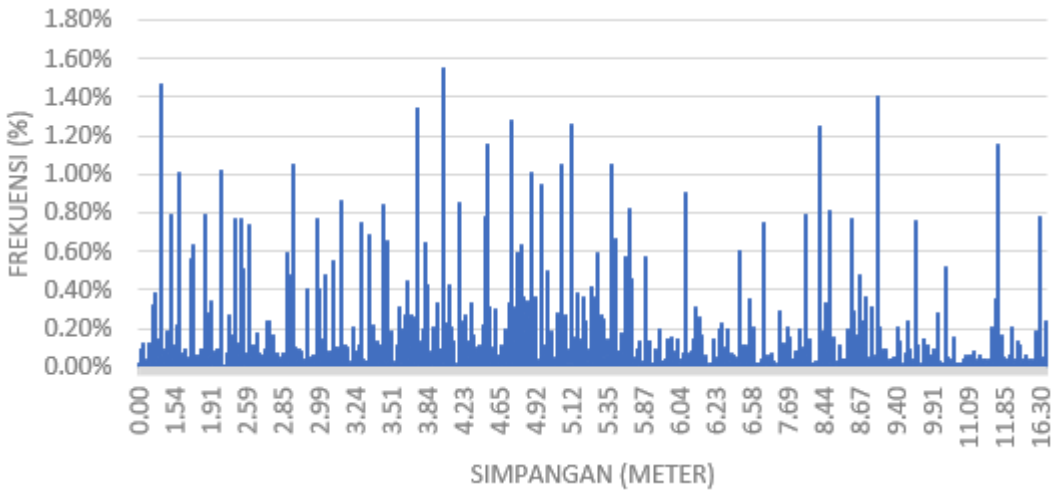
Pengujian subsistem elektronik dilakukan dengan konfigurasi sebagai berikut.

- Laptop Intel i5-6200U 2.8 GHz, RAM 4 GB, system operasi Windows 10 sebagai klien untuk menangkap data waktu pengiriman.
- Daya subsistem elektronik dari baterai Li-Po 5000 mAh 20C.
- CAN Bus dari Mini AGT di PT. LEN Industri.

Pengujian ini melingkupi 2 aspek yaitu pengujian fungsional subsistem elektronik dan verifikasi terpenuhi atau tidaknya spesifikasi yang telah dijabarkan pada bagian sebelumnya.

4.1 Penyimpangan Data GPS

Pada pengujian ini dilakukan pengamatan terhadap akurasi data GPS yang diambil dan *error* yang terjadi. Pengujian ini dilakukan dengan meletakkan subsistem elektronik pada satu titik uji yang telah diketahui koordinatnya, kemudian direkam data posisi yang dikirimkan oleh subsistem elektronik pada suatu program untuk kemudian dihitung jaraknya terhadap titik uji. Dari 4148 data yang diperoleh saat pengujian, berikut grafik persebaran data yang diperoleh.



Gambar 4. 1 Persebaran penyimpangan data GPS

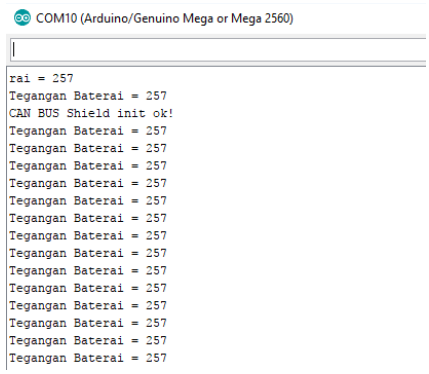
Dari gambar di atas dapat dilihat persebaran penyimpangan yang terjadi tidak terdistribusi secara normal. Dari seluruh data tersebut, terdapat 11.52 % data yang

melebihi spesifikasi, sehingga dapat dikatakan GPS yang digunakan 88.48 % memenuhi spesifikasi error pembacaan lokasi.

4.2 Akuisisi Data dari CAN Bus

Pada pengujian ini dilihat keberhasilan komunikasi antara subsistem elektronik dengan ECU lain yang ada dalam *guided bus* melalui CAN Bus. Subsistem elektronik dihubungkan dengan CAN Bus yang ada di Mini AGT PT LEN Industri untuk dilihat data baterai dan faultnya sesuai dengan tabel data CAN pada gambar 3.7.

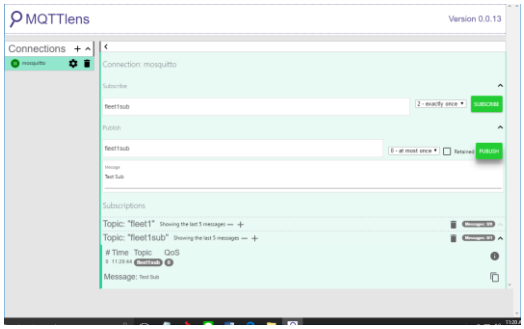
Pada Mini AGT ini, data dengan ID 4 dan 5 dikirimkan secara periodic, sedangkan data pada ID 1 hanya dikirimkan saat terjadi fault. Berikut gambar hasil pembacaan data dari CAN Bus Mini AGT.



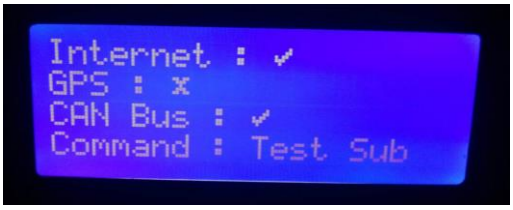
Gambar 4. 2 Hasil pembacaan data CAN Bus

4.3 Penerimaan Data dari Server

Pada pengujian ini digunakan aplikasi MQTTLens untuk mengirimkan dummy data. Berikut gambar aplikasi MQTTLens dan hasil tampilan data yang dikirimkan.



Gambar 4. 3 Pengiriman data dari MQTTLens



Gambar 4. 4 Hasil tampilan data dari server

Dari gambar di atas dapat dilihat pesan yang dikirimkan dari MQTTLens sudah dapat ditampilkan pada LCD Display.

4.4 Pengujian di Lapangan

Pada pengujian di lapangan, subsistem elektronik digunakan untuk mengirim data posisi kendaraan. Dalam pengujian ini kendaraan yang digunakan adalah sepeda motor. Selama subsistem elektronik mengirimkan data posisi, GUI dinyalakan untuk mengamati data yang dikirimkan dari subsistem elektronik. Selain selama kondisi normal, dilakukan juga pengujian pada kondisi emergency, yaitu subsistem elektronik akan mengirimkan pesan emergency ke server.

Hasil pengujian ini yaitu data yang dikirimkan dari subsistem elektronik sudah sesuai dengan susunan data yang telah dijelaskan pada bagian implementasi.

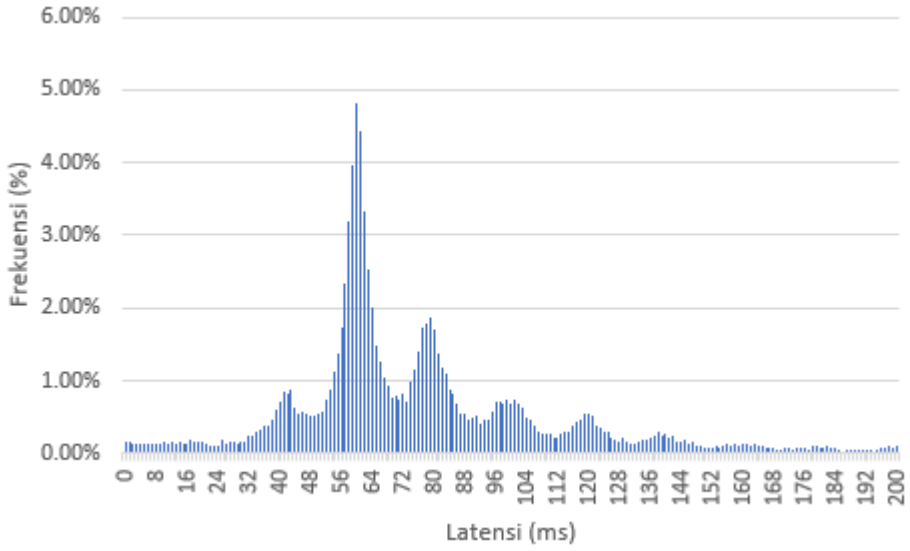
Untuk keadaan emergency, subsistem elektronik juga sudah dapat mengirimkan pesan emergency ke server.

4.5 Pengiriman Data ke Server

Pada pengujian pengiriman data ke server, dilakukan pengamatan terhadap latensi yang diakibatkan oleh jaringan GSM dan server itu sendiri. Untuk melihat latensi, subsistem elektronik dibiarkan menyala selama 12 jam sambil mengirimkan data sesuai dengan fungsinya dengan periode 600ms. Selain itu, dibuat sebuah program yang menangkap data tersebut untuk kemudian direkam waktu interval penerimaan antar data. Waktu interval ini kemudian dikurangi dengan waktu periode pengiriman data (yaitu 600ms) untuk melihat latensi pengirimannya saja. Latensi ini kemudian diplot dalam sebuah grafik yang menggambarkan persebaran latensi.

Dari pengujian selama 12 jam, diperoleh 84722 data interval pengiriman data. Dari seluruh data ini, terdapat 1.3% data latensi yang melebihi interval pengiriman data. Jika data dikirim dengan latensi tersebut, posisi data yang diterima oleh *control station* sudah tidak tidak presisi.

Untuk mengukur latensi pengiriman yang valid, digunakan 45927 data latensi yang tidak terpengaruh kondisi data tertahan di jaringan. Dari seluruh data yang valid ini kemudian dibuat dalam bentuk grafik persebaran seperti gambar berikut.



Gambar 4. 5 Grafik persebaran latensi

Rata-rata latensi yang diperoleh adalah 76.1 ms. Nilai ini berbeda dengan mediannya yaitu 66 ms. Perbedaan ini menunjukkan persebaran yang kurang terdistribusi secara normal sempurna. Namun, jika dilihat dari grafik di atas, bentuk persebaran latensi ini mendekati persebaran Gaussian yang tergeser. Oleh karena itu, perhitungan statistika latensi pengiriman ini dapat didekati dengan persebaran Gaussian.

Dari data yang diperoleh, dilakukan pendekatan persebaran Gaussian untuk ditentukan parameter statistika sebagai berikut.

Rata-Rata	**Standar Deviasi**	**Margin of Error**
76.1 ms	33.8 ms	0.3 ms

Margin or error yang diperoleh menggunakan selang kepercayaan 95%. Dari pengukuran di atas, dapat dikatakan latensi pengiriman data yang dilakukan adalah sebesar **76.1** ± **0.3 ms** dengan selang kepercayaan **95%**.

Dari nilai latensi ini, dapat dilihat bahwa data dari subsistem elektronik sampai ke *control station* setiap interval maksimal yaitu 600 + 76.1 + 0.3 ms, atau sekitar 676.4 ms. Nilai ini masih berada dalam batas spesifikasi.

BAB V

PENUTUP

5.1 Simpulan

Dari hasil pengujian yang dilakukan, subsistem elektronik sudah berhasil mengakuisisi data dari ECU lain yang ada dalam guided bus. Subsistem elektronik juga sudah dapat menghasilkan data posisi armada dengan 88.48% data *error* tidak melebihi 6 meter. Data-data ini juga telah berhasil dikirimkan ke server melalui jaringan internet GSM. Pengiriman ini memiliki latensi sebesar 76.1 ± 0.3 ms, dimana latensi ini dapat ditoleransi dan masuk ke dalam batas spesifikasi.

5.2 Saran

Pada pengujian latensi jaringan, masih terdapat sebagian kecil pengiriman data dengan latensi mencapai 200 ms. Agar tidak terjadi latensi yang melebihi batas spesifikasi, modul GSM yang digunakan dapat diganti menjadi SIM5215A atau Telit UC864-E. Modul ini dapat bekerja dalam jaringan 3G dengan kecepatan mencapai 384 kbps. Untuk data GPS yang lebih baik, dapat digunakan modul GPS dengan akurasi yang lebih tinggi.

DAFTAR PUSTAKA

Corrigan, Steve. "Introduction to the Controller Area Network (CAN)". 2008. Texas Instruments Application Report.

DePriest, Dale. “NMEA Data”. http://www.gpsinformation.org/dale/nmea.htm

Fabbri G., dkk. "Development of an On-Board Unit for the Monitoring and Management of an Electronic Fleet". 2012. XXth International Conference on Electrical Machines.

Lukihardianti, Arie. 2014. “Kerugian Akibat Macet di Bandung Capai Rp 4,36 Triliun”. http://www.republika.co.id, 16 November 2014.

Salceanu, Andrei, dkk. "Monitoring the Environment by Enhancing the Capabilities of a Fleet Tracking System". 2014. International Conference and Exposition on Electrical and Power Engineering (EPE).

SIMCom. “SIM900 Hardware Design V2.00”. 2010. SIMCom SIM Tech Company

Sveum P., dkk. "Wireless Monitoring of an Electric Fleet Hub". 2011. IEEE Vehicle Power and Propulsion Conference.

LAMPIRAN

Source code program utama

```
/*  Nama Program     :  TA161701094_FMCS
 *  Developer        :  Ali Zaenal Abidin (13213106) - TA161701094
 *  Deskripsi        :  Program FMCS ini dibuat sebagai bagian
dari tugas akhir teknik elektro 2016/2017.
 *                      Bagian elektrik dari FMCS berfungsi untuk
mengambil data dari CAN Bus dan GPS
 *                      untuk kemudian dienkripsi dan dikirimkan
ke server. Flowchart program terdapat
 *                      pada dokumen terkait.
 *  Setting Hardware :  Mikrokontroller Arduino Mega 2560
 *                      GPS Neo-M8N dihubungkan ke Serial1 Arduino
 *                      Shield CAN Bus dari DFRobot dihubungkan
dengan pin CS di pin 10
 *                      Shield SIM900 dihubungkan ke Serial utama
Arduino Mega. Untuk debugging dihubungkan
 *                        ke SoftSerial 2 dan 3 pada Arduino Uno
 *                      LCD 20x4 dihubungkan ke pin I2C (SDA dan
SCL) pada Arduino Mega
 *                      Power menggunakan external power supply
antara 7-12 Volt (bisa menggunakan baterai
 *                        Li-Po atau power supply dari jala-jala)
 */

/********************************** ******** GLOBALS ********
**********************************/
#define NORMAL 0
#define EMERGENCY 1

int counter = 0;
int i = 0;
int jarak = 0;
int jaraktemp0 = 0;
int jaraktemp1 = 0;
int state = 0;
bool sentEm = false;
/********************************** ******** END GLOBALS ********
**********************************/


/************************************ ******** LCD ********
************************************/
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x3F, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);  //
LCD I2C address
uint8_t check[8] = {0x0,0x1,0x3,0x16,0x1c,0x8,0x0};
uint8_t cross[8] = {0x0,0x1b,0xe,0x4,0xe,0x1b,0x0};
/********************************** ******** END LCD ********
**********************************/
```

```
/****************************** ******** DATA CONSTRUCTION
******** ******************************/
byte data[32];
/***************************** ******** END DATA CONSTRUCTION
******** *****************************/


/************************************* ******** GPS ********
*************************************/
#include "Ublox.h"
#define GPS_BAUD 57600
#define N_FLOATS 4

char latitude_chars[10];
char longitude_chars[11];

Ublox M8_Gps;
// Altitude - Latitude - Longitude - N Satellites
float gpsArray[N_FLOATS] = { 0, 0, 0, 0 };
/************************************ ******** END GPS ********
************************************/


/****************************** ******** AES Encryption ********
******************************/
#include <AES.h>
#include <AES_config.h>
#if (defined(__AVR__))
#include <avr\pgmspace.h>
#else
#include <pgmspace.h>
#endif

AES aes ;

byte key[16] = {0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41}; // key untuk
enkripsi
byte plain[32];
byte cipher[48];
byte decrypted[32];
void prekey (int bits)
/*
 * fungsi ini digunakan untuk melakukan enkripsi data. variabel
 * plain diisi dengan isi dari variabel data, kemudian variabel
 * plain ini dienkripsi dengan key dan iv yang telah diset dalam
 * fungsi ini. hasil enkripsi dimasukkan ke variabel cipher.
 */
{
  for (i=0; i<=31; i++)
    plain[i] = data[i];
  aes.iv_inc();
  byte iv [16] = {0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,
0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41};
  aes.do_aes_encrypt(plain, 32, cipher, key, bits, iv);
  delay(10);
}
```

```
void prekey_test ()
/*
 * fungsi ini digunakan untuk memanggil fungsi prekey dengan
 * 128 bits.
 */
{
  prekey(128);
}
/****************************** ******** END AES Encryption
******** ******************************/


/********************************** ******** CAN ********
**********************************/
#include <SPI.h>
#include "mcp_can.h"
const int SPI_CS_PIN = 10;
MCP_CAN CAN(SPI_CS_PIN);
/********************************** ******** END CAN ********
**********************************/


/********************************** ******** MQTT ********
**********************************/
#include "GSM_MQTT.h"
#include <SoftwareSerial.h>
String MQTT_HOST = "id.tunnel.my.id";
String MQTT_PORT = "1126";
SoftwareSerial cobac(3, 4);
int elapsedTime = 0;
int elapsedPing = 0;

void GSM_MQTT::AutoConnect(void)
/*
 * fungsi ini dipanggil setiap koneksi TCP berhasil dibuka
 * oleh modul GSM SIM900. fungsi ini membuka koneksi client
 * ke server dengan ID TA161701094.
 */
{
  connect("TA161701094", 0, 0, "", "", 1, 0, 0, 0, "", "");
}
void GSM_MQTT::OnConnect(void)
/*
 * fungsi ini dipanggil setiap koneksi client dengan ID tertentu
 * berhasil. fungsi ini melakukan subscribe ke topik fleet1sub dan
 * mempblish "connected" ke topik fleet1.
 */
{
  subscribe(0, _generateMessageID(), "fleet1sub", 1);
  publish(1, 1, 0, _generateMessageID(), "fleet1a", "connected");
}
void GSM_MQTT::OnMessage(char *Topic, int TopicLength, char
*Message, int MessageLength)
/*
 * fungsi ini dipanggil setiap ada pesan masuk melalui modul GSM
 * SIM900. fungsi ini akan menampilkan pesan tersebut ke layar.
```

```
 */
{
  lcd.setCursor(12, 3);
  lcd.print(Message);
}

GSM_MQTT MQTT(20);
/********************************** ******** END MQTT ********
**********************************/


/******************************** ******** MAIN PROGRAM ********
********************************/
void setup()
/* SETUP DIJALANKAN DI AWAL PROGRAM
 * pada setup dilakukan inisialisasi fungsi-fungsi dan modul
 * yang digunakan pada fleet hardware. inisialisasi ini meliputi
 * gps, lcd, can bus dan modul gsm untuk mqtt.
 */
{
/*
 * inisialisasi data dari CAN
 */
  data[23] = (byte)'X';
  data[24] = (byte)'X';
  data[26] = (byte)'X';
  data[27] = (byte)'X';
  data[28] = (byte)'X';
  data[29] = (byte)'X';
  data[30] = (byte)'X';
  data[31] = (byte)'1';
  /******** GPS ********/
  Serial1.begin(9600);
  delay(1);
  byte Update5Hz[22] = { 0xB5, 0x62, 0x06, 0x08, 0x06, 0x00, 0xC8,
0x00, 0x01, 0x00,  0x01, 0x00, 0xDE, 0x6A, 0xB5, 0x62, 0x06, 0x08,
0x00, 0x00, 0x0E, 0x30 };
  byte BaudRate57600[28] = { 0xB5, 0x62, 0x06, 0x00, 0x14, 0x00,
0x01, 0x00, 0x00, 0x00,  0xD0, 0x08, 0x00, 0x00, 0x00, 0xE1, 0x00,
0x00, 0x07, 0x00, 0x03, 0x00, 0x00, 0x00, 0x00, 0xDE, 0xC9
};
  Serial1.write(Update5Hz, 22);
  delay(1000);
  Serial1.write(BaudRate57600, 28);
  delay(1);
  Serial1.end();
  delay(10);
  Serial1.begin(GPS_BAUD);
  /******** LCD ********/
  lcd.begin(20,4);
  lcd.createChar(0, check);
  lcd.createChar(1, cross);
  lcd.backlight();
  lcd.setCursor(0, 3);
  lcd.print("Kirim : ");

  /******** CAN ********/
```

```
  while (CAN_OK != CAN.begin(CAN_1000KBPS))
  {
    lcd.setCursor(0, 2);
    lcd.print("CAN INIT FAILED");
    delay(100);
  }
  lcd.setCursor(0, 2);
  lcd.print("CAN INIT OK");

  /******** MQTT ********/
  MQTT.begin();
  delay(100);
  MQTT.turnOn();
  delay(1);
}

void loop()
{
  char packetSend[32];
  /* MENYIAPKAN DELAY
   * delay digunakan untuk membatasi pengiriman
   * data agar tidak terlalu cepat, tetapi masih
   * dalam batas spesifikasi.
   */
  if (elapsedTime > 500) elapsedTime = 500;
  delay(500 - elapsedTime);
  elapsedTime = millis();

  /******** GPS ********/
  /* MEMBACA DATA GPS
   * data yang diambil adalah latitude dan
   * longitude.
   */
  while(Serial1.available())
  {
    char c = Serial1.read();
    if (M8_Gps.encode(c))
    {
    //gpsArray[0] = M8_Gps.altitude;
    gpsArray[1] = M8_Gps.latitude;
    gpsArray[2] = M8_Gps.longitude;
    //gpsArray[3] = M8_Gps.sats_in_use;
    }
  }
  /* KONSTRUKSI DATA GPS
   * mengubah data float menjadi data bertipe array of char,
   * kemudian menyusun data koordinat dalam variabel data.
   */
  dtostrf(gpsArray[1], 4, 7, latitude_chars);
  dtostrf(gpsArray[2], 4, 7, longitude_chars);
  for (i=0; i<11; i++)
    data[i] = (byte)longitude_chars[i];
  data[11] = (byte)':';
  for (i=0; i<10; i++)
    data[i+12] = (byte)latitude_chars[i];
  data[22] = (byte)':';
  data[12] = (byte)'-';
```

```
/* PENANGANAN DATA GPS TIDAK VALID
 * jika data yang diterima dari gps tidak valid (atau tidak
 * ada data dari gps), hardware akan mengirimkan pesan eror
 * (dengan karakter 'X') pada data koordinat.
 */
if (data[0] != (byte)'1')
{
  data[11] = (byte)':';
  data[22] = (byte)':';
  for (i=0; i<11; i++)
  {
    data[i] = (byte)'X';
  }
  for (i=0; i<10; i++)
  {
    data[i+12] = (byte)'X';
  }
}

/******** CAN ********/
unsigned char len = 0;
unsigned char buf[8];

/* MEMBACA DATA CAN BUS
 * mengambil data dari CAN Bus. data yang diambil
 * memiliki ID 1.
 * Jika belum ada data masuk dari CAN Bus, akan
 * dikirimkan pesan dengan data 'X'
 */
data[25] = (byte)':';
if(CAN_MSGAVAIL == CAN.checkReceive())
{
  CAN.readMsgBuf(&len, buf);
  unsigned char canId = CAN.getCanId();
  lcd.setCursor(11, 5);
  lcd.print(canId);

  /* data baterai */
  if (canId == 5)
  {
    if (buf[0] == 1)
    {
      data[23] = (byte)buf[2];
      data[24] = (byte)buf[1];
    }
  }
  /* data fault */
  if (canId == 1)
  {
    data[26] = (byte)buf[0];
    data[27] = (byte)buf[1];
    data[28] = (byte)buf[2];
  }
  /* data rpm */
  else if (canId == 3)
  {
    data[30] = (byte)buf[3];
```

```
      data[29] = (byte)buf[4];
    }
  }

  /******** DEBUGGING ********/
  lcd.setCursor(0, 0);
  char coba[32];
  lcd.print("data :");
  for (i=0; i<32; i++)
  {
    coba[i] = (char)data[i];
    lcd.print(coba[i]);
  }

  /******** AES Encryption ********/
  /* PROSES ENKRIPSI
   * prekey_test dipanggil untuk melakukan enkripsi pada
   * variabel data (bertipe byte). hasil dekripsi dimasukkan
   * ke variabel cipher (bertipe byte).
   */
  prekey_test();

  /* PERSIAPAN PENGIRIMAN DATA
   * mengubah data hasil enkripsi dari tipe byte menjadi
   * tipe karakter
   */
  for (i=0; i<=31; i++)
    packetSend[i] = (char)(cipher[i]);

  /******** MQTT ********/
  /* PENGIRIMAN DATA
   * pengiriman data dibagi menjadi dua keadaan, yaitu
   * saat keadaan normal dan emergency.
   * pada keadaan normal, alat akan mengirimkan fleet data
   * seperti seharusnya. pada keadaan emergency, alat akan
   * mengirimkan pesan ke topik yang berbeda untuk menandakan
   * keadaan emergency.
   */
  if (MQTT.available())
  {
    if (digitalRead(3) == LOW) state = EMERGENCY;
    else state = NORMAL;

    if ((state == EMERGENCY) && (!sentEm))
    {
      MQTT.publish(1, 0, 0, MQTT._generateMessageID(), "fleetem",
"fleet1");
      sentEm = true;
    }
    else if (state == NORMAL)
    {
      sentEm = false;
      MQTT.publish(1, 0, 0, MQTT._generateMessageID(), "fleet1",
packetSend);
      /* Hitung interval pengiriman data */
      counter++;
    }
```

```
  }
  lcd.setCursor(8, 3);
  lcd.print(counter);
  MQTT.processing();
  elapsedTime = millis() - elapsedTime;
  lcd.setCursor(15, 3);
  lcd.print(elapsedTime);
  if (elapsedTime < 1000)
  {
    lcd.setCursor(18, 3);
    lcd.print("  ");
  }
}
```