



INSTITUT TEKNOLOGI BANDUNG

PROGRAM STUDI TEKNIK ELEKTRO

JALAN GANESHA NO. 10 Gedung Labtek V Lantai 2 **Telepon:** (022)2508135-36, **Faks:** (022)250 0940
BANDUNG 40132

Dokumentasi Produk Tugas Akhir

Lembar Sampul Dokumen

Judul Dokumen

TUGAS AKHIR TEKNIK ELEKTRO:
Fleet Monitoring and Control System pada Guided Bus

Jenis Dokumen

IMPLEMENTASI

Catatan: Dokumen ini dikendalikan penyebarannya oleh Prodi Teknik Elektro ITB

Nomor Dokumen

B400-02-TA1617.01.094

Nomor Revisi

02

Nama File

B400-02-TA1617.01.094

Tanggal Penerbitan

12 May 2017

Unit Penerbit

Prodi Teknik Elektro – ITB

Jumlah Halaman

94

(termasuk lembar sampul ini)

Data Pengusul

Pengusul	Nama	Ali Zaenal Abidin	Jabatan	Mahasiswa
	Tanggal	4 Mei 2017	Tanda Tangan	
	Nama	Shah Dehan Lazuardi	Jabatan	Mahasiswa
	Tanggal	4 Mei 2017	Tanda Tangan	
	Nama	Aulia Hening Darmasti	Jabatan	Mahasiswa
	Tanggal	4 Mei 2017	Tanda Tangan	
Pembimbing	Nama	Arief Syaichu R.	Tanda Tangan	
	Tanggal	4 Mei 2017		

Lembaga

Program Studi Teknik Elektro
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

Alamat

Labtek V, Lantai 2, Jalan Ganesha no. 10, Bandung

Telepon : +62 22 250 2260

Faks :+62 22 253 4222

Email:stei@stei.itb.ac.id

DAFTAR ISI

DAFTAR ISI.....	2
CATATAN SEJARAH PERBAIKAN DOKUMEN.....	3
FLEET MONITORING AND CONTROL SYSTEM PADA GUIDED BUS.....	4
1 PENGANTAR	4
1.1 RINGKASAN ISI DOKUMEN	4
1.2 TUJUAN PENULISAN DAN APLIKASI/KEGUNAAN DOKUMEN	4
1.3 REFERENSI	4
1.4 DAFTAR SINGKATAN.....	5
2 IMPLEMENTASI.....	5
2.1 IMPLEMENTASI SUB-SISTEM ELEKTRIK	5
2.1.1 <i>Struktur Implementasi</i>	6
2.1.2 <i>Flowchart Implementasi</i>	6
2.1.3 <i>Environment</i>	8
2.1.4 <i>Prosedur Implementasi</i>	12
2.2 IMPLEMENTASI SUB-SISTEM SERVER.....	30
2.2.1 <i>Struktur Implementasi</i>	31
2.2.2 <i>Environment</i>	31
2.2.3 <i>Prosedur Implementasi</i>	31
2.3 IMPLEMENTASI SUB-SISTEM GRAPHICAL USER INTERFACE (GUI)....	36
2.3.1 <i>Struktur Implementasi</i>	36
2.3.2 <i>Environment</i>	37
2.3.3 <i>Prosedur Implementasi</i>	38
2.4 IMPLEMENTASI ALGORITMA PENJADWALAN.....	57
2.4.1 <i>Struktur Implementasi</i>	57
2.4.2 <i>Environment</i>	57
2.4.3 <i>Prosedur Implementasi</i>	58
3 SIMPULAN	69
4 LAMPIRAN.....	71

Catatan Sejarah Perbaikan Dokumen

VERSI, TGL, OLEH	PERBAIKAN
2, 4 Mei 2017, Ali Zaenal Abidin	Mengganti implementasi modul GPS Mengganti desain skematik dan pcb shield Mengganti implementasi modul akuisisi data CAN

Fleet Monitoring and Control System pada Guided Bus

1 Pengantar

1.1 RINGKASAN ISI DOKUMEN

Dokumen ini berisi dokumentasi implementasi desain *fleet monitoring and control system* pada *guided bus*. Dalam dokumen ini terdapat implementasi dari sub-sistem elektrik, sub-sistem server dan sub-sistem *graphical user interface*(GUI). Selain itu juga terdapat proses uji coba, implementasi, permasalahan pada implementasi dan solusi untuk permasalahan yang terjadi.

1.2 Tujuan Penulisan dan Aplikasi/Kegunaan Dokumen

Tujuan dari penulisan dokumen ini yaitu sebagai berikut.

1. Mendokumentasikan proses dan hasil implementasi produk *fleet monitoring and control system* pada *guided bus*.
2. Sebagai referensi untuk pihak lain yang ingin merancang sistem sama atau serupa dengan *fleet monitoring and control system* pada *guided bus*.

1.3 REFERENSI

- [1] NMEA Data. <http://www.gpsinformation.org/dale/nmea.htm>.
- [2] Simcom. 2010. SIM900 AT Command Manual V1.03.
- [3] Ublox. U-Center GNSS Evaluation Software for Windows User Guide.
- [4] <https://www.eclipse.org/paho/files/mqtt/doc/Cclient/qos.html>
- [5] [https://en.wikipedia.org/wiki/Python_\(programming_language\)](https://en.wikipedia.org/wiki/Python_(programming_language))
- [6] <http://www.movable-type.co.uk/scripts/latlong.html>
- [7] <http://pramukaria.blogspot.co.id/2015/10/teknik-menentukan-azimuth-dan-back.html>
- [8] <https://yulikausman.wordpress.com/2015/04/08/menghitung-jarak-sudut-dan-azimut/>
- [9] <http://learntocodewith.me/programming/python/python-2-vs-python-3/>
- [10] Fabbri G., dkk. "Development of an On-Board Unit for the Monitoring and Management of an Electrical Fleet". 2012. XXth International Conference on Electrical Machines.
- [11] 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).
- [12] Sveum P., dkk. "Wireless Monitoring of an Electric Fleet Hub". 2011. IEEE Vehicle Power and Propulsion Conference.

[13] Sriborriux, Wiroon, dkk. "The Design of RFID Sensor Network for Bus Fleet Monitoring". 2008. 8th International Conference on ITS Telecommunications.

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

1.4 DAFTAR SINGKATAN

SINGKATAN	ARTI
GPS	<i>Global Positioning System</i>
FMCS	<i>Fleet Monitoring and Control System</i>
ECU	<i>Engine Control Unit</i>
GUI	<i>Graphical User Interface</i>
CAN	<i>Controller Area Network</i>
UART	<i>Universal Asynchronous Receiver Transmitter</i>
HILS	<i>Hardware in the Loop Simulation</i>
MQTT	<i>Message Queuing Telemetry Transport</i>

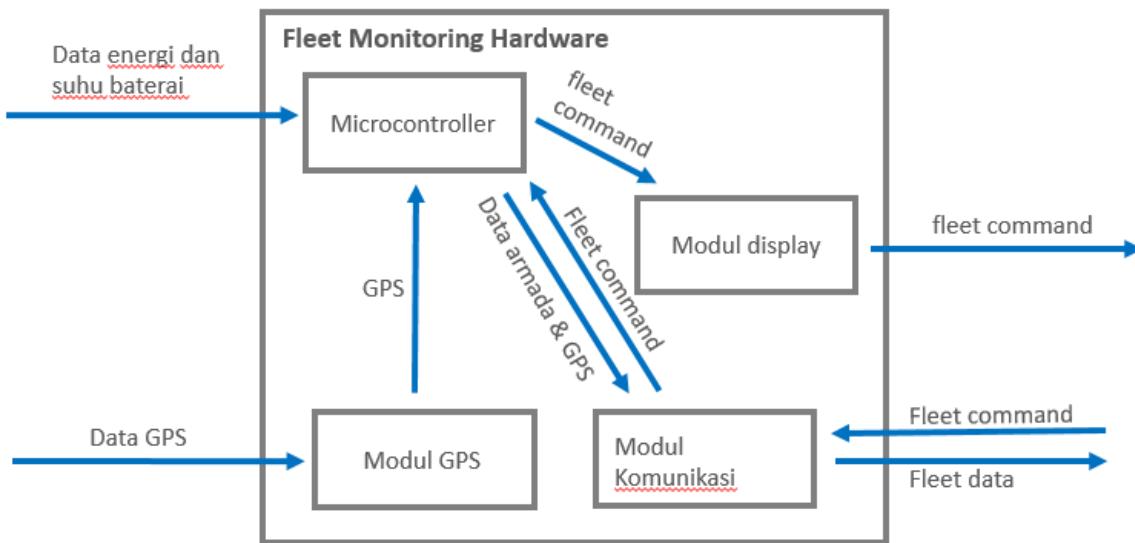
2 IMPLEMENTASI

2.1 IMPLEMENTASI SUB-SISTEM ELEKTRIK

Sub-sistem elektrik merupakan sub-sistem yang berfungsi untuk mengambil data yang diperlukan dari setiap armada *guided bus* dan mengirimkan data tersebut ke *control station* melalui jaringan internet. Selain itu, sub-sistem elektrik juga berfungsi untuk menerima *fleet command* dan menampilkannya melalui sebuah antarmuka agar pengendara armada dapat memahami *fleet command* tersebut. Sub-sistem ini terdiri dari beberapa perangkat keras berupa modul-modul, *display* dan kontroller.

Modul-modul yang digunakan yaitu modul akuisisi data CAN untuk mengambil data dari CAN Bus pada armada *guided bus*, modul pelacak posisi GPS untuk mengambil data posisi armada *guided bus* dan modul internet GSM untuk komunikasi dengan *control station*. *Display* yang digunakan pada sub-sistem ini yaitu LCD *Display* berukuran 20x4. Terakhir, kontroller yang digunakan untuk sub-sistem elektrik yaitu mikrokontroller Arduino Mega 2560.

2.1.1 Struktur Implementasi



Gambar 1 Data flow diagram level 2 *hardware*

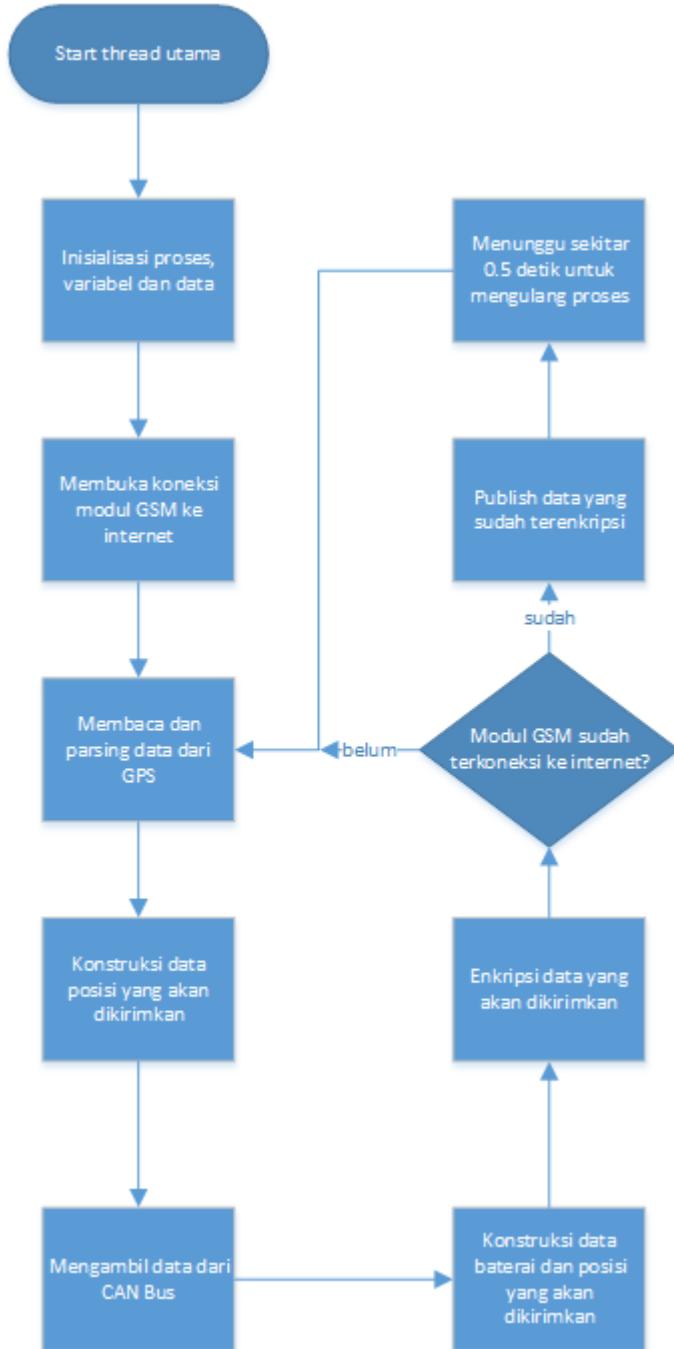
Terdapat empat modul yang ada pada sub-sistem elektrik. Keempat modul ini berkomunikasi dengan kontroller untuk mengirim dan menerima data sesuai dengan yang dibutuhkan.

Bagian pertama pada struktur implementasi ini adalah komunikasi serial kontroller dengan modul GPS untuk memperoleh data GPS. Komunikasi yang terjadi adalah modul GPS memberikan informasi lokasi NMEA yang diperoleh dari satelit yang terdeteksi. Bagian kedua yaitu komunikasi SPI kontroller dengan modul CAN Bus. Modul CAN Bus mengirimkan data baterai yang diperoleh dari CAN Bus armada. Bagian ketiga adalah komunikasi serial kontroller dengan modul internet untuk mengirimkan seluruh data yang diperoleh (*fleet data*) dan menerima *fleet command* dari *control station*. Komunikasi yang terjadi yaitu kontroller mengirimkan perintah pada modul komunikasi untuk membuka koneksi ke *server* dan mengirimkan *fleet data* ke *control station*, kemudian modul komunikasi akan mengirimkan balasan kepada kontroller untuk menentukan keberhasilan pengiriman data tersebut. Selain itu, modul komunikasi juga menerima *fleet command* yang akan diteruskan ke kontroller untuk kemudian diterjemahkan. Bagian keempat yaitu komunikasi I²C kontroller dengan modul *display*. Komunikasi yang terjadi yaitu kontroller memberi perintah kepada modul *display* untuk menampilkan kondisi koneksi *server* dan *fleet command*.

2.1.2 Flowchart Implementasi

Pada implementasinya, sub-sistem elektrik terdiri dari tiga *thread*. Thread utamanya yaitu bagian yang melakukan pengambilan data, enkripsi dan pengiriman data. Thread kedua

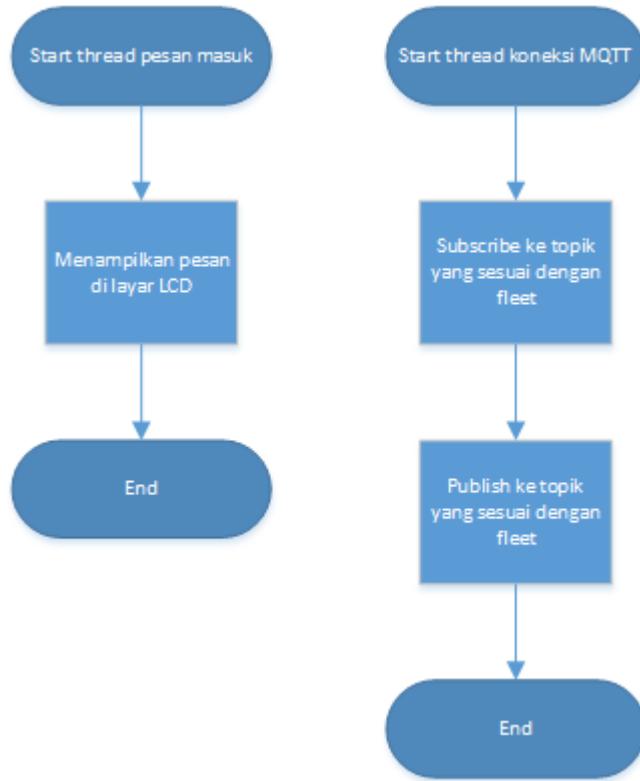
yaitu bagian yang membuka koneksi dari *fleet* ke server. Thread ketiga yaitu bagian yang melakukan tindakan setiap ada pesan masuk ke *fleet*.



Gambar 2 Flowchart program utama

Pada program utama, saat alat dinyalakan, dilakukan inisialisasi proses, data dan variabel yang diperlukan. Proses, data dan variabel yang diinisialisasi yaitu komunikasi dengan modul GSM, CAN dan GPS, variabel-variabel yang menampung data dan implementasi fungsi-fungsi yang dibutuhkan. Setelah itu, sistem akan memulai membuka koneksi ke internet. Kemudian, data dari GPS dibaca dan dipilah untuk memperoleh nilai koordinat fleet. Data tersebut dikonstruksi untuk menjadi data koordinat yang akan dikirimkan.

Setelah itu, sistem membaca data dari CAN Bus dan dikonstruksi untuk menjadi kesatuan data GPS dan CAN. Data yang sudah dikonstruksi ini dienkripsi untuk kemudian dikirimkan ke server. Proses ini diulang terus menerus selama alat dinyalakan.



Gambar 3 Flowchart komunikasi dengan server

Selain program utama, terdapat dua thread lain yang masing-masing berjalan saat ada pesan masuk ke fleet dan saat koneksi MQTT berhasil dibuka.

Pada thread pesan masuk, sistem akan menampilkan pesan tersebut ke layar LCD. Pada thread koneksi MQTT, sistem akan melakukan *subscribe* ke topik yang sesuai dan melakukan *publish* ke server untuk memberitahu GCS bahwa *fleet* yang bersangkutan berhasil terkoneksi.

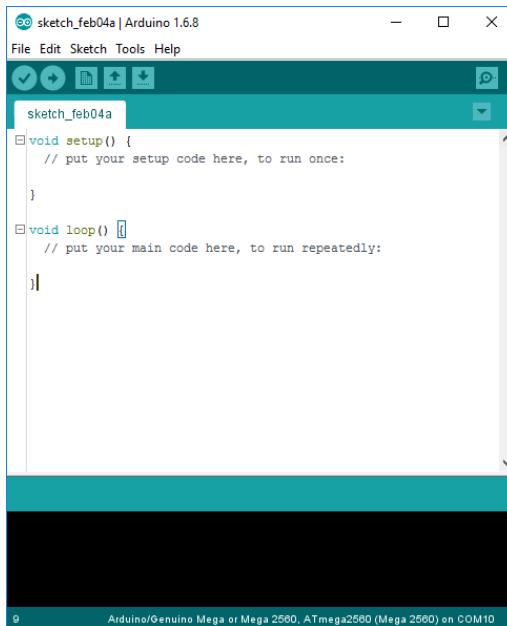
2.1.3 *Environment*

Dalam implementasi sub-sistem elektrik digunakan beberapa perangkat keras dan perangkat lunak sebagai berikut.

2.1.3.1 Arduino

Karena kontroller yang digunakan pada implementasi sub-sistem elektrik ini adalah Arduino Mega 2560, IDE yang digunakan untuk mengembangkan program pada ECU *monitoring* adalah Arduino IDE. Arduino IDE menggunakan bahasa pemrograman Arduino yang berbasis C++.

Arduino IDE yang digunakan pada pengembangan sub-sistem elektrik ini yaitu IDE versi 1.6.8. Arduino IDE dapat digunakan untuk mengedit program sekaligus memprogram Arduino yang terkoneksi ke PC secara serial melalui kabel USB.



Gambar 4 Arduino IDE

2.1.3.2 NMEA dan RMC Sentence

National Marine Electronics Association (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 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

Untuk mengambil data yang diperlukan dari GPS, dilakukan penentuan jenis *sentence* yang sesuai dengan kebutuhan. Kemudian, dilakukan *parsing* data yang diperlukan. Pada implementasi sub-sistem elektrik, *sentence* yang dipakai adalah RMC, yaitu *recommended minimum data for GPS*. RMC dipakai karena data yang ada pada *sentence* ini lebih sedikit dari *sentence* lain dan sudah mencakup data GPS yang esensial untuk implementasi ini, yaitu data posisi (*longitude* dan *latitude*). Karena datanya lebih sedikit, proses pengiriman data bisa lebih cepat untuk menghemat waktu pemrosesan data GPS menjadi data posisi yang *valid*.

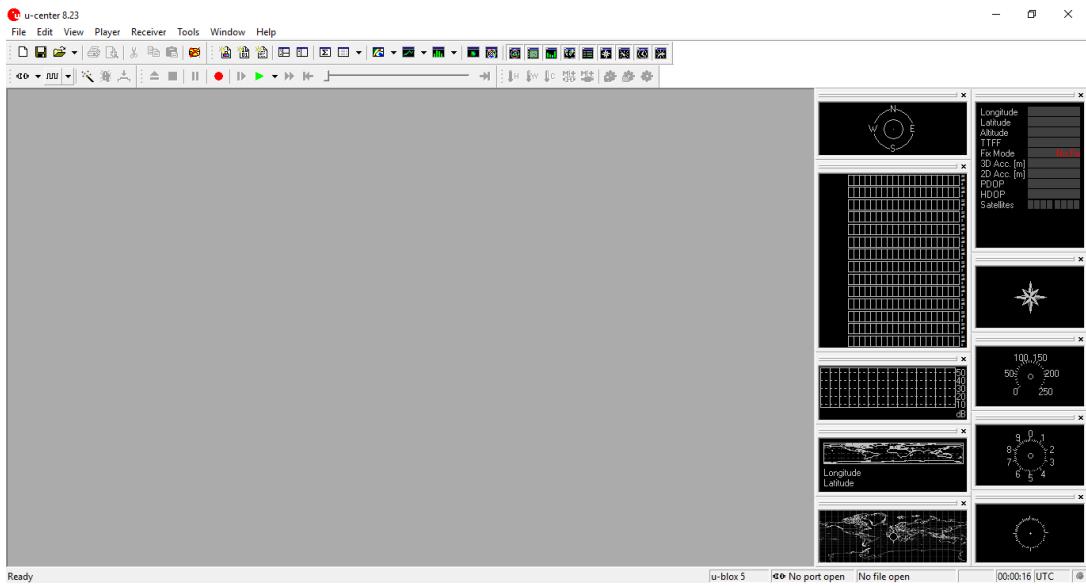
Tabel 2 Format RMC

Name	Example	Unit	Description
Message ID	\$GPRMC		RMC protocol header
UTC Time	161229.487		hhmmss.sss
Status ¹	A		A=data valid or V=data not valid
Latitude	3723.2475		ddmm.mmmm
N/S Indicator	N		N=north or S=south
Longitude	12158.3416		dddmm.mmmm
E/W Indicator	W		E=east or W=west
Speed Over Ground	0.13	knots	
Course Over Ground	309.62	degrees	True
Date	120598		ddmmyy
Magnetic Variation ²		degrees	E=east or W=west
East/West Indicator ²	E		E=east
<i>Mode</i>	<i>A</i>		<i>A=Autonomous, D=DGPS, E=DR</i>
Checksum	*10		
<CR> <LF>			End of message termination

2.1.3.3 U-Center

U-Center merupakan perangkat lunak berbasis Windows yang disediakan oleh u-blox untuk membaca dan memberikan perintah untuk GPS. Fitur-fitur yang ada pada U-Center antara lain sebagai yaitu:

1. interaktif dan mudah digunakan,
2. dapat digunakan untuk semua *receiver* GPS u-blox,
3. konfigurasi dan kendali GPS yang luas, dan
4. *real-time display* dari GPS *receiver* melalui RS232 atau USB.



Gambar 5 U-Center

Pada implementasi ini, U-Center digunakan untuk melakukan pengecekan awal fungsionalitas GPS *receiver* dan konfigurasi GPS *receiver* (seperti *baudrate* dan satelit).

2.1.3.4 AT Command

AT *Command* merupakan perintah untuk mesin yang terdiri dari serangkaian karakter dan diakhiri dengan tanda *carriage return* (<cr>) dan *line feed* (<lf>). Setiap *command* yang diberikan ke suatu mesin akan direspon oleh mesin tersebut dengan respon-respon yang unik bergantung pada *command* yang diberikan.

AT Command pada sub-sistem elektrik digunakan untuk memberikan perintah ke modul internet GSM. Perintah-perintah yang digunakan antara lain seperti pada tabel 3.

Tabel 3 Beberapa AT Command

Command	Fungsi
AT	<i>Command</i> dasar untuk mengecek kondisi modul GSM
AT+IPR	Mengatur <i>baudrate</i> komunikasi antara modul GSM dengan Arduino
AT+CREG	Melihat kondisi kartu SIM yang ada di modul GSM apakah sudah teregistrasi atau belum
AT+CIPMUX	Mengatur jumlah koneksi
AT+CIPMODE	Mengatur mode koneksi TCP/IP
AT+CIPSTATUS	Melihat kondisi koneksi yang sedang dibuka (atau tidak ada koneksi yang terbuka)
AT+CSTT	Mengatur APN, <i>user name</i> dan <i>password</i> untuk mengaktifkan mode internet pada kartu SIM
AT+CIICR	Mengaktifkan koneksi GPRS atau CSD
AT+CIFSR	Memperoleh alamat IP <i>client</i>

AT+CIPSTART	Memulai koneksi TCP atau UDP dengan suatu <i>server</i>
AT+CIPSHUT	Menutup koneksi TCP atau UDP dengan suatu <i>server</i>
AT+CIPSEND	Mengirim paket data ke <i>server</i> yang telah dibuka koneksinya dengan <i>command</i> AT+CIPSTART

Pada AT *Command*, terdapat beberapa *command* yang harus diikuti oleh parameter-parameter tertentu. Sebagai contoh, *command* AT+CIPSTART harus diikuti oleh jenis koneksi, alamat *server* dan *port* koneksinya (dengan format AT+CIPSTART="TCP","server_name","port"). *Command* seperti ini biasanya digunakan untuk melakukan pengaturan koneksi. Selain itu juga ada *command* yang digunakan untuk melihat kondisi koneksi yang sudah terjadi. *Command* seperti ini biasanya dipanggil tanpa membutuhkan parameter-parameter tertentu dan akan merespon dengan kondisi saat *command* dipanggil. Sebagai contoh, AT+CIPSTATUS akan direspon dengan kondisi koneksi modul.

2.1.3.5 Komunikasi Serial

Pada sub-sistem elektrik, Arduino berkomunikasi dengan modul internet GSM dan modul pelacak posisi GPS secara serial. Komunikasi serial ini terjadi melalui dua UART berbeda pada Arduino dengan UART pada kedua modul tersebut.

Selain untuk komunikasi Arduino dengan kedua modul, pemrograman Arduino dari PC juga menggunakan komunikasi serial. Komunikasi antara PC dengan Arduino terjadi melalui kabel USB Type-B.

2.1.3.6 Komunikasi SPI

Pada sub-sistem elektrik, Arduino berkomunikasi dengan modul akuisisi data CAN menggunakan komunikasi SPI. Komunikasi SPI ini terjadi melalui pin SPI (MISO, MOSI dan SCK) pada Arduino dan modul akuisisi data CAN.

2.1.3.7 Komunikasi I²C

Pada sub-sistem elektrik, Arduino berkomunikasi dengan modul *display* menggunakan komunikasi I²C. Komunikasi ini terjadi melalui pin I²C (SDA dan SCL) pada Arduino dan modul *display*.

2.1.3.8 Protokol Message Queuing Telemetry Transport (MQTT)

Protokol MQTT merupakan salah satu protokol komunikasi *machine to machine* (M2M) yang berbasis *lightweight data* dan metode *publish-subscribe*. Protokol MQTT digunakan pada komunikasi sub-sistem elektrik dengan *server*. Protokol ini akan dijelaskan lebih rinci pada bagian implementasi *server*.

2.1.4 Prosedur Implementasi

Dalam melakukan implementasi, dilakukan beberapa tahap implementasi sebagai berikut.

1. Melakukan pemilihan jenis perangkat keras di antara alternatif desain yang telah dijabarkan pada bagian desain.
2. Melakukan pemilihan metode implementasi dari jenis perangkat keras yang telah dipilih.

3. Melakukan pemilihan perangkat lunak yang digunakan untuk mengembangkan sub-sistem elektrik.
4. Mendesain pengujian fungsionalitas tiap modul pada sub-sistem elektrik.
5. Menguji fungsionalitas tiap modul pada sub-sistem elektrik.
6. Mengintegrasikan seluruh modul menjadi satu kesatuan prototipe sub-sistem elektrik.
7. Melakukan pengujian, *debugging*, *troubleshooting* dan revisi terhadap prototipe.
8. Melakukan manufaktur sub-sistem elektrik.

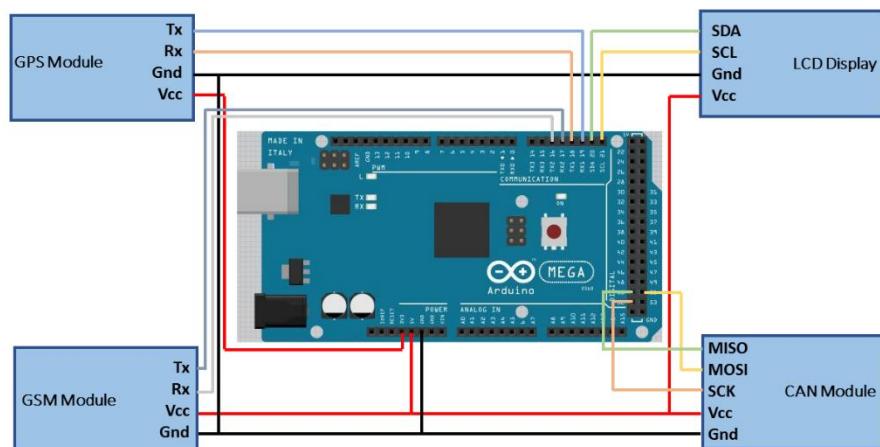
Sub-sistem elektrik terdiri dari modul-modul yang telah dijelaskan pada bagian sebelumnya. Pada tahap 1-5, implementasi dilakukan untuk tiap modul pada sub-sistem elektrik. Pada tahap 6-8, implementasi dilakukan sudah berupa penggabungan seluruh modul. Berikut penjelasan implementasi dan permasalahan yang terjadi selama proses implementasi sub-sistem elektrik.

2.1.4.1 Implementasi Kontroller

Pada sub-sistem elektrik, dipilih Arduino Mega 2560 sebagai kontroller. Arduino Mega 2560 memiliki spesifikasi teknis yang memenuhi spesifikasi desain yang diperlukan, antara lain jumlah UART (minimal tersedia dua), 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 sub-sistem elektrik 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 sub-sistem elektrik akan diuji dan dikendalikan menggunakan kontroller ini. Modul-modul yang digunakan akan memanfaatkan komunikasi SPI, I²C dan serial pada Arduino.



Gambar 6 Skematik implementasi hardware

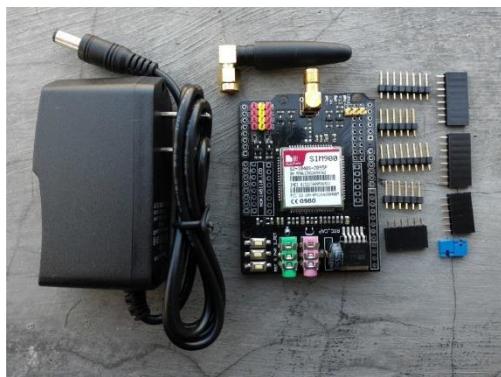
Gambar 6 menunjukkan skematik implementasi integrasi *ECU monitoring*. Keempat sub-modul dihubungkan ke pin-pin tertentu pada kontroller Arduino Mega.

Sub-modul GSM dan GPS menggunakan 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 I²C. Pada implementasinya, komunikasi I²C pada Arduino Mega menggunakan dua pin yaitu SDA dan SCL.

2.1.4.2 Implementasi Komunikasi *Hardware* dengan Server

Pada implementasi modul ini, dilakukan beberapa tahap untuk memilih perangkat dan melakukan pengujian fungsionalitas perangkat. 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*.

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 sehingga *debugging* dan *troubleshooting* dapat dilakukan dengan lebih mudah.



Gambar 7 SIM900 EFCOM Shield

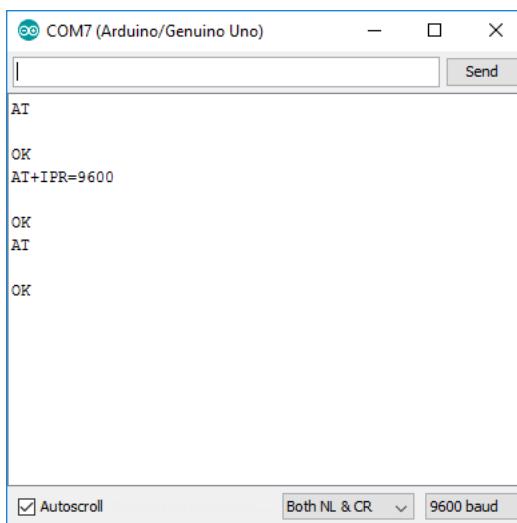
Pengujian fungsionalitas dan implementasi modul ini dilakukan secara bertahap sebagai berikut.

2.1.4.2.1 Pengujian *AT Command* secara Manual

Pada pengujian tahap pertama, dilakukan pengujian *AT Command* secara manual menggunakan bantuan Arduino untuk menghubungkan PC dengan modul GSM SIM900. Pin receive (Rx) pada Arduino dihubungkan dengan pin Rx pada modul GSM, sedangkan pin transmit (Tx) pada Arduino dihubungkan dengan pin Tx pada modul GSM. Pin-pin ini dihubungkan dengan tujuan agar data yang ditransmit oleh PC akan diteruskan ke pin Rx modul GSM. Kemudian, modul GSM akan membala *AT Command* yang ditransmit oleh PC tersebut melalui pin Tx, yang kemudian ditampilkan di serial monitor pada PC. *AT Command* yang dikirimkan pada tahap ini hanya command dasar seperti AT dan AT+IPR.

AT+IPR berfungsi untuk mengatur *baudrate* SIM900. Pada keadaan *default*, *baudrate* SIM900 diatur menjadi *autobaud* yaitu *baudrate* secara otomatis diatur sesuai dengan yang

dikirimkan oleh pengirim *command*. Untuk Command AT dan AT+IPR, respon dari SIM900 adalah OK. Jika pada serial monitor terbaca OK, SIM900 berfungsi dengan baik.



Gambar 8 Pengujian AT *Command* manual

Pengujian tahap ini dianggap selesai jika AT *Command* berhasil dikirimkan ke modul GSM dan responnya dapat terbaca benar melalui serial *monitor*.

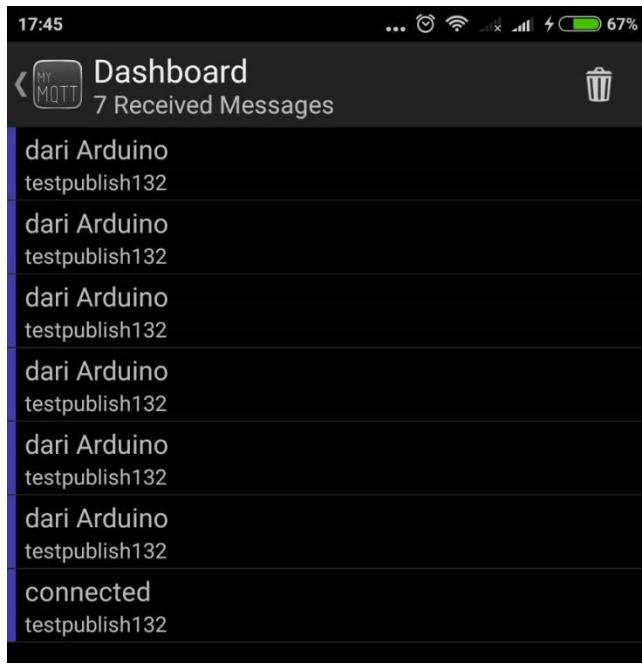
2.1.4.2.2 Pengujian AT *Command* Menggunakan Arduino Uno dan SoftSerial

Pengujian tahap ini dilakukan dengan memprogram Arduino untuk mengirimkan AT Command ke modul GSM SIM900. Pin Tx pada modul GSM dihubungkan dengan pin Rx SoftSerial pada Arduino, sedangkan pin Rx pada modul GSM dihubungkan dengan pin Tx SoftSerial pada Arduino. SoftSerial digunakan agar debugging dapat dilakukan dengan melihat respon modul GSM untuk tiap command yang diberikan oleh Arduino melalui UART pada Arduino. Command yang diberikan pada tahap ini cukup menggunakan command dasar seperti sebelumnya.

Pengujian tahap ini dianggap selesai jika AT *Command* berhasil dikirimkan ke modul GSM dan responnya dapat terbaca benar melalui serial *monitor*.

2.1.4.2.3 Pengujian Komunikasi dengan Server MQTT

Setelah berhasil membuat program pada Arduino untuk mengirimkan AT *Command*, berikutnya yaitu mencoba berkomunikasi dengan *server*. *Server MQTT* yang digunakan pada pengujian ini yaitu HiveMQ MQTT Broker. Koneksi perangkat yang digunakan sama dengan koneksi pada pengujian sebelumnya. Command yang diberikan adalah command untuk membuka koneksi TCP ke *server*, kemudian command untuk mengirimkan paket data sesuai dengan protokol komunikasi MQTT (protokol komunikasi MQTT dijelaskan lebih detil pada bagian implementasi *server*).



Gambar 9 Hasil pengiriman data dari Arduino

Untuk membuka dan memanfaatkan koneksi MQTT dengan modul GSM SIM900 dan Arduino, digunakan *library* MQTT yang dimodifikasi sehingga sesuai dengan modul GSM yang digunakan. Program pada *smartphone* yang digunakan untuk pengujian ini merupakan MQTT Client pada smartphone, yaitu MyMQTT.

Pada pengujian ini, dilihat respon modul GSM untuk tiap AT *Command* yang diberikan dan dilihat pula keberhasilan *publish* dan *subscribe*. Pengujian tahap ini dianggap selesai jika broker MQTT menerima pesan sesuai dengan yang dikirimkan dan Arduino dapat menangkap pesan dari topik yang telah di-*subscribe* (pesan yang diterima dicetak kembali ke serial *monitor*).

2.1.4.2.4 Implementasi pada Arduino Mega 2560

Setelah melakukan tahapan-tahapan pengujian fungsional pada bagian sebelumnya, dilakukan implementasi modul GSM SIM900 pada Arduino Mega 2560. Untuk implementasi, pin Tx pada modul GSM dihubungkan dengan pin Rx pada Arduino, sedangkan pin Rx pada modul GSM dihubungkan dengan pin Tx pada Arduino. Port serial pada Arduino yang digunakan adalah port hardware serial agar lebih stabil.

```
/* 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, 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 *ECU monitoring* 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 *ECU*

monitoring 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).

2.1.4.2.5 Permasalahan dan Solusi

Berikut adalah permasalahan yang mungkin terjadi dalam pengembangan implementasi modul GSM SIM900 dengan Arduino.

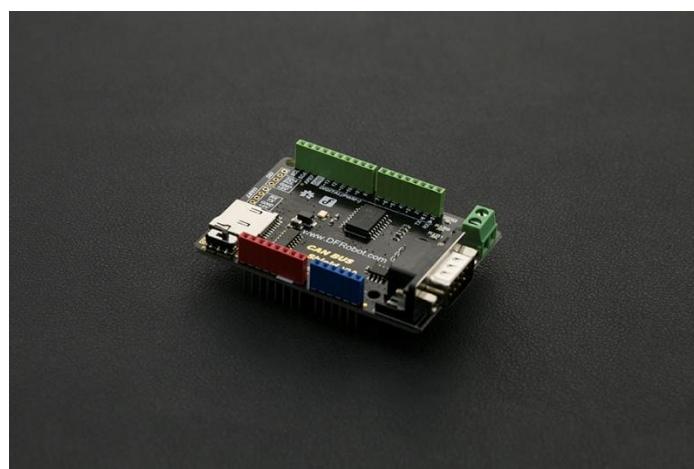
Tabel 4 Permasalahan dan solusi modul GSM

Permasalahan	Solusi
Kesulitan sinyal GSM di tempat yang memang sulit terjangkau sinyal	Menggunakan <i>provider</i> kartu SIM yang memiliki jaringan cukup bagus
Modul GSM tidak terkoneksi ke internet walaupun sudah menggunakan <i>library</i>	Mengubah isi <i>library</i> agar dapat dipakai oleh perangkat keras yang digunakan
Tidak dapat implementasi fungsi <i>subscribe</i> dengan Arduino	Mengubah isi <i>library</i> agar dapat dipakai oleh perangkat keras yang digunakan

2.1.4.3 Implementasi Modul Akuisisi Data dari CAN Bus

Modul akuisisi data CAN merupakan modul yang berfungsi untuk mengambil data baterai fleet dari CAN Bus yang ada pada fleet. CAN Bus ini digunakan untuk komunikasi machine to machine dalam satu fleet.

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



Gambar 10 DFRobot CAN Bus *shield*

CAN Bus ini dapat menggunakan komunikasi serial maupun SPI. Pada implementasi sub-sistem elektrik ini, digunakan komunikasi SPI karena komunikasi serial utama kontrollernya sudah terpakai oleh modul GSM.

2.1.4.3.1 Pembacaan Modul CAN Bus dengan Arduino

Tahap pengujian pertama sebelum implementasi akuisisi data CAN Bus adalah membaca modul CAN Bus dengan Arduino. Untuk pengujian ini, *shield* dipasangkan ke Arduino untuk kemudian dites komunikasi *chip* dengan Arduino.

2.1.4.3.2 Pembacaan Data dari CAN Bus Prototipe *Guided Bus*

Setelah Arduino berhasil berkomunikasi dengan CAN Bus *shield*, berikutnya adalah membaca data dari CAN Bus prototipe *guided bus* yang ada di LEN. Data dari CAN Bus ini terdiri dari beberapa parameter, antara lain CAN ID dan data.

2.1.4.3.3 Implementasi Modul CAN Bus pada Prototipe *Guided Bus*

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 (FAILUREZERODE)	0	0	0	0	0
				0	0	0	0	0	0
	BMS	ID 2	FAULT_REG						
	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
	ECU	ID 4							
2	ECU to Inv	Mode	Throttle 0: SensorInIt 2: ZeroDeg 1: Neutral 4: Forward 8: Reverse	Brake 0 - 100	0	0	0	0	0
3	Inv to ECU	Mode Status	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
			1: ZeroDeg 2: Neutral 3: Forward 4: Reverse						
4	ECU to BMS	Mode	Mode 1: Close Contactor 0: Open Contactor	1. Charge 0. Idle	0	0	0	0	0
5	BMS to ECU	1	V_BAT Byte0	V_BAT Byte1	LBAT Byte 0	LBAT Byte 1	V_HVDC Byte 0	V_HVDC Byte 1	0
			2	SOC	g	BMS_STATE	0	0	0
16	ECU to DC-DC 1	Status	0	0	0	0	0	0	0
			1: Comm OK						
26	ECU to DC-DC 2	Status	0	0	0	0	0	0	0
			1: Comm OK						
7	DC-DC 1 to ECU	Status	1	0	0	0	0	0	0
			1: OK 2: Auto Restart						
7	DC-DC 2 to ECU	Status	2	0	0	0	0	0	0
			1: OK 2: Auto Restart						
8	ECU to BCU								
9	BCU to ECU								

Gambar 11 Isi data CAN Mini AGT LEN

Pada implementasinya, modul CAN ini akan digunakan untuk membaca data yang dibutuhkan saja dari CAN Bus, tidak semua data yang akan diolah. Gambar 10 merupakan daftar CAN ID dan data yang ada pada ID tersebut.

```
/* 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_500KBPS))
{
```

```

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_500KBPS))` berfungsi untuk memastikan modul CAN Bus dapat digunakan, dan memulai komunikasi dengan CAN Bus lain dengan *baudrate* 500kbps. 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.

2.1.4.4 Implementasi Modul Pelacak Lokasi GPS

Pada implementasi modul ini, dilakukan beberapa tahap untuk memilih perangkat dan melakukan pengujian fungsionalitas perangkat. Perangkat yang digunakan yaitu GPS UBlox Neo-8M. GPS ini dapat memenuhi spesifikasi yang dibutuhkan dengan akurasi 2.5 meter, update rate hingga 10 Hz, komunikasi dengan UART dan dapat membaca sinyal dari satelit GLONASS yang sangat membantu dalam meningkatkan presisi dan akurasi pembacaan GPS di Indonesia.

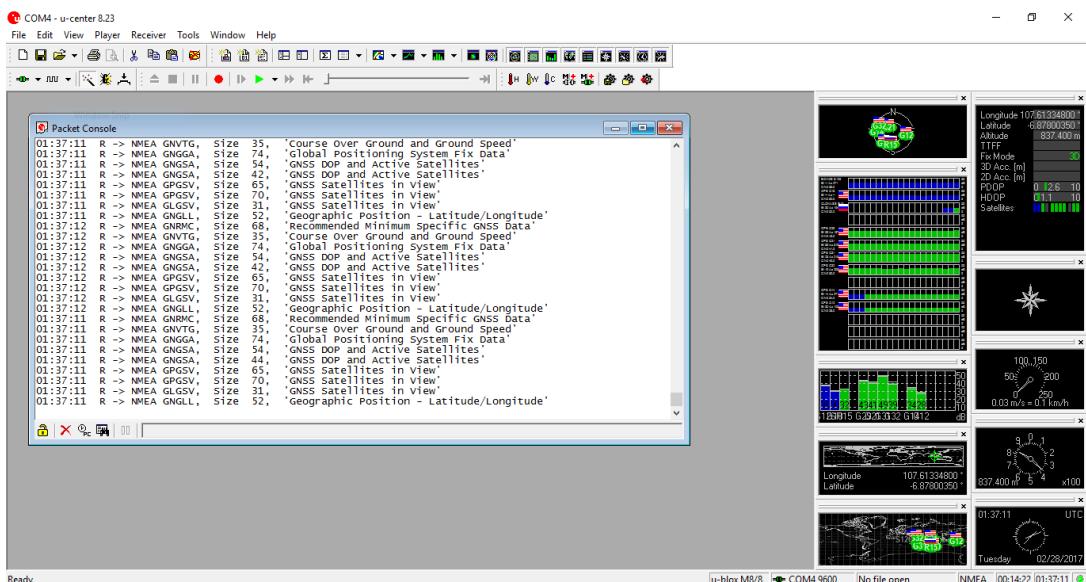
Implementasi modul ini dilakukan secara bertahap sebagai berikut.

2.1.4.4.1 Pembacaan dan Setting Modul GPS Melalui U-Center

Untuk melakukan pengujian fungsional modul GPS dan untuk memenuhi spesifikasi yang telah dijelaskan pada dokumen sebelumnya, diperlukan sebuah perangkat lunak tambahan yaitu U-Center.

Pada bagian ini, pertama akan dilakukan pengujian fungsionalitas modul GPS yang akan digunakan. Pengujian fungsionalitas ini meliputi pengamatan satelit yang terdeteksi, sentence yang terkirim/terbaca, sampai pada akhirnya dilihat keberhasilan pembacaan lokasi oleh modul GPS. Setelah lokasi berhasil terbaca pada U-Center, kemudian dilakukan pengamatan dan pengaturan untuk memenuhi spesifikasi sub-sistem elektrik. Pengamatan yang dilakukan yaitu mengamati *error* pembacaan lokasi oleh GPS. Pengaturan yang dilakukan yaitu pengaturan *baudrate*, *refresh rate* dan *sentence* yang akan dikirimkan.

Pertama, modul GPS dihubungkan dengan sebuah USB to Serial Converter untuk dapat berkomunikasi dengan U-Center. Setelah itu, U-Center dibuka dan mulai koneksi dengan modul GPS dengan *autobauding*. Perlu dipastikan antenna GPS berada di tempat yang cukup terbuka agar mendapat sinyal dari satelit yang ada. Kemudian, modul GPS akan berusaha mencari sinyal dari satelit-satelit yang ada. Proses ini membutuhkan waktu cukup lama jika dilakukan pertama kali setelah modul GPS dalam keadaan mati, sekitar satu menit. Setelah modul GPS dapat me-*lock* sinyal dari satelit-satelit yang ada, akan muncul tampilan sebagai berikut.

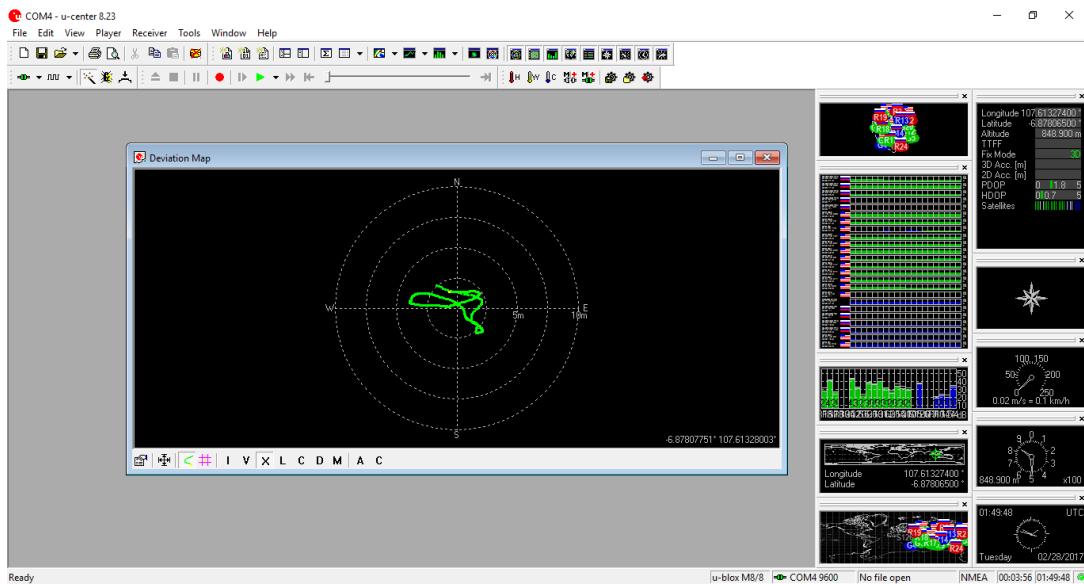


Gambar 12 GPS Fix

Pada bagian kanan U-Center dapat dilihat jumlah, kekuatan sinyal dan jenis satelit yang terkoneksi, arah modul GPS, *longitude* dan *latitude* modul GPS berada. Untuk melakukan

pengecekan kebenaran informasi lokasi ini, dapat digunakan Google Maps. Koordinat yang diperoleh dimasukkan ke dalam Google Maps kemudian dilihat posisinya.

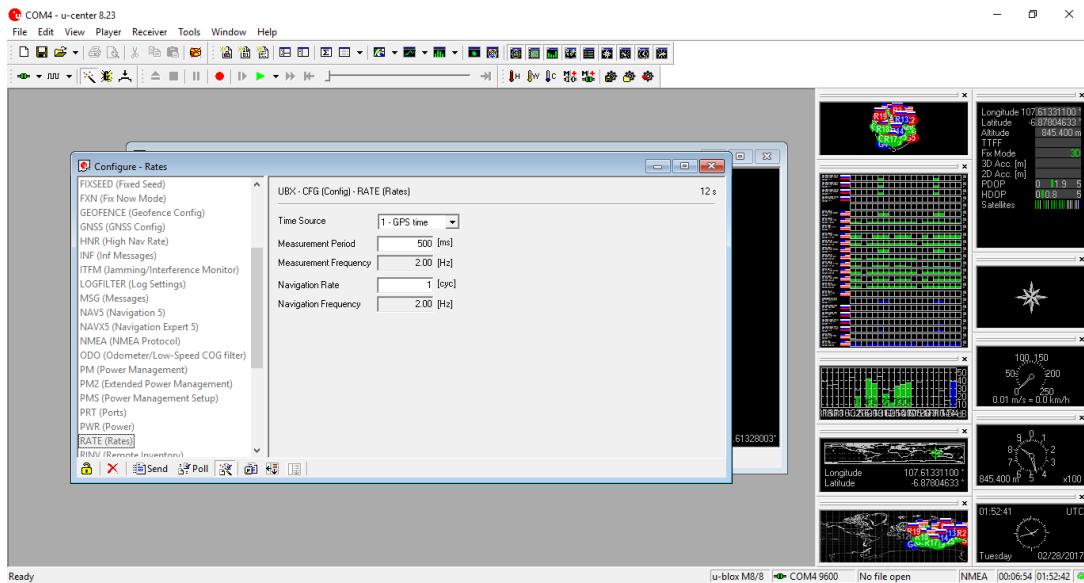
Error GPS dapat dilihat melalui bagian *deviation map*. Bagian ini ada pada menu *view*.



Gambar 13 Deviation map

Pada gambar 12 dapat dilihat penyimpangan data lokasi yang terbaca berkisar 0 sampai 5 meter.

Setelah data dapat terbaca dan divalidasi, berikutnya adalah melakukan pengaturan-pengaturan yang diperlukan untuk memenuhi spesifikasi. Spesifikasi yang akan dipenuhi pada bagian ini yaitu *refresh rate* GPS.



Gambar 14 Pengaturan refresh rate

Refresh rate dapat diatur pada bagian *configure* di menu *view->configuration menu*. *Refresh rate* yang digunakan yaitu 5 Hz (periode 0.2 detik) untuk memenuhi spesifikasi.

Pengaturan ini akan dilakukan setiap kali modul dinyalakan. Untuk melakukan pengaturan ini, dapat dilakukan dengan cara mengirimkan perintah berupa urutan *bytes*. Perintah ini

dikirimkan oleh kontroller. terdapat dua perintah yang akan dikirimkan ke modul GPS, yaitu untuk *refresh rate* 5Hz dan mengganti *baudrate* menjadi 57600.

```
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, 0x00, 0xDE, 0xC9 };
```

2.1.4.4.2 Pembacaan Modul GPS Melalui UART Arduino

Tahap berikutnya yaitu melakukan pembacaan data GPS melalui UART Arduino. Setelah dilakukan setting pada U-Center, GPS akan selalu memberikan data secara serial sesuai dengan *baudrate* yang telah ditentukan.

Pin Rx pada GPS dihubungkan ke pin Tx Arduino, sedangkan pin Tx pada GPS dihubungkan ke pin Rx Arduino. Setelah itu, serial monitor Arduino dibuka dan diatur *baudrate* yang sesuai untuk dilihat hasil pengiriman data oleh modul GPS. Tampilan pada serial monitor Arduino seharusnya akan sama dengan gambar 15.

Pengujian tahap ini dianggap selesai jika data GPS berhasil dibaca di serial monitor Arduino.

2.1.4.4.3 Implementasi pada Arduino Mega 2560

Pada implementasinya, sentence yang diterima Arduino harus dipilah (*parsing*) untuk hanya mengambil nilai koordinatnya. Parsing data GPS ini memanfaatkan library Ublox (library dari referensi dan kode terlampir).

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.

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, 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 */
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’.

Delay di bagian awal digunakan untuk mengatur frekuensi pengiriman data ke server. Nilai delay bergantung pada *processing time* keseluruhan kode, sehingga dapat dipastikan data dikirimkan setiap 500 detik.

2.1.4.4.4 Permasalahan dan Solusi

Tabel 5 Permasalahan dan solusi modul GPS

Permasalahan	Solusi
Tempat penggerjaan tugas akhir berada di <i>basement</i> sehingga kesulitan sinyal GPS	Mencoba GPS di tempat berbeda yang mudah mendapat sinyal GPS
USB to Serial Converter untuk pengaturan GPS pada U-Center tidak dapat terdeteksi oleh PC	Menginstall <i>driver</i> untuk USB to Serial Converter yang sesuai
Data GPS tidak diambil oleh Arduino secara <i>real-time</i>	Menambah <i>buffer size</i> Arduino menjadi 256 byte (maksimal)

2.1.4.5 Implementasi Modul LCD 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 menyalaikan 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.

2.1.4.6 Implementasi Persiapan Data dan Enkripsi AES

Sebelum dikirimkan ke server, data-data yang diperoleh dari server diolah oleh *ECU monitoring* menjadi satu kesatuan paket data yang siap dikirim ke server. Setelah menjadi satu paket data, data ini dienkripsi dengan metode enkripsi AES.

Data dari sensor-sensor pertama digabungkan menjadi satu kesatuan paket data. Urutan data yang akan disiapkan yaitu longitude, latitude, energi baterai, suhu baterai dan ID. Kelima data ini kemudian digabungkan menjadi satu kesatuan paket data yang dipisahkan dengan tanda titik dua untuk setiap datanya.

Setelah menjadi satu kesatuan paket data, data ini kemudian dienkripsi dengan metode enkripsi AES. Pada implementasinya, enkripsi AES pada Arduino memanfaatkan *library* AESlib (sumber di bagian referensi, kode dan *library* terlampir). *Library* ini menyediakan dua jenis enkripsi AES yang dapat digunakan, yaitu enkripsi 128bit dan 256bit. Untuk implementasi *fleet monitoring and control system* ini menggunakan enkripsi 128bit.

```

/* deklarasi */
#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, 0x41};
byte plain[32];
byte cipher[48];

void prekey (int bits)
{
    for (int 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, 0x41};
    aes.do_aes_encrypt(plain, 41, cipher, key, bits, iv);
}

void prekey_test ()
{
    prekey(128);
}

/* program utama */
prekey_test();
for (int i=0; i<=31; i++)
    packetSend[i] = (char)(cipher[i]);

```

Kode di atas merupakan implementasi enkripsi AES pada Arduino Mega 2560. Isi dari fungsi-fungsi yang digunakan terlampir.

Pada bagian awal kode dideklarasikan penggunaan *library* AES.h dan AES_config.h dengan menggunakan fungsi #include. AES aes berfungsi untuk membuat *class* aes yang akan digunakan pada program utama. Byte key digunakan sebagai key saat melakukan enkripsi AES. Byte plain digunakan untuk menampung data yang akan dienkripsi, sedangkan byte cipher digunakan untuk menampung data setelah dienkripsi.

Fungsi prekey_test merupakan fungsi yang digunakan untuk memanggil fungsi prekey. Fungsi prekey merupakan fungsi untuk mempersiapkan data yang dibutuhkan untuk enkripsi. Data tersebut yaitu plain dan initial vector. Variabel plain diisi sama persis dengan variabel data, sedangkan initial vector (iv) yang dideklarasikan dengan bute iv diisi dengan initial vector yang akan digunakan dalam enkripsi AES.

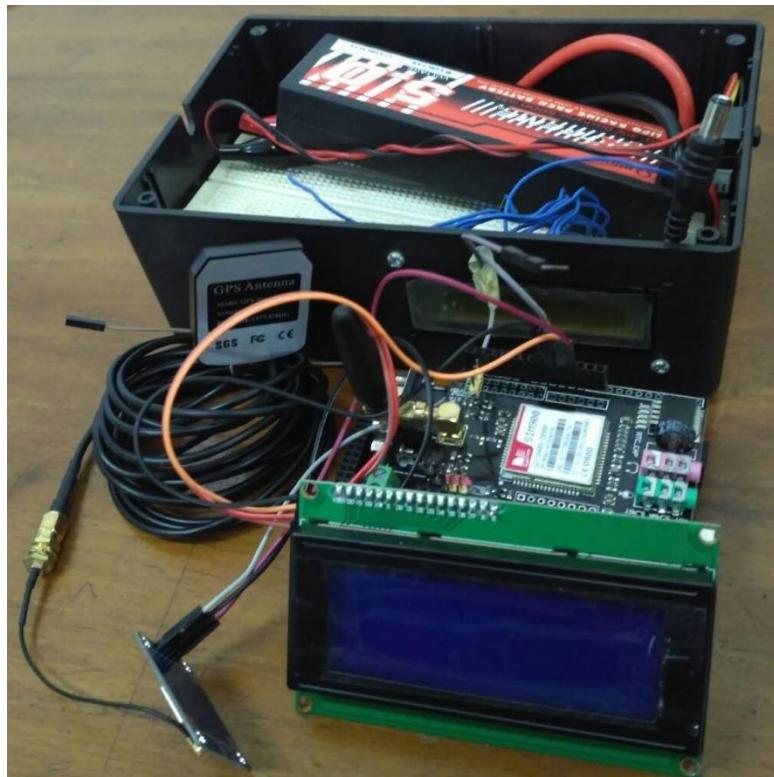
Fungsi do_aes_encrypt merupakan fungsi yang melakukan enkripsi data. Data yang dienkripsi berasal dari variabel plain, sedangkan hasil enkripsi dimasukkan ke variabel cipher.

Pada program utama, fungsi prekey_test dipanggil untuk melakukan enkripsi sehingga variabel cipher berisi data yang telah dienkripsi. Setelah itu, pada bagian iterasi, data pada

cipher dikonversi ke tipe data char dan dimasukkan ke variabel packetSend untuk dapat dikirimkan ke server. Variabel packetSend ini yang menjadi parameter dalam fungsi *publish* pada implementasi modul GSM SIM900.

2.1.4.7 Integrasi Sub-Sistem Elektrik

Pada integrasi keseluruhan sub-sistem elektrik, dibuat sebuah prototipe yang dapat melakukan fungsional tiap modul sesuai spesifikasi *ECU monitoring* yang dibutuhkan. Setelah itu, dilakukan pembuatan PCB tambahan untuk memasang LCD dan modul GPS. PCB ini digunakan sebagai shield yang dipasangkan ke Arduino Mega 2560 untuk langsung menghubungkan LCD dan modul GPS dengan Arduino Mega 2560.

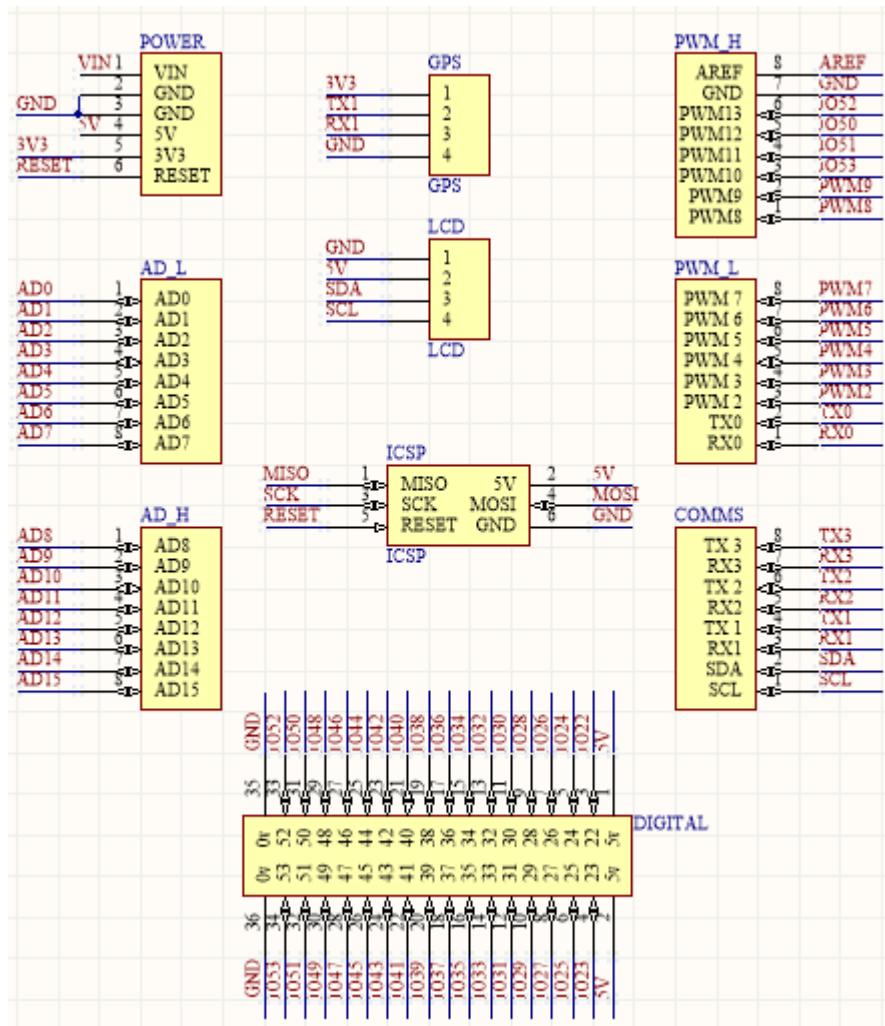


Gambar 15 Prototipe sub-sistem elektrik

Setelah membuat *shield*, Arduino Mega 2560, CAN Bus, GSM SIM900 dan *shield* digabungkan. Kemudian, dibuat implementasi kode untuk melakukan fungsi-fungsi dan memenuhi spesifikasi yang harus dilakukan *fleet monitoring hardware*. Kode yang dibuat terlampir.

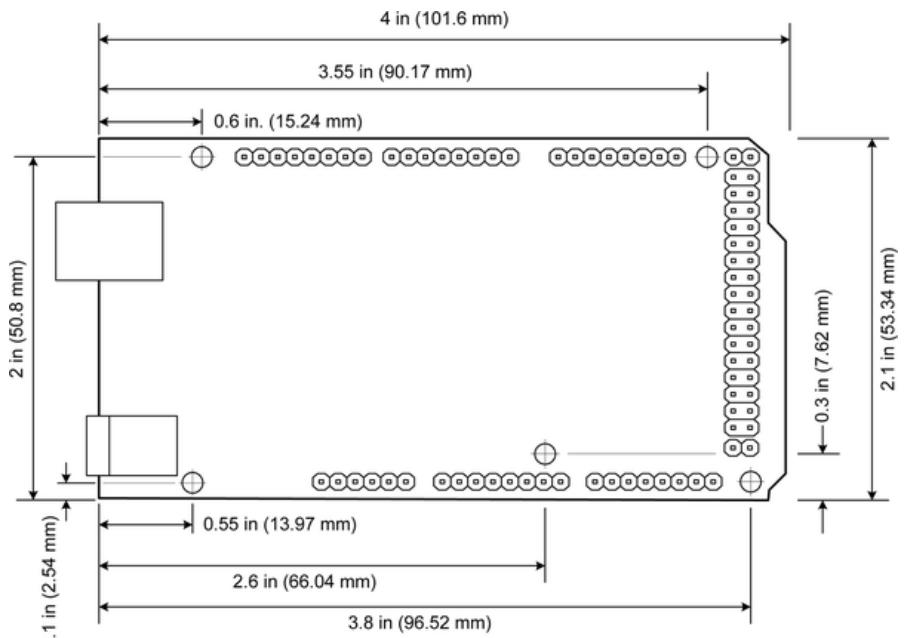
2.1.4.7.1 Desain Skematik dan PCB *Shield*

Tahap pertama pada integrasi sub-sistem elektrik adalah membuat 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. Berikut skematik yang digunakan untuk membuat shield Arduino Mega 2560.

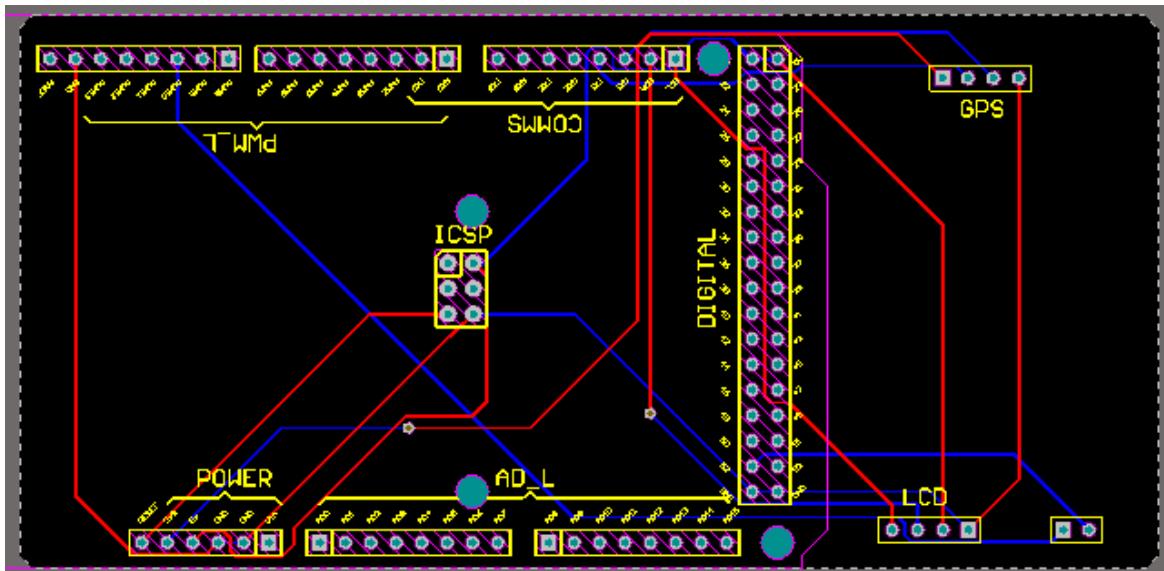


Gambar 16 Skematik shield

Kemudian setelah skematik selesai, dibuat implementasi desain PCBnya. Untuk melakukan implementasi desain PCB, diperlukan dimensi Arduino Mega 2560 agar sesuai. Berikut dimensi Arduino Mega 2560 dan desain PCB yang dibuat.



Gambar 17 Dimensi Arduino Mega 2560



Gambar 18 Desain PCB shield

2.1.4.7.2 Implementasi Integrasi Keseluruhan

Implementasi kode-kode pada bagian sebelumnya digabungkan pada tahap ini. Setiap bagian kode yang ada pada bagian “setup” dimasukkan ke dalam fungsi void setup() pada Arduino. Setiap bagian kode yang ada pada bagian “program utama” dimasukkan ke dalam fungsi void loop() pada Arduino. Setiap bagian kode yang ada pada bagian “deklarasi” diletakkan sebelum fungsi void setup() pada Arduino. Kode yang dibuat terlampir.



Gambar 19 ECU Monitoring

2.1.4.7.3 Permasalahan dan Solusi

Pada implementasinya, permasalahan pada sub-sistem elektrik ini umumnya terletak pada saat diintegrasikan dengan sub-sistem server. Berikut permasalahan yang dapat terjadi pada integrasi sub-sistem elektrik ini.

Tabel 6 Permasalahan dan solusi integrasi sub-sistem elektrik

Permasalahan	Solusi
Data dari sinyal GPS tidak bisa disimpan terlalu lama di memori Arduino sedangkan proses lain juga membutuhkan waktu	Delay 600 ms yang sebelumnya diimplementasikan untuk spesifikasi pengiriman data diubah menjadi delay yang lebih kecil
Byte data terenkripsi yang dikirimkan dari Arduino tidak sesuai dengan yang terbaca di server	Memastikan tipe data dan ukuran yang digunakan pada Arduino sudah benar

2.2 IMPLEMENTASI SUB-SISTEM SERVER

Server merupakan penghubung antara *fleet* dengan *ground control*. Data informasi dari *fleet* akan di *publish* ke melalui jaringan *internetserver* setelah itu server meneruskan data ke *ground control*. Dalam implementasi kali ini digunakan *laptop* Lenovo idepad Z410 dengan spesifikasi sebagai berikut:

- RAM 8GB
- Intel core i5-4200 @ 2.5 GHz
- Windows 10 Education 64-bit
- NVidia GT740m

2.2.1 Struktur Implementasi



Gambar 20 DFD level 2 server

Server yang digunakan berbasis MQTT dari mosquitto. Mosquitto menyediakan *server* MQTT gratis yang dapat di *install* pada platform Windows maupun Linux.

2.2.2 Environment

Dalam implementasi server ini digunakan perangkat lunak sebagai berikut

- Aplikasi Mosquitto 1.4.11 sebagai server MQTT
- OpenSSL
- Posix threads

2.2.3 Prosedur Implementasi

Untuk membuat komputer menjadi server MQTT terdapat beberapa langkah sebagai berikut

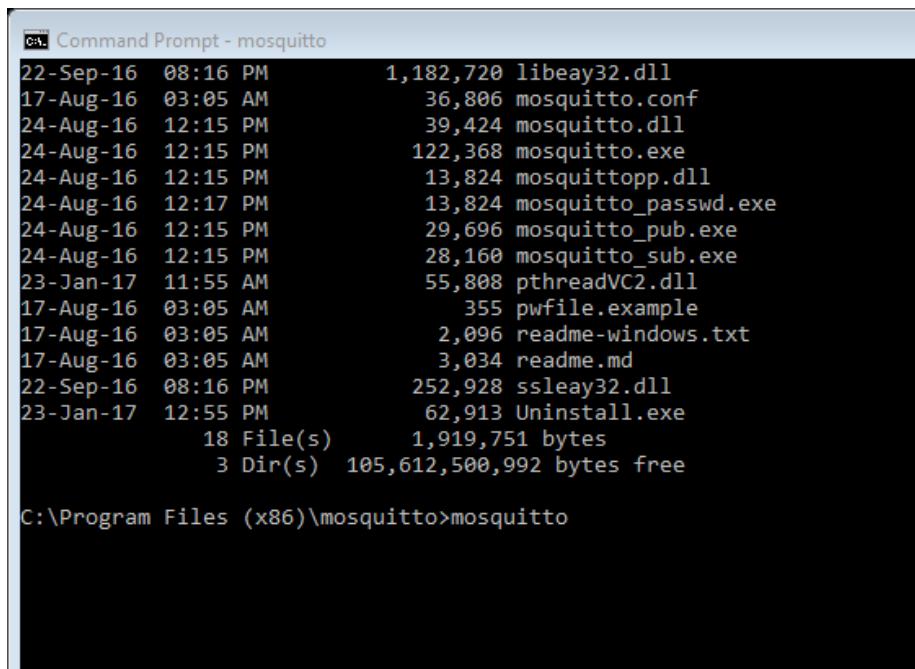
- *Install* MQTT dan mulai MQTT *service*
- *Port-forwarding* IP komputer server

2.2.3.1 Install MQTT dan mulai MQTT service

Terdapat beberapa langkah dalam memasang *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



```

22-Sep-16 08:16 PM      1,182,720 libeay32.dll
17-Aug-16 03:05 AM      36,806 mosquito.conf
24-Aug-16 12:15 PM      39,424 mosquito.dll
24-Aug-16 12:15 PM      122,368 mosquito.exe
24-Aug-16 12:15 PM      13,824 mosquitopp.dll
24-Aug-16 12:17 PM      13,824 mosquito_passwd.exe
24-Aug-16 12:15 PM      29,696 mosquito_pub.exe
24-Aug-16 12:15 PM      28,160 mosquito_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

```

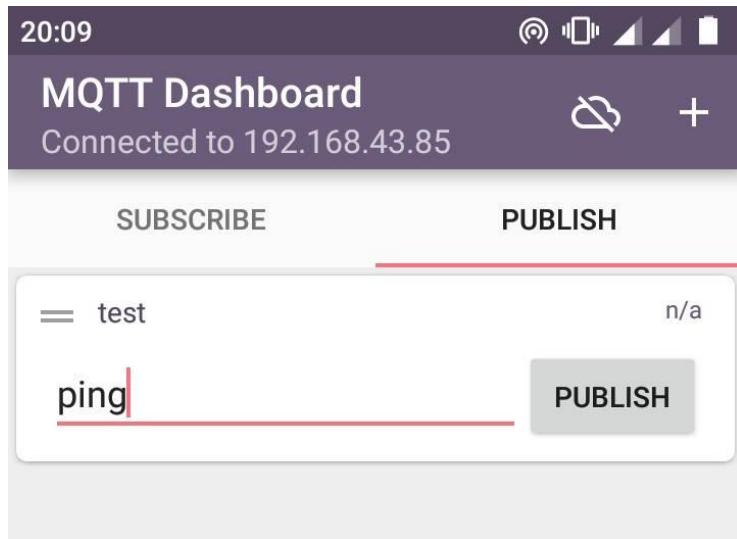
Gambar 21 menjalankan server melalui Command Prompt

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 test *server* MQTT melalui aplikasi *publish* dan *subscribe* MQTT pada *smartphone*. Dalam uji coba ini digunakan aplikasi “IoT MQTT dashboard”, uji coba dapat dilakukan dengan aplikasi serupa.

Pada aplikasi ubah pengaturan server menjadi

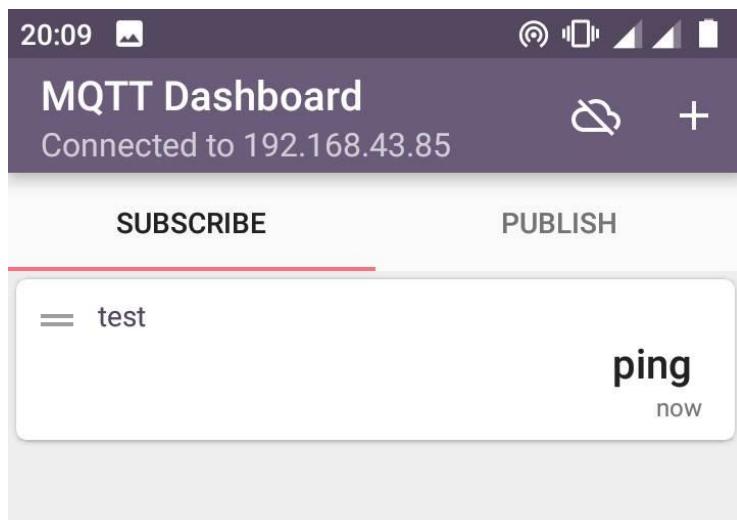
- *IP server:* 192.168.43.85 (sesuai dengan IP local dari komputer, dapat dilihat pada Command Prompt dengan perintah “ipconfig” cari pada bagian “IPv4 address”)
- *Port:* 1883 (port yang digunakan server MQTT pada komputer)

Pastikan komputer dan *smartphone* terhubung pada jaringan yang sama, penulis menggunakan jaringan wifi yang sama. Setelah terhubung dengan server *subscribe* pada topik tertentu, penulis menggunakan topik “test”. setelah itu *publish* pada topik tersebut sebuah pesan dalam hal ini penulis mem-*publish* pesan “ping” seperti terlihat pada gambar dibawah.



Gambar 22 proses publish pesan “ping”

Lihat pada tab subscribe akan ada pesan yang diterima seperti gambar dibawah ini.



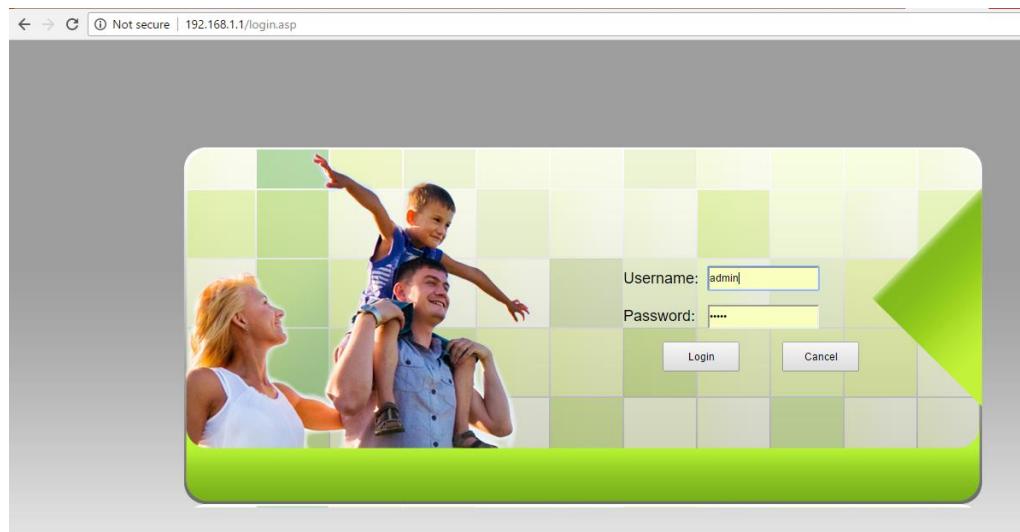
Gambar 23 tampilan subscribe

Ketika pesan diterima hal ini menandakan server telah berjalan pada *local network*.

2.2.3.2 Port Fowarding IP komputer server

Dalam langkah ini penulis melakukan *port fowarding* 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 fowarding* 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 24 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

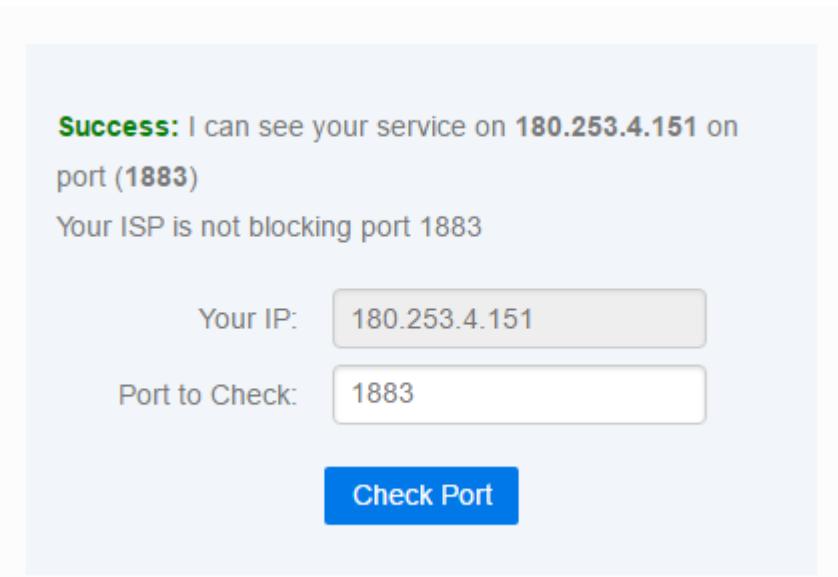
Lalu simpan pengaturan dan jalankan *profile* tersebut

WAN	WAN0
Discription	mqtt
Public Port	1883 - 1883
IP	192.168.1.7
Private Port	1883 - 1883
Protocol	ALL
Enable	Enable

Apply Cancel

Gambar 25 pengaturan *port fowarding*

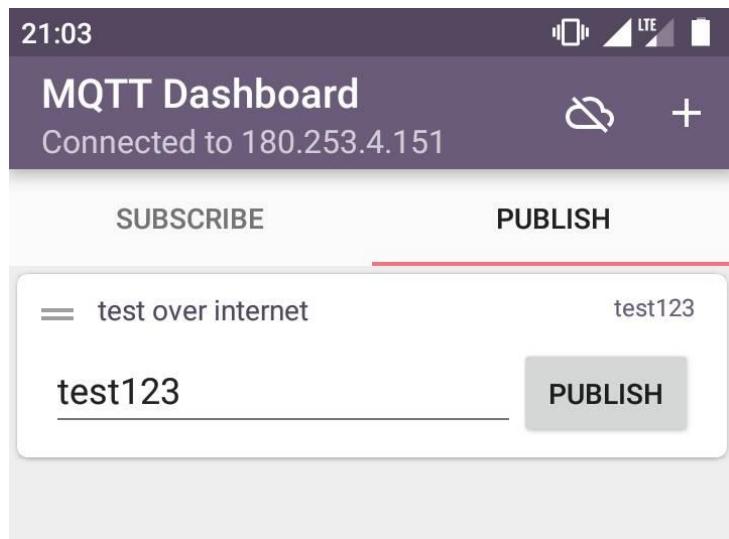
- Cek apakah port usdah dibuka melaui website <http://www.canyouseeme.org/> atau menggunakan website port checker lainnya



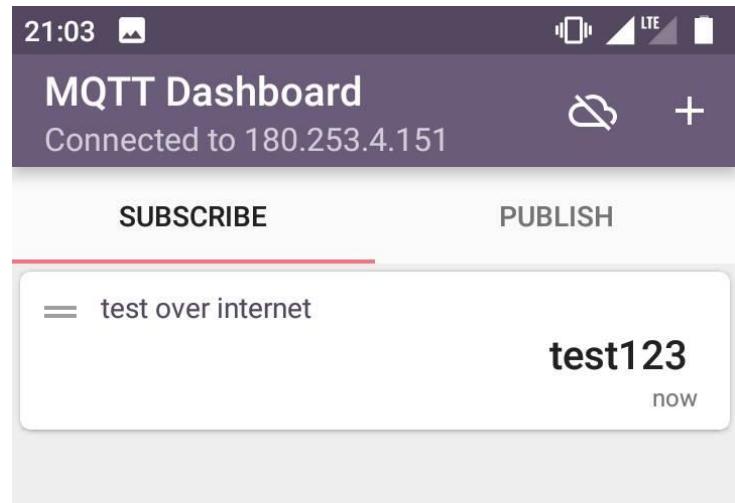
Gambar 26 hasil *port checker*

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

Untuk memastika port terbuka, uji MQTT melalui aplikasi android, kali ini gunakan IP internet dari komputer dan tidak harus terhubung pada jaringan yang sama. Hasil pegujian menggunakan aplikasi android dapat dilihat pada gambar di bawah



Gambar 27 proses *publish* pesan “test123”

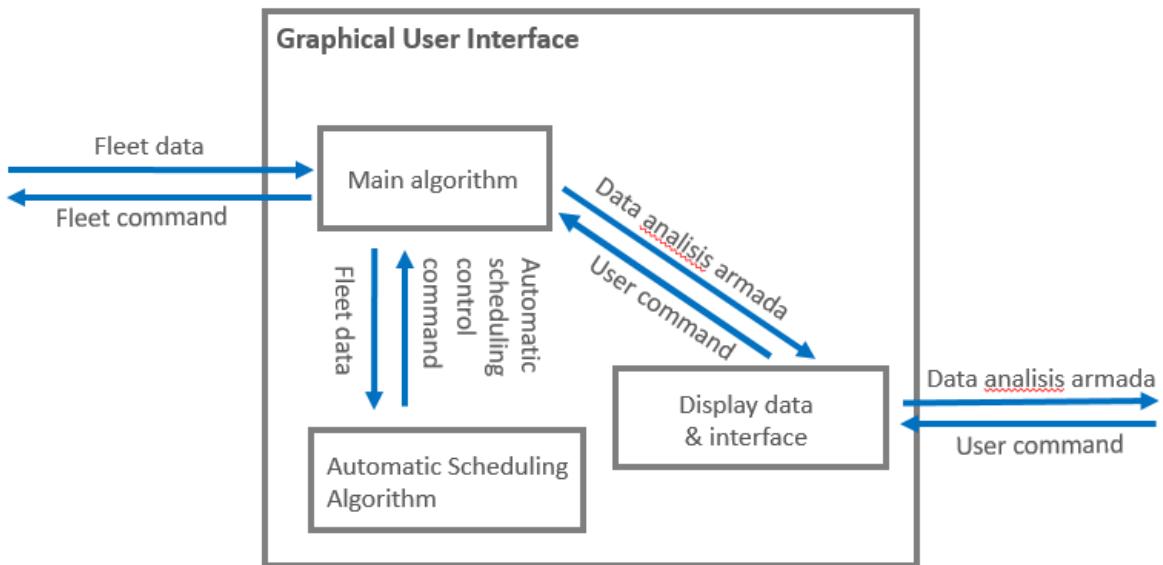


Gambar 28 tampilan subscribe

Hasil pengujian menunjukan server dapat diakses melalui internet.

2.3 IMPLEMENTASI SUB-SISTEM GRAPHICAL USER INTERFACE (GUI)

2.3.1 Struktur Implementasi



Gambar 29 Data flow diagram level 2 GUI

2.3.2 Environment

2.3.2.1 Microsoft Visual Studio



Gambar 30 Logo Visual Studio

Microsoft Visual Studio merupakan sebuah software yang dapat digunakan untuk membuat sebuah aplikasi. Software ini sendiri terdiri dari compiler, SDK, IDE, serta MSDN Library. Compiler yang tersedia di visual studio diantaranya adalah Visual C++, Visual C#, Visual Basic, Visual Basic .NET, Visual InterDev, Visual J++, Visual J#, Visual FoxPro, dan Visual SourceSafe. Untuk membuat system *monitoring* ini digunakan Bahasa C# dengan pemrograman berbasis OOP (Object Oriented Programming).

Spesifikasi minimum hardware yang dapat menggunakan atau menginstalasi Visual Studio (pada project ini digunakan VS Community 2015) beserta SDK adalah sebagai berikut:

1. Processor 1.6 GHz keatas
2. RAM 1 GB
3. Available hard disk space sebesar 4 GB
4. 5400 rpm hard disk drive
5. Rata-rata membutuhkan resolusi layar minimal 1024 x 768 atau lebih.

2.3.2.2 Python



Gambar 31 Logo Python

Python adalah sebuah bahasa pemrograman tingkat tinggi yang banyak digunakan untuk programming jenis apapun. Python didesain untuk memudahkan pembacaan kode (terkenal dalam penggunaan *whitespace* sebagai pengganti kurung atau *bracket*). Hal ini membuat python disukai banyak programmer karena dapat menghasilkan program yang sama dengan baris code yang lebih sedikit, dibandingkan menggunakan C++ atau Java.

Python memiliki sistem dinamik dan manajemen memori otomatis sehingga penggunaannya ringan. Selain itu Python juga menyupport beberapa jenis paradigma programming, diantaranya object-oriented, imperative, functional programming, dan procedural programming. Python tersedia dalam banyak sistem operasi, diantaranya adalah:

- | | |
|------------------------|-----------|
| - Linux/Unix | - OS/2 |
| - Windows | - Amiga |
| - Mac OS X | - Palm |
| - Java Virtual Machine | - Symbian |

Python mempunyai gap antara versi 2 dengan 3, beberapa diantaranya berupa perbedaan syntax. Pada project ini digunakan python versi 3 karena python 3.x merupakan *bench* yang dipersiapkan untuk *future programming*.

2.3.3 Prosedur Implementasi

Dalam melaksanakan implementasi GUI, dilakukan beberapa proses sebagai berikut:

1. Menentukan konten GUI
2. Mengambil dan mengirim data dengan MQTT
3. Mendesain dan meingimplementasikan peta geografis pada GUI
4. Mendesain dan meingimplementasikan peta diagramatis pada GUI
5. Mengintegrasikan algoritma penjadwalan pada GUI
6. Mengintegrasikan seluruh bagian dari GUI
7. Melakukan pengujian, *debugging*, *troubleshooting* dan revisi terhadap prototipe.
8. Finalisasi layout dan tampilan akhir

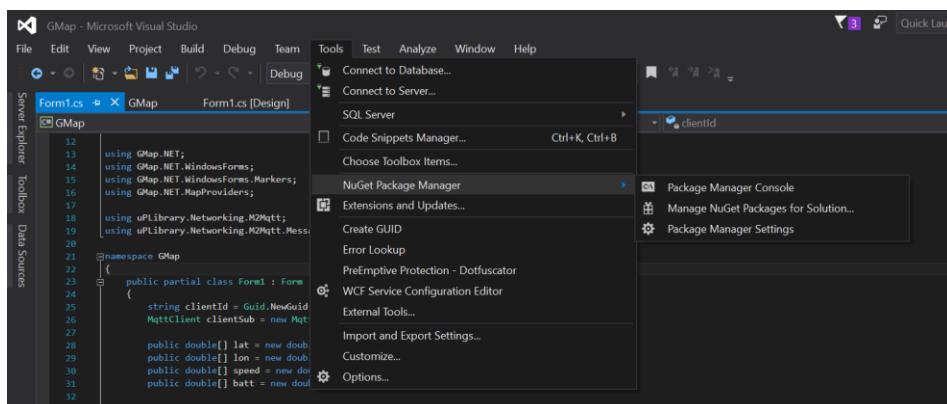
Perancangan GUI ini dilakukan dengan menggunakan Visual Studio, adapun platform yang digunakan adalah windows forms, dengan bahasa pemrograman C# dan berbasis OOP. Berikut penjelasan proses penggerjaan, prosedur implementasi, dan permasalahan yang dialami selama proses perancangan GUI.

2.3.3.1 Implementasi MQTT pada GUI untuk mengirim dan menerima data

2.3.3.1.1 Connect ke MQTT

Karena modul hardware mengirim data dengan menggunakan protokol MQTT, maka WinForms juga disiapkan untuk menerima data mengugnakan protokol MQTT. Adapun step-step untuk connect ke MQTT adalah sebagai berikut :

1. Pada WinForms ini, digunakan M2Mqtt sebagai library MQTT. M2Mqtt merupakan library klien MQTT yang disediakan untuk .NET Framework untuk komunikasi M2M (*machine-to-machine*, antar mesin). Ada banyak cara untuk menginstall library M2Mqtt pada WinForms, salah satunya dengan menggunakan NuGet Package Manager.
2. Instalasi dapat dilakukan dengan menggunakan Package Manager Console (Tools → NuGet Package Manager → Package Manager Console).



Gambar 32 Package Manager Console

Pada Package Manager Console, tulis command **PM > Install-Package M2Mqtt**, apabila PC terkoneksi dengan internet, Visual Studio secara otomatis akan men-download Package tersebut. Kemudian untuk mempermudah penulisan syntax, tambahkan prefix pada deklarasi using :

```
using uPLibrary.Networking.M2Mqtt;
using uPLibrary.Networking.M2Mqtt.Messages;
```

3. Membuka gate untuk koneksi MQTT, variabel-variabel berikut dituliskan pada bagian deklarasi variabel :

```
string clientId = Guid.NewGuid().ToString(); //Client ID
MqttClient clientSub = new MqttClient("broker.hivemq.com"); // Buat subscribe
```

Deklarasikan variabel clientID yang digunakan untuk menampung ID Client, clientID ini harus unik. Lalu deklarasikan MqttClient beserta alamat broker atau host yang digunakan. Pada ujicoba ini masih digunakan broker.hivemq.com sebagai penyedia akses.

4. Perintah connect diletakkan pada Form1(), setelah inisialisasi komponen WinForms:

```
clientSub.Connect(clientId);
```

Sehingga setiap program baru dijalankan, program langsung berusaha terhubung ke broker MQTT.

2.3.3.1.2 Subscribe

Setelah berhasil terhubung dengan broker, hal yang wajib dilakukan dari penggunaan protokol MQTT adalah melakukan subscribe. Pada MQTT dapat dibuat beberapa topik unik, dan bisa disubscribe oleh banyak *device*. Sesuai namanya, ketika kita meminta izin *subscribe* untuk suatu topik, maka setiap informasi yang di *post* pada topik tersebut akan langsung kita terima. Syntax untuk subscribe adalah sebagai berikut :

```
ushort msgId = clientSub.Subscribe(new string[] { "fleet1" , "fleet2" ,
"fleet3" , "fleet4" , "fleet5" , "fleet6" , "fleet7" , "fleet8" , "fleet9" , "fleet10" },
new byte[] { MqttMsgBase.QOS_LEVEL_AT_LEAST_ONCE, MqttMsgBase.QOS_LEVEL_AT_LEAST_ONCE ,
MqttMsgBase.QOS_LEVEL_AT_LEAST_ONCE, MqttMsgBase.QOS_LEVEL_AT_LEAST_ONCE ,
MqttMsgBase.QOS_LEVEL_AT_LEAST_ONCE, MqttMsgBase.QOS_LEVEL_AT_LEAST_ONCE ,
MqttMsgBase.QOS_LEVEL_AT_LEAST_ONCE, MqttMsgBase.QOS_LEVEL_AT_LEAST_ONCE ,
MqttMsgBase.QOS_LEVEL_AT_LEAST_ONCE, MqttMsgBase.QOS_LEVEL_AT_LEAST_ONCE ,});
```

QoS (Quality of service) merupakan kualitas layanan dari masing-masing pesan yang di *publish*. QoS memiliki tiga level berbeda, yaitu :

- QoS0, At most once : Pada QoS ini, pesan tidak disimpan dan dapat hilang jika client terputus atau server gagal. Tetapi QoS0 merupakan mode transfer yang paling ringan.
- QoS1, At least once : Pada QoS ini, pesan akan dikirim beberapa kali jika terjadi kegagalan pada penerimaan. Pesan akan disimpan oleh pengirim hingga pengirim menerima konfirmasi bahwa pesan sudah diterima.
- QoS2, Exactly once : Cara pengiriman pesan kurang lebih sama dengan QoS1, yang membedakan adalah pada QoS2 pesan akan dipastikan tidak terduplikat.

Untuk diketahui, semakin tinggi level QoS maka pengiriman data akan semakin aman, tetapi akan semakin lambat. Karena kebutuhan kita hanya memastikan pesan terkirim, maka digunakan QoS1. Karena pada simulasi program ini digunakan 10 fleet, maka topik yang disubscribe adalah fleet1 hingga fleet10, dan semuanya menggunakan QoS1.

Sementara untuk menerima pesan yang dipublish, digunakan event yang bernama *MqttMsgPublishReceived*.

```
clientSub.MqttMsgPublishReceived += client_MqttMsgPublishReceived;
```

Fungsi ini dideklarasikan setelah *clientSub.Connect(clientId)*, sehingga setelah program terkoneksi lewat mqtt, program akan selalu siap untuk menerima message yang dipublish.

```
void client_MqttMsgPublishReceived(object sender, MqttMsgEventArgs e)
{
    Application.DoEvents();

    if (e.Topic == "fleet1")
    {
        string datafleet1 = Encoding.UTF8.GetString(e.Message);
        string[] data = datafleet1.Split(',');
        //label14.Text = "Status : Data fleet1 masuk";
        lat[0] = double.Parse(data[0], CultureInfo.InvariantCulture);
        lon[0] = double.Parse(data[1], CultureInfo.InvariantCulture);
        speed[0] = double.Parse(data[2], CultureInfo.InvariantCulture);
        batt[0] = double.Parse(data[3], CultureInfo.InvariantCulture);
    }
}
```

Pada fungsi *client_MqttMsgPublishReceived*, terlebih dahulu data yang dikirim akan masuk fungsi *if* untuk mengecek topik yang dikirimkan. Apabila fleet1, maka data akan dimasukkan ke array *fleet1*. Data yang diterima akan berupa sebuah string. Pada ujicoba ini, limiter yang digunakan berbentuk koma, sehingga setiap data yang dipisahkan oleh koma kemudian diinput ke masing-masing variabel *fleet1*.

2.3.3.1.3 Publish

Selain melakukan subscribe, pada program ini juga dibutuhkan fungsi publish untuk mengirimkan instruksi ke hardware. Digunakan syntax sebagai berikut:

```
void commandfleet1(object sender, EventArgs e)
{
    clientSub.Publish("fleet1/comm", Encoding.UTF8.GetBytes(speedcomm[0].ToString()), MqttMsgBase.QOS_LEVEL_EXACTLY_ONCE, false);
    timer1.Enabled = true;
}
```

Data yang dikirimkan adalah data kecepatan fleet1 (*speedcomm[0]*) lewat topik *fleet1/comm*. Pesan ini menggunakan QoS1.

2.3.3.2 Desain dan implementasi peta geografis pada GUI

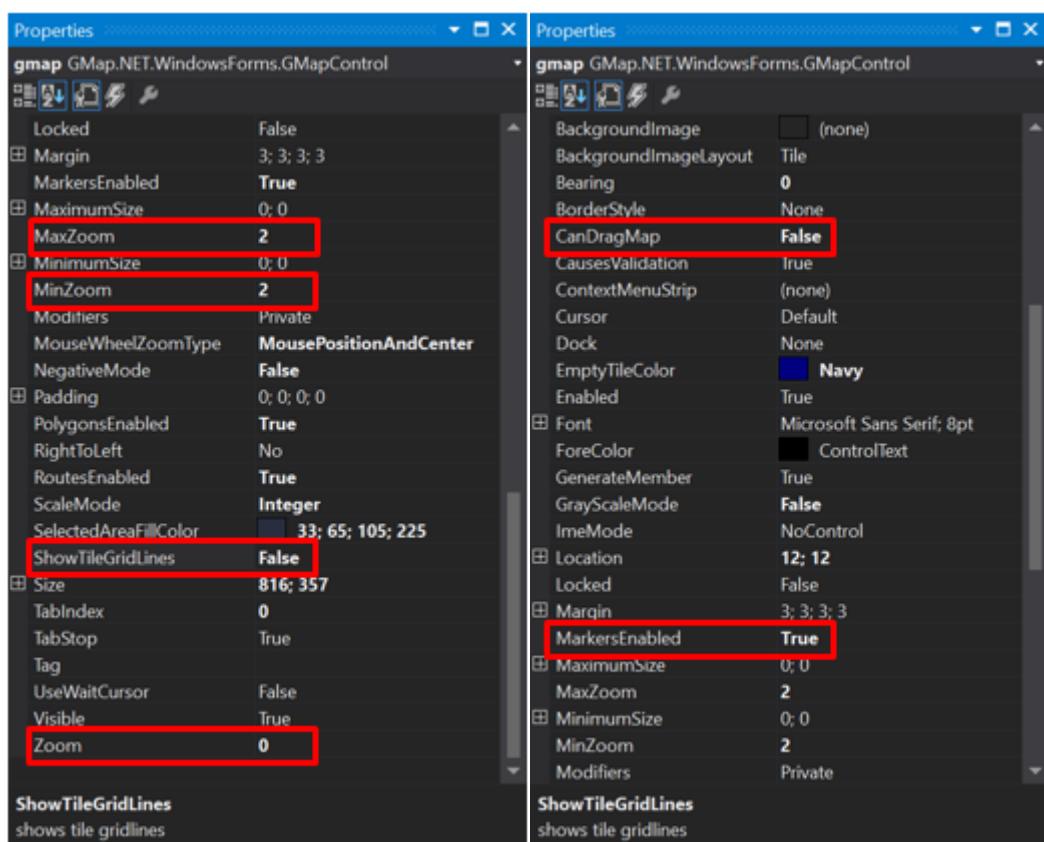
2.3.3.2.1 Menampilkan peta *real* pada WinForms dengan menggunakan GMap.NET

GMap.NET merupakan sebuah framework yang menyediakan layanan open source untuk mengakses server Google Map. GMap.NET dapat ditambahkan sebagai library ke Windows Forms yang dibangun di Visual Studio. Pada tampilan ini digunakan GMap.NET versi 4.0.

Langkah pertama yang harus dilakukan adalah mendownload dan meletakkan folder library GMap.NET yang berisi file *GMap.NET.Core.dll* dan *GMap.NET.WindowsForms.dll* di dalam folder project Visual Studio yang sedang dibuat. Untuk menambahkan DLL files agar bisa digunakan, klik kanan References di Solution Explorer, kemudian Add Reference. Browse file *GMap.NET.Core.dll* dan *GMap.NET.WindowsForms.dll* dan tambahkan sebagai Reference.

Setelah menambahkan file .dll, klik kanan Toolbox di bawah desain WinForms, dan pilih *Choose Items*. Kemudian browse file .DLL di tab *.NET Framework Components* dan centang GmapControl untuk menambahkannya kedalam Toolbox. Jika berhasil, panel GMapControl akan muncul di Tab General dan dapat di drag ke bawah desain.

Panel GMapControl ini akan menampung peta dari server Google Map dan menjadi layer untuk menampilkannya. Adapun konfigurasi yang harus dilakukan pada GMapControl ini adalah:



Gambar 33 GMap Control

1. Mematikan fitur drag map. GMapControl memiliki fitur untuk drag peta, karena pada project ini dibutuhkan sebuah peta dengan tampilan yang tetap, maka fitur CanDragMap diganti menjadi False.
2. Menyalakan fitur markers. Jika fitur ini dinyalakan, GMap dapat menampilkan seluruh marker yang dibuat pada panel tersebut. Karena project ini harus dapat menampilkan marker, maka di MarkersEnabled diganti menjadi true.
3. Mematikan GridLines. Umumnya panel display memiliki gridlines agar dapat memudahkan pembaca melakukan pengukuran atau pemeriksaan posisi. Karena pada project ini tidak dibutuhkan, ShowTileGridLines dapat diset false.
4. Menentukan Zoom. Skala zoom level Google Maps berada diantara 0 (skala paling jauh) hingga 18 (skala paling detail). Zoom merupakan zoom level default ketika

panel baru dinyalakan. Sementara MinZoom dan MaxZoom digunakan apabila pengguna butuh untuk melihat map lebih detil. Karena project ini membutuhkan skala view yang statis, maka Zoom, MinZoom, dan MaxZoom diset pada angka yang sama.

Setelah konfigurasi panel dilakukan, tambahkan library-library berikut untuk mempermudah penulisan syntax GMap.NET di form.cs :

```
using GMap.NET;
using GMap.NET.WindowsForms;
using GMap.NET.WindowsForms.Markers;
using GMap.NET.MapProviders;
```

Kemudian dilakukan konfigurasi konten panel sebagai berikut:

1. Menentukan penyedia tampilan peta. Pada project ini digunakan BingMapProvider sebagai Map Provider karena memiliki tampilan yang user-friendly dan ringan. Setiap provider memiliki API yang berbeda, tetapi penyettingan API tidak perlu dilakukan secara manual karena sudah dilakukan oleh library GMap.NET (hanya tinggal memilih jenis Map Provider yang diinginkan). Sehingga syntaxnya adalah sebagai berikut:

```
gmap.MapProvider = BingMapProvider.Instance;
```

2. Menentukan mode akses. GMap dapat mengambil data dari server, dari server dan local cache, atau local chace saja. Pada project ini koneksi dilakukan lewat server sehingga setting mode akses menjadi GMap.NET.AccessMode.ServerOnly, dengan syntax sebagai berikut :

```
GMaps.Instance.Mode = GMap.NET.AccessMode.ServerOnly;
```

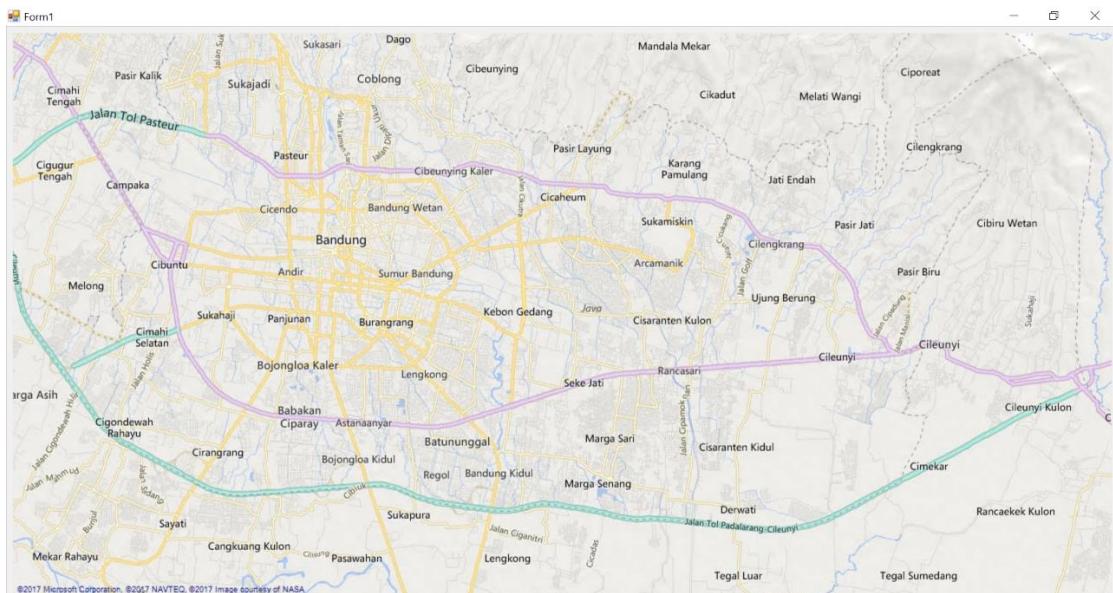
3. Menentukan default posisi peta. Inisiasi posisi awal tampilan peta dapat dilakukan dengan menuliskan keyword lokasi maupun koordinat pada peta. Karena pada project ini membutuhkan peta wilayah bandung, maka lokasi default akan dituliskan sebagai berikut:

```
gmap.SetPositionByKeywords("Bandung, Indonesia");
```

Setelah menentukan posisi default, dilakukan konfigurasi untuk menghilangkan tanda center bawaan dari GMapControl dengan syntax sebagai berikut:

```
gmap.ShowCenter = false;
```

Setelah konfigurasi panel dan konten panel, hasil tampilan awal adalah sebagai berikut:



Gambar 34 Tampilan awal peta geografis

2.3.3.2.2 Menampilkan marker di posisi tertentu pada peta geografis

Panel GMapControl memiliki fungsi overlay, yaitu proses penambahan data baru ke data yang telah ada. Pada panel GMapControl ini, layer utamanya merupakan tampilan peta, sehingga penambahan marker dilakukan dengan menambahkan layer baru diatas tampilan peta agar tidak merusak layer yang telah ada. Dengan menggunakan fungsi overlay ini, jumlah marker yang dapat ditampilkan dapat mencapai tak hingga.

Syntax untuk menambahkan marker pada GMapControl adalah sebagai berikut:

```
GMapOverlay markers = new GMapOverlay("markers");
GMapMarker marker = new GMarkerGoogle(
    new PointLatLng(-6.946575, 107.637882),
    GMarkerGoogleType.blue_pushpin);
markers.Markers.Add(marker);
gmap.Overlays.Add(markers);
```

- *markers* merupakan variabel local marker yang ditambahkan.
 - New PointLatLang : penempatan marker dengan menggunakan koordinat/keyword
 - GMarkerGoogleType : jenis tampilan marker yang digunakan

Pada project ini marker pada peta geografis akan ditentukan dengan menggunakan koordinat.

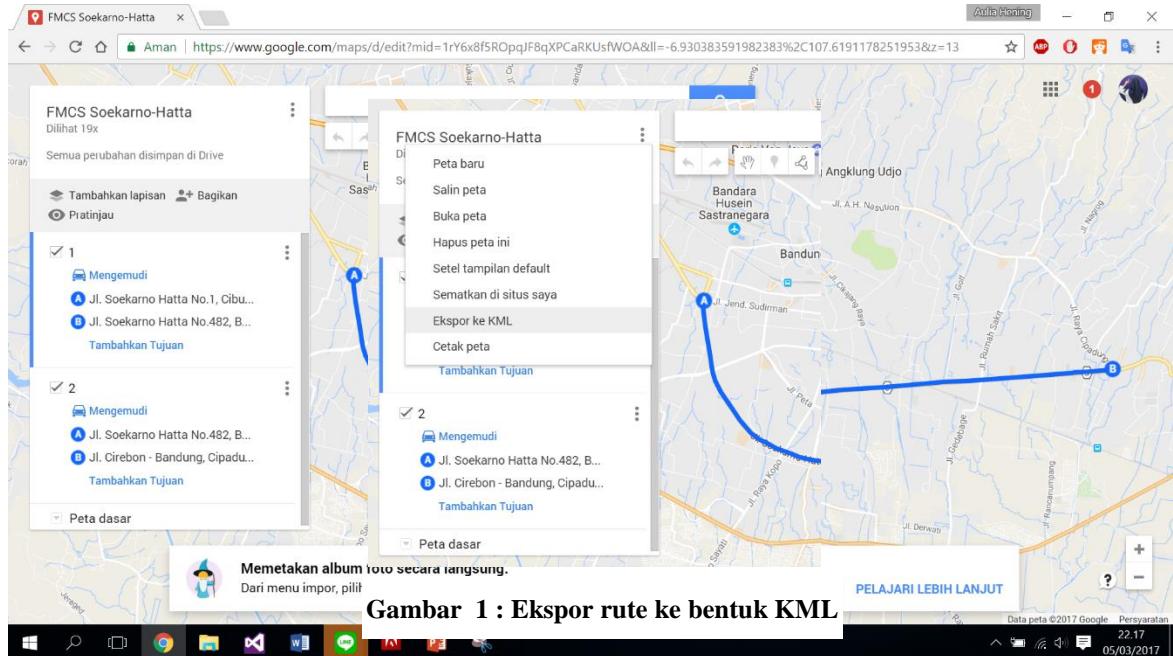
2.3.3.2.3 Pembuatan array segmentasi koordinat geografis

Pada B200, disebutkan spesifikasi yang diinginkan data posisi memiliki error maksimal 6 meter. Hal ini menjadi dasar dibuatnya array segmentasi koordinat geografis. Agar error dari GPS dapat diminimalisir, maka dibuat array koordinat disepanjang jalur trayek Elang-Cibiru. Hal ini dibuat untuk menghandle situasi apabila data koordinat yang dikirim GPS meleset, ia akan tetap ditampilkan pada posisi yang benar.

Agar error yang diperoleh maksimal 6 meter, dilakukan pencacahan koordinat setiap 3 meter sepanjang rute Soekarno-Hatta (dengan pertimbangan 3 meter ke belakang dan 3 meter ke depan).

Adapun untuk melakukan pencacahan, langkah pertama yang dilakukan adalah mencari koordinat garis lurus pada Google Map :

1. Masuk ke <https://www.google.com/maps/d> dan klik “Buat Peta Baru”. Tandai Elang sebagai titik mulai dan Cibiru sebagai titik akhir, rute Soekarno-Hatta telah jadi.



Gambar 1 : Ekspor rute ke bentuk KML

Gambar 35 Pembuatan rute

2. Klik titik tiga di pojok kanan atas sebelah nama peta, dan pilih “Ekspor ke KML”. KML merupakan format file yang digunakan untuk menampilkan data geografis pada browser.
3. Setelah mendownload file, buka file di notepad dan akan didapatkan array koordinat yang menunjukkan seluruh titik perubahan garis lurus yang terdapat pada sepanjang rute.

```

<?xml version="1.0" encoding="UTF-8"?>
<kml>
<Document>
<Folder id="1">
<name>Jl. Soekarno Hatta</name>
<Placemark id="1">
<name>Peta jalur arah dari Jl. Soekarno Hatta No.1, Cibuntu, Bandung Kulon, Kota Bandung, Jawa Barat 40212, Indonesia ke Jl. Soekarno Hatta No.482, Batununggal, Bandung Kidul, Jawa Barat 40132</name>
<LineString id="line1267FF-5-nodes0c">
<coordinates>
107.57443,-6.91742,0
107.57443,-6.91761,0
107.57441,-6.91811,0
107.5744,-6.91904,0
107.5744,-6.91914,0
107.57441,-6.91942,0
107.57441,-6.91952,0
107.57441,-6.92059,0
107.57458,-6.92099,0
107.57476,-6.92166,0
107.57476,-6.92199,0
107.57476,-6.92201,0
107.57481,-6.92235,0
107.57491,-6.92293,0
107.57492,-6.92304,0
107.57494,-6.92312,0
107.57504,-6.92368,0
107.57509,-6.92378,0
107.57513,-6.92445,0
107.57516,-6.92468,0
107.57526,-6.92526,0
107.57528,-6.9254,0
107.57532,-6.92574,0
107.57532,-6.92605,0
107.57552,-6.92633,0
107.57578,-6.92871,0
107.57584,-6.92904,0
110.0 107.57596,-6.92988,0
111.0 107.57605,-6.93024,0
112.0 107.57605,-6.93041,0
113.0 107.57607,-6.93049,0
114.0 107.57611,-6.93067,0
115.0 107.57616,-6.93085,0
116.0 107.5762,-6.9311,0
</coordinates>
</LineString>
</Placemark>
</Folder>
</Document>
</kml>

```

Gambar 36 File rute dalam bentuk teks

4. Data koordinat garis lurus ini kemudian disatukan dalam sebuah file .txt yang akan diproses oleh sebuah program pembagi interval dalam bahasa python. Pada program ini, data akan diolah untuk dibagi setiap 3 meter sekali dan hasilnya akan di *write* pada output file. Library yang digunakan pada program ini diantaranya adalah:

```

import math
import decimal
import struct

```

5. Dilakukan pembacaan pada file input *jlurus.txt* sekaligus menginput data ke array. Karena dalam hal ini data yang dibaca adalah koordinat, data dibatasi hingga 11 karakter (instruksi |S11).

```

with open("jlurus.txt","r") as f:
    for line in f:
        if line:
            array.append(line)

import numpy as np
x = np.array(array, dtype='|S11')
farray = x.astype(np.float)

```

6. Tujuan kita adalah mencari titik-titik koordinat disepanjang interval antara dua titik koordinat yang diketahui. Program ini akan menerima input berupa dua titik koordinat dan jarak interval yang diinginkan.

2.3.3.2.3.1 Terlebih dahulu dicari panjang antara dua titik. Karena disini titiknya berbentuk kordinat, perhitungannya tidak cukup hanya dengan menggunakan rumus pencarian jarak diantara dua titik, tetapi harus memperhitungkan kelengkungan bumi, sudut dan beberapa parameter lain.

Metode pencarian jarak terdekat antara dua titik koordinat salah satunya adalah menggunakan **Formula Haversine**, yang mempunyai rumus sebagai berikut :

1. $a = \left(\sin\frac{\Delta lat}{2}\right)^2 + \cos(lat1rads) \times \cos(lat2rads) \times \left(\sin\frac{\Delta long}{2}\right)^2$
2. $c = 2 \times \text{atan2}(\sqrt{a}, \sqrt{1-a})$
3. $d = R \times c$

Dengan Δlat merupakan selisih antara latitude 1 dan latitude 2, $\Delta long$ selisih dari longitude 1 dan longitude 2, dan R jari-jari bumi. Sehingga implementasinya dalam python menjadi sebagai berikut :

```
def getPathLength(lat1,lng1,lat2,lng2):
    '''calculates the distance between two lat, long coordinate pairs'''
    R = 6371000 # radius of earth in m
    lat1rads = math.radians(lat1)
    lat2rads = math.radians(lat2)
    deltaLat = math.radians((lat2-lat1))
    deltaLng = math.radians((lng2-lng1))
    a = math.sin(deltaLat/2) * math.sin(deltaLat/2) + math.cos(lat1rads)
    * math.cos(lat2rads) * math.sin(deltaLng/2) * math.sin(deltaLng/2)
    c = 2 * math.atan2(math.sqrt(a), math.sqrt(1-a))
    d = R * c
    return d
```

7. Yang selanjutnya harus dilakukan adalah mencari azimuth dengan menggunakan bearing. Azimuth merupakan sudut yang terbentuk antara titik utara peta dengan titik sasaran. Ilustrasi :

- o 45 derajat azimuth = 45 derajat bearing ke timur dari utara (N 45 E)
- o 135 derajat azimuth = 45 derajat bearing ke timur dari selatan (S 45 E)

Karena itu azimuth dapat juga dicari dengan menggunakan bearing dengan rumus berikut :

$$\text{bearing} = \left(\text{atan2} \left(\Delta long, \log \left(\frac{\tan \left(\frac{\Delta lat}{2} + \frac{\pi}{4} \right)}{\tan \left(\frac{\Delta lat}{2} - \frac{\pi}{4} \right)} \right) \right) + 360 \right) \text{ mod } 360$$

Yang jika diimplementasikan dalam python adalah sebagai berikut :

```
def calculateBearing(lat1,lng1,lat2,lng2):
    '''calculates the azimuth in degrees from start point to end point'''
    startLat = math.radians(lat1)
    startLong = math.radians(lng1)
    endLat = math.radians(lat2)
    endLong = math.radians(lng2)
    dLong = endLong - startLong
    dPhi = math.log(math.tan(endLat/2.0+math.pi/4.0)/math.tan(startLat/2.0+math.pi/4.0))
    if abs(dLong) > math.pi:
        if dLong > 0.0:
            dLong = -(2.0 * math.pi - dLong)
        else:
            dLong = (2.0 * math.pi + dLong)
    bearing = (math.degrees(math.atan2(dLong, dPhi)) + 360.0) % 360.0;
    return bearing
```

8. Setelah didapatkan azimuth dan jarak, dicari latitude dan longitude konversi hasil perhitungan dengan menggunakan fungsi getDestinationLatLong() yang terdapat pada line 35 – line 48 di lampiran coordinate_interval.py.

9. Pada algoritma utama, program akan menerima banyak titik koordinat pada file input jlurus.txt dan mencari koordinat-koordinat sepanjang interval diantara dua titik.

```

if __name__ == "__main__":
    #INI YANG DIGANTI
    #point interval in meters
    interval = 3.0
    #direction of line in degrees
    #start point
    for a in range(0,621):
        if (a%2 == 0):
            lng1 = farray[a]
            lat1 = farray[a+1]
            lng2 = farray[a+2]
            lat2 = farray[a+3]

            azimuth = calculateBearing(lat1,lng1,lat2,lng2)
            coords = main(interval,azimuth,lat1,lng1,lat2,lng2)
            print(coords, file=dest)

```

Karena yang diinginkan adalah koordinat disetiap 3 meter, maka input interval diisi dengan 3.0. Sementara 621 adalah jumlah koordinat yang akan diinput pada file jlurus.txt

10. Hasil akhirnya didapatkan koordinat-koordinat yang diinginkan setiap 3 meter sepanjang trayek Soekarno-Hatta pada file hasilcoord.txt

2.3.3.3 Desain dan implementasi peta diagramatis pada GUI

2.3.3.3.1 Pembuatan peta diagramatis

Peta diagramatis (atau banyak disebut *schematic maps*) merupakan peta yang umum digunakan untuk menunjukkan pergerakan kendaraan pada pengguna (baik penumpang maupun operator control station). Pembuatan peta diagramatis bertujuan untuk memudahkan pengguna memantau posisi kendaraan dengan tampilan yang lebih mudah dimengerti, dibandingkan dengan menampilkan pada peta yang sesungguhnya (peta geografis).

Peta diagramatis ini dibuat dengan cara menghilangkan informasi yang tidak dibutuhkan dan cenderung menggunakan simbol-simbol yang dapat memberi informasi tambahan seperti halte kecil, halte besar, interchange, dan nama halte.

Karena bertujuan untuk memudahkan pengguna memahami trayek, peta diagramatis digambarkan dengan garis lurus. Metode pembuatannya sendiri terbagi dua:

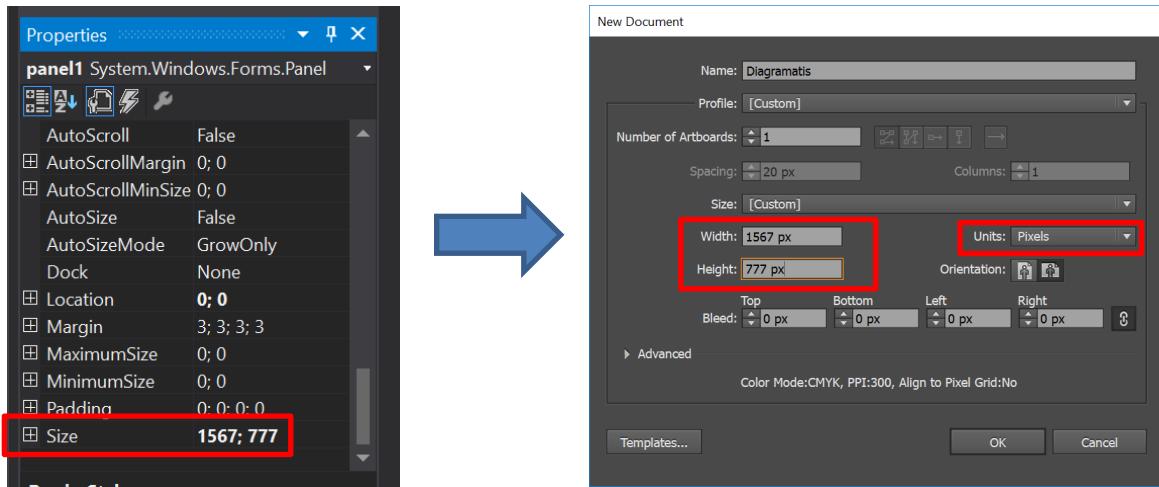
1. Isometric grid (menggunakan kemiringan 30 derajat)
2. Orthogonal method (menggunakan kemiringan 45 derajat)

Pada project ini digunakan tampilan orthogonal karena lebih user-friendly.

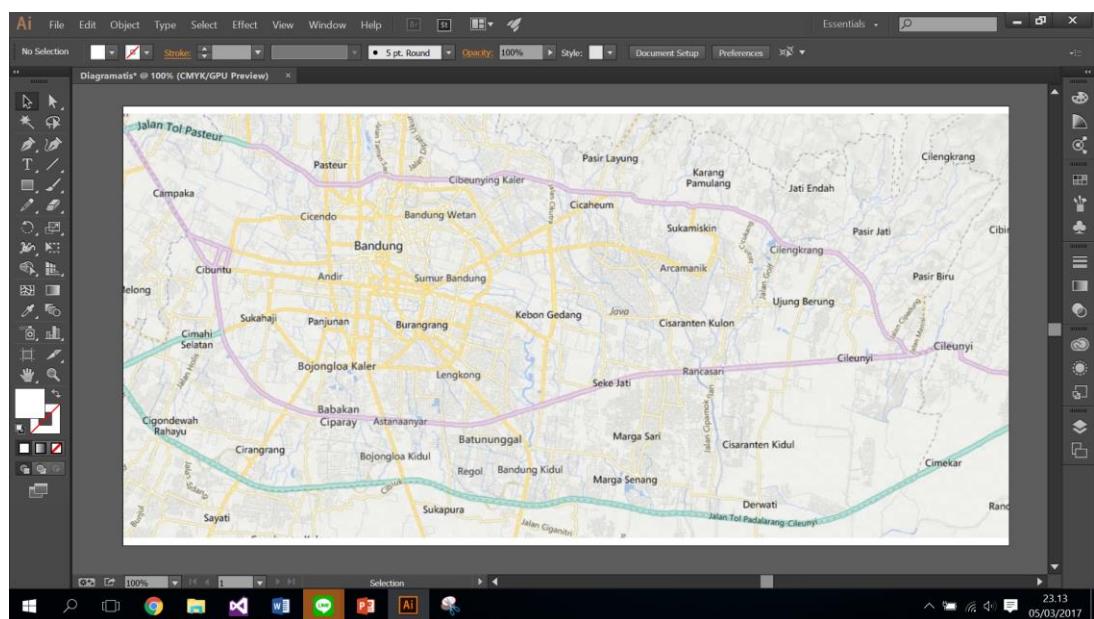
Pemodelan peta diagramatis dengan garis lurus ini menyebabkan peta menjadi tidak berskala (karena itu peta diagramatis banyak disebut sebagai peta tanpa skala), karena penarikan garis dilakukan hanya berdasarkan pendekatan dan hanya mempertimbangkan tampilan.

Peta diagramatis didesain secara manual dengan menggunakan Adobe Illustrator dan kemudian hasilnya di *import* ke visual studio. Adapun proses pembuatannya adalah sebagai berikut:

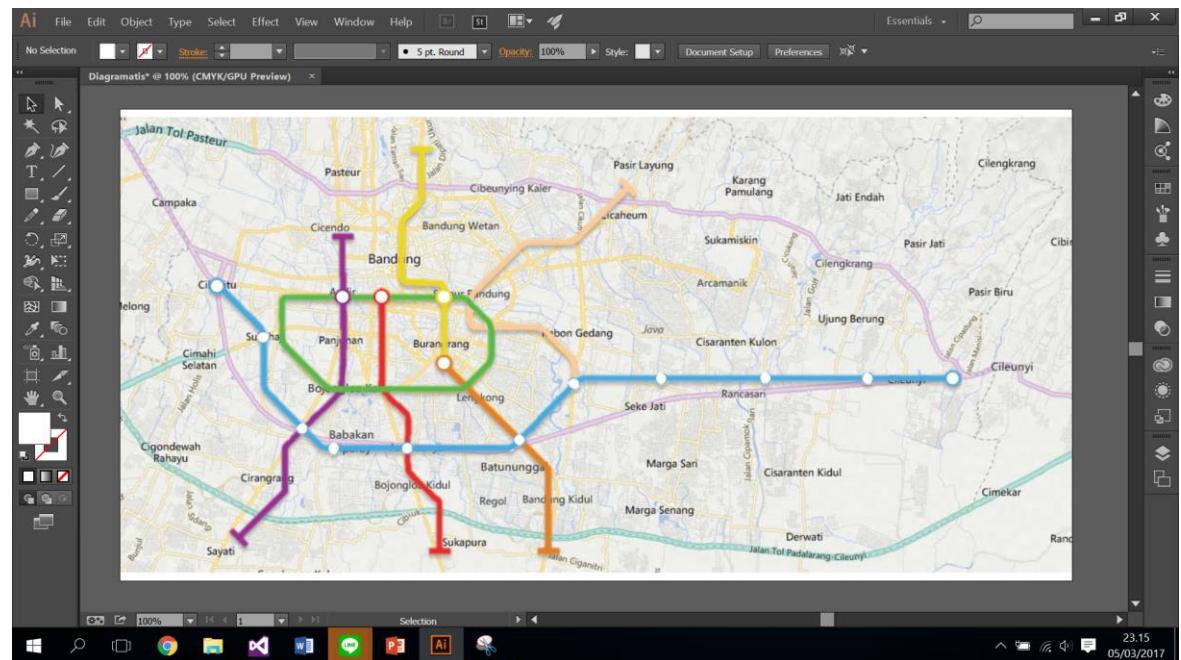
1. Membuat halaman kerja pada Adobe Illustrator. Karena gambar ini akan langsung di *import* ke panel Visual Studio, besar canvas yang digunakan pada Adobe Illustrator harus sama dengan ukuran panel tempat gambar akan diletakkan.



2. Tempatkan gambar peta geografis pada background layer sehingga dapat dijadikan patokan dalam penarikan jalur.



3. Tarik garis sesuai rute yang telah dirancang dan se bisa mungkin mendekati skala aslinya. Garis yang dibuat hanya boleh memiliki kemiringan dengan kelipatan 45° , dan diberi warna yang berbeda untuk masing-masing trayeknya.



4. Beri informasi tambahan berupa simbol halte, nama halte, dan nama trayek.



5. Setelah peta diagramatis jadi, file gambar kemudian di ekspor ke format .PNG dan dijadikan sebagai background image panel.

2.3.3.3.2 Pembuatan array segmentasi koordinat diagramatis

Sama seperti peta geografis, pada peta diagramatis juga dibutuhkan array segmentasi untuk jalur pergerakan marker. Karena gambar dibuat secara manual, pembuatan array juga akan dibuat secara manual terhadap point di panel diagramatis.

1. Tandai setiap belokan dengan picturebox, kemudian catat *Location* dari masing-masing picturebox.



Didapatkan 7 point sebagai berikut, yang kemudian dicari panjang antar pointnya, persentase panjang masing-masing garis dari keseluruhan panjang, sehingga kemudian dapat diketahui pembagian jumlah array untuk masing-masing garis.

Point	X	Y	D	%	Pixel segment (rounded)
1	152	252	0	0	0
2	229	329	108,8944443	7,525240111	467
3	230	423	94,005319	6,496314864	403
4	330	527	144,2775104	9,970415985	618
5	642	527	312,0016026	21,56112728	1336
6	760	409	166,8772004	11,53218614	715
7	1381	409	621	42,91471561	2660
Total	18,2	100	1447,056077	100	6199

Pixel segment merupakan persentase panjang masing-masing garis yang dikali dengan total array segmentasi geografis agar bisa didapat mapping yang sesuai.

2. Selanjutnya setelah didapat point-point garis lurus, dilakukan pencacahan dengan menggunakan rumus pencarian panjang antara dua titik sebagai berikut :

$$d = \sqrt{(\Delta x)^2 + (\Delta y)^2}$$

Dengan Δx adalah selisih antara x_1 dan x_2 , dan Δy selisih antara y_1 dan y_2 . Implementasi rumus pada C# adalah sebagai berikut :

```

diffX = p2.X - p1.X;
diffY = p2.Y - p1.Y;

double d = (double)Math.Sqrt(diffX*diffX + diffY*diffY);
string dstring = ("Panjang garis = " + d);

StreamWriter file = new StreamWriter("E:\\Hasil1.txt");

for (index = 0; index<(quantity+1); index++)
{
    pointx = ((index / quantity) * diffX)+p1.X;
    pointy = ((index / quantity) * diffY)+p1.Y;
    string s = ("{" + pointx + "*" + pointy + "},");
    file.WriteLine(s);

    if (index == (quantity))
    {
        file.WriteLine(dstring);
        MessageBox.Show("DONE!");
    }
}

file.Close();

```

Akan didapatkan array yang berjumlah 6199 point.

3. Kemudian agar bisa didapatkan posisi halte yang sesuai, dilakukan perhitungan berikut :

Halte	Kilometer	%	Range array	Array ke	Range pixel
Elang	0		0	0	0
Pasirkoja	1,5	8,241758242	511	511	122
Kopo	2,4	13,18681319	817	1328	195
Cibaduyut	0,65	3,571428571	221	1549	53
Moh. Toha	1,6	8,791208791	545	2094	130
Batununggal	1,8	9,89010989	613	2707	146
Buah Batu	0,85	4,67032967	289	2996	69
Kiaracondong	1	5,494505495	341	3337	81
Margahayu	2,9	15,93406593	988	4325	235
Gedebage	2,8	15,38461538	954	5279	227
Cibiru	2,7	14,83516484	920	6199	219
Total	18,2	100	6199	6199	1477

Kolom array ke merupakan posisi dari masing-masing halte pada array diagramatis.

2.3.3.3 Marker pada peta diagramatis

Jika pada sebelumnya di peta geografis digunakan fitur *markers* yang disediakan oleh GMap.NET, pada peta diagramatis ini karena pembuatannya manual, marker yang digunakanpun juga manual.

Pada peta diagramatis ini, marker diimplementasikan dengan menggunakan *pictureBox*. Picturebox ini dirancang untuk dapat mengikuti jalur sepanjang array point yang telah dibuat. Picturebox dapat digerakkan dengan mengubah koordinatnya dengan mengubah start point left dan topnya.

```
void displayfleet1dia(int i)
{
    Mfl1.Visible = true;
    Mfl1.Left = (int)(pointdia[i,0]);
    Mfl1.Top = (int)(pointdia[i,1]);
}
```

Dengan i adalah index array point diagramatis dari posisi fleet.

2.3.3.4 Implementasi Dekripsi

Dekripsi yang digunakan adalah AES yaitu *Advance Encryption System*. Enkripsi AES bersifat *symmetric-key* sehingga kedua pihak (pihak enkriptor dan dekriptor) membutuhkan *key* dan juga *initial vector* disingkat “IV” yang sama. Data yang diterima dari hasil proses enkripsi sepanjang 32 byte akan didekripsi menjadi informasi sepanjang 32 byte juga. Adapun fungsi fungsi dan prosedur yang digunakan untuk melakukan dekripsi adalah sebagai berikut

- Fungsi DecryptStringFromBytes_Aes(byte[] cipherText, byte[] Key, byte[] IV)
Adapun penjelasan parameter fungsi tersebut adalah:

- cipherText adalah *variable array of byte* yang berisi hasil enkripsi
- key adalah password yang digunakan saat proses enkripsi
- IV adalah *initial vector* yang digunakan saat proses enkripsi

```

static string DecryptStringFromBytes_Aes(byte[] cipherText, byte[]
Key, byte[] IV)
{
    // Check arguments.
    if (cipherText == null || cipherText.Length <= 0)
        throw new ArgumentNullException("cipherText");
    if (Key == null || Key.Length <= 0)
        throw new ArgumentNullException("Key");
    if (IV == null || IV.Length <= 0)
        throw new ArgumentNullException("IV");

    byte[] a = new byte[32];

    for (int i = 0; i < 32; i++) { a[i] = cipherText [i];}

    //string result =
System.Text.Encoding.UTF8.GetString(cipherText);

    // Declare the string used to hold
    // the decrypted text.
    string plaintext = null;

    // Create an Aes object
    // with the specified key and IV.
    using (Aes aesAlg = Aes.Create())
    {
        aesAlg.Key = Key;
        aesAlg.IV = IV;
        aesAlg.Padding = PaddingMode.None;

        // Create a decryptor to perform the stream transform.
        ICryptoTransform decryptor =
aesAlg.CreateDecryptor(aesAlg.Key, aesAlg.IV);

        // Create the streams used for decryption.
        using (MemoryStream msDecrypt = new MemoryStream(a))
        {
            using (CryptoStream csDecrypt = new
CryptoStream(msDecrypt, decryptor, CryptoStreamMode.Read))
            {
                using (StreamReader srDecrypt = new
StreamReader(csDecrypt))
                {

                    // Read the decrypted bytes from the
decrypting stream
                    // and place them in a string.
                    plaintext = srDecrypt.ReadToEnd();
                }
            }
        }
    }
    return plaintext;
}

```

```
}
```

Fungsi ini mengembalikan *string* yang berisi hasil dekripsi. Perlu diperhatikan besar cipherText harus tepat 32byte/16 byte (disesuaikan dengan hasil enkripsi) agar proses dekripsi dapat berjalan.

- Prosedur cobacoba()

Prosedur ini berfungsi untuk memanggil fungsi DecryptStringFromBytes_Aes dan hasilnya disimpan dalam *variable string*. Pada fungsi ini juga key dan IV dideklarasikan.

```
void cobacoba()
{
    if (!RO)
    {
        myAes = Aes.Create();
        RO = true;
    }

    myAes.Key = new byte[] { 0x41, 0x41, 0x41, 0x41,
                           0x41, 0x41, 0x41, 0x41,
                           0x41, 0x41, 0x41, 0x41,
                           0x41, 0x41, 0x41, 0x41 };

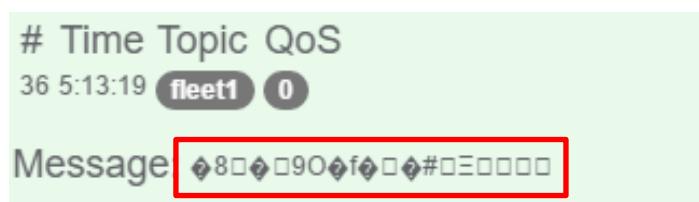
    myAes.IV = new byte[] { 0x41, 0x41, 0x41, 0x41,
                           0x41, 0x41, 0x41, 0x41,
                           0x41, 0x41, 0x41, 0x41,
                           0x41, 0x41, 0x41, 0x41 };

    string roundtrip = DecryptStringFromBytes_Aes(coba,
myAes.Key, myAes.IV);

    Console.WriteLine("Round Trip: {0}", roundtrip);
}
```

Adapun hasil implementasi dekripsi adalah sebagai berikut.

Enkripsi dilakukan di Arduino. Data yang dienkripsi berupa *string* sepanjang 32 karakter yang berisi huruf ‘X’ sebanyak 32 kali. Data tersebut dienkripsi dan dikirimkan menggunakan MQTT untuk dapat didekripsi di GUI.



Gambar 37 pesan yang dikirimkan Arduino pada ke GUI menggunakan MQTT

Dapat diamati bahwa pesan yang dikirimkan Arduino merupakan merupakan kumpulan byte yang tidak dapat direpresentasikan dalam abjad ataupun angka. Pesan tersebut diterima GUI lalu didekripsi dan menghasilkan seperti gambar dibawah

```

Output
Show output from: Debug
besar data      :31
data masuk!!!!!!1
IV      : 41 ,0x41 ,0x41 ,0x41 ,0x41 ,0x41 ,0
panjang data yang mau di enkrip bat :33
panjang data yang mau di enkrip bat aaaaaa :32
Round Trip: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

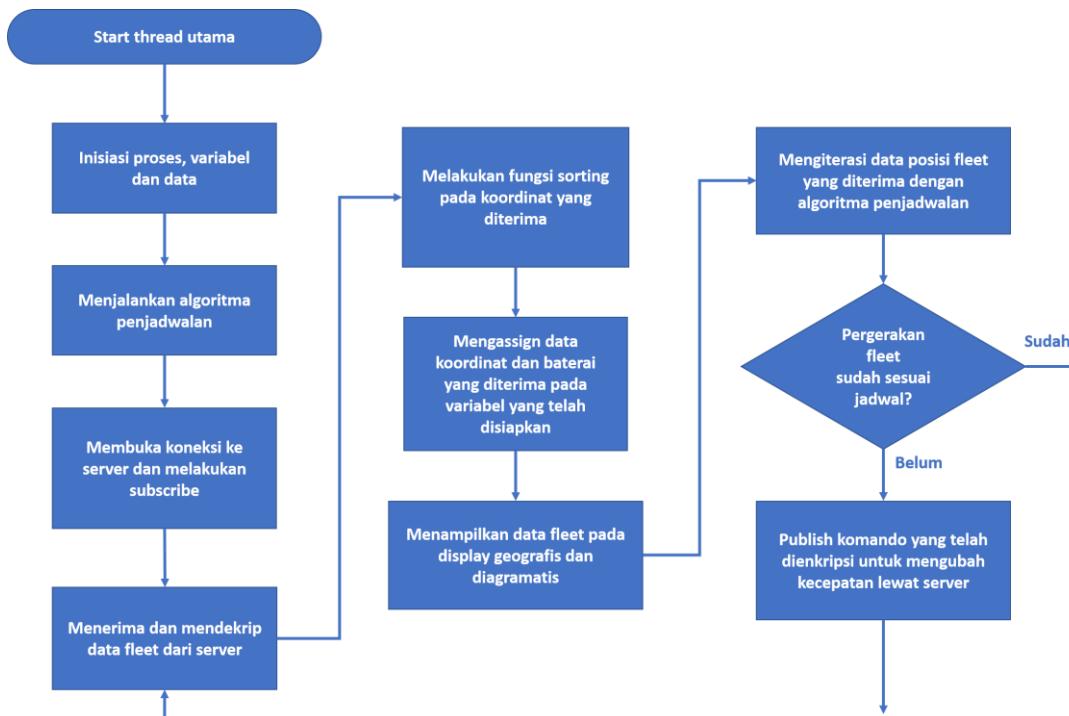
```

Gambar 38 hasil enkripsi

Kotak berwarna merah merupakan hasil dekripsi berupa sekumpulan char ‘X’ sebanyak 32 kali. Proses dekripsi berjalan dengan benar.

2.3.3.5 Integrasi keseluruhan sub-sistem GUI

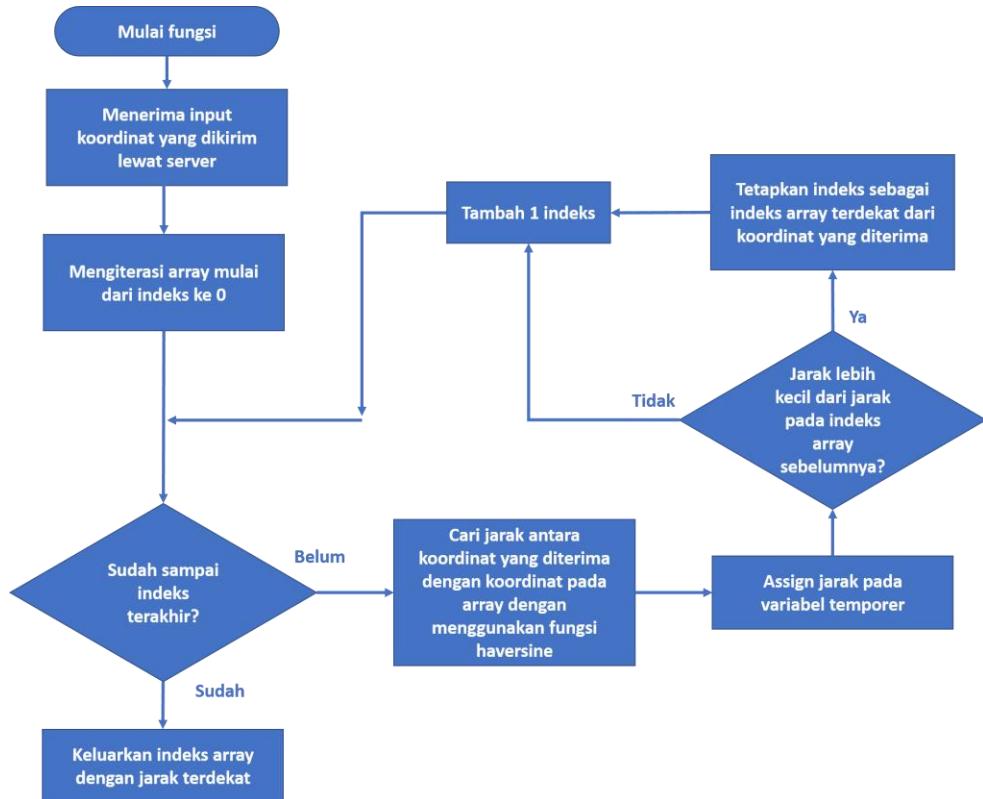
GUI bertugas untuk menampilkan data fleet yang dikirim oleh hardware. Semua data yang diterima lewat server akan diolah dan kemudian ditampilkan. Jika ada fleet yang tidak sesuai dengan jadwalnya, maka akan dikirim *feedback* ke driver berupa besaran kecepatan yang harus diikuti. Secara keseluruhan, flowchart algoritma utama GUI adalah sebagai berikut :



Adapun beberapa fungsi dan class spesifik yang terdapat dalam algoritma utama adalah sebagai berikut :

1. public int sorting(double lat, double lon)

Fungsi ini bertugas untuk membulatkan data koordinat yang diterima ke array terdekat. Hal ini dilakukan untuk menimbulkan ketidak-akuratan data posisi yang dikirim oleh hardware. Fungsi ini memiliki flowchart sebagai berikut :



2. public static class haversine

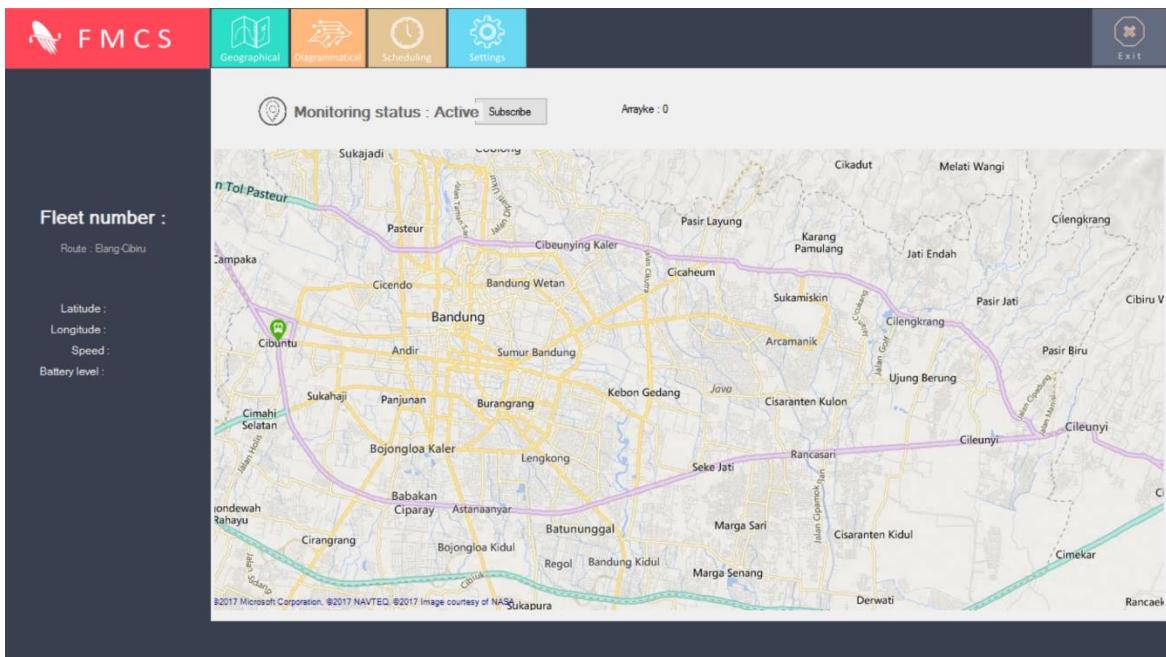
Public class ini memiliki dua fungsi :

- Fungsi toRad bertugas untuk mengubah angka yang diterima menjadi dalam bentuk sudut.
- Fungsi calculate berfungsi untuk menghitung jarak terdekat diantara dua koordinat. Karena koordinat sangat berdekatan, penggunaan azimuth diabaikan sehingga hanya digunakan rumus pencarian biasa diantara dua titik.

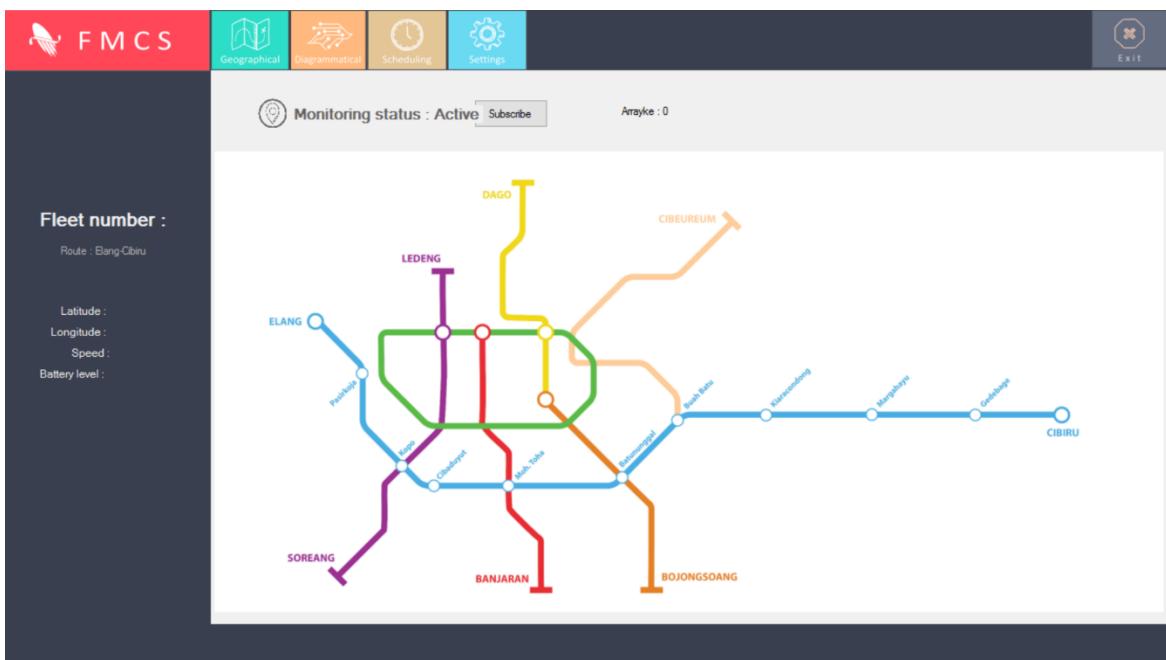
2.3.3.6 Desain akhir tampilan GUI

Main screen GUI berada pada tampilan peta geografis. Tersedia 5 button untuk tab utama, yaitu tampilan geografis, tampilan diagramatis, pengaturan penjadwalan, setting user dan aplikasi, serta exit button.

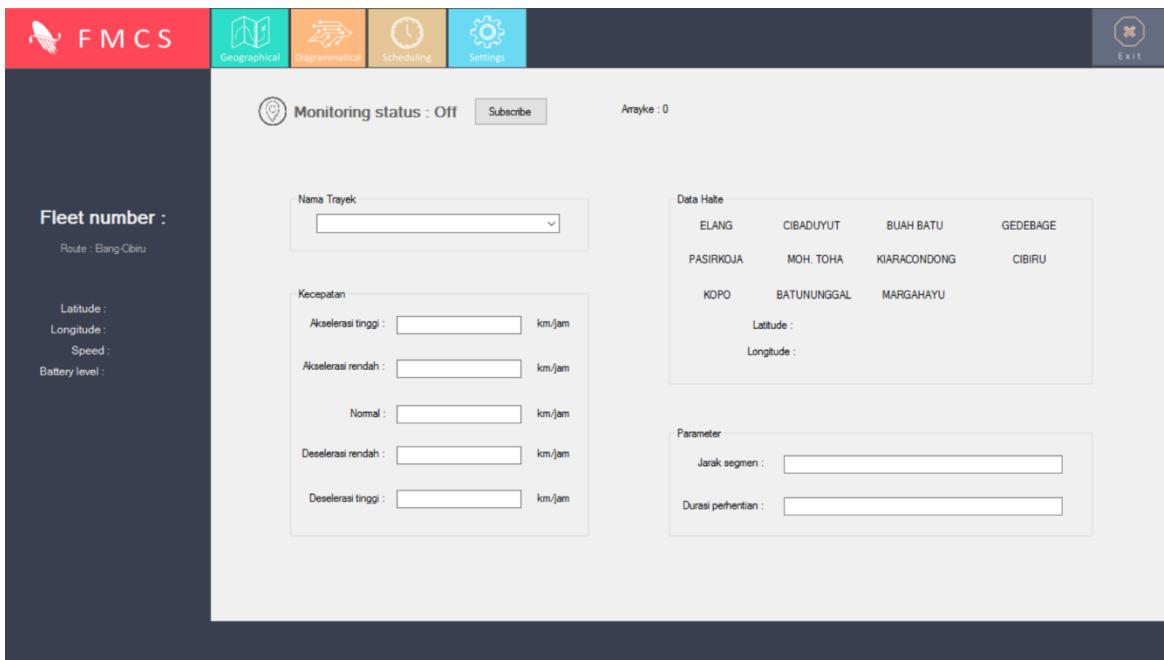
Tampilan akhir GUI adalah sebagai berikut :



Gambar 39 : Tampilan screen peta geografis



Gambar 40 : Tampilan screen peta diagramatis



Gambar 41 : Tampilan screen pengaturan penjadwalan

2.4 IMPLEMENTASI ALGORITMA PENJADWALAN

Algoritma penjadwalan bertujuan agar fleet tersebar merata dan menjaga agra fleet tetap pada jadwalnya.

2.4.1 Struktur Implementasi

Algoritma yang dipilih dalam implementasi ini adalah algoritma dengan pendekatan mengatur kecepatan *fleet*. Algoritma penjadwalan akan menentukan posisi dari tiap armada setiap waktu. Algoritma ini membutuhkan data masukan berupa posisi fleet. Posisi fleet tersebut akan dibandingkan dengan jadwal seharusnya jika terdapat perbedaan antara posisi fleet dengan posisi fleet seharusnya algoritma ini akan menentukan kecepatan fleet untuk menyesuaikan dengan posisi fleet seharusnya dan menghitung perkiraan waktu fleet untuk menyesuaikan dengan jadwal seharusnya. Adapun Data Flow Diagram level 3 seperti dibawah ini

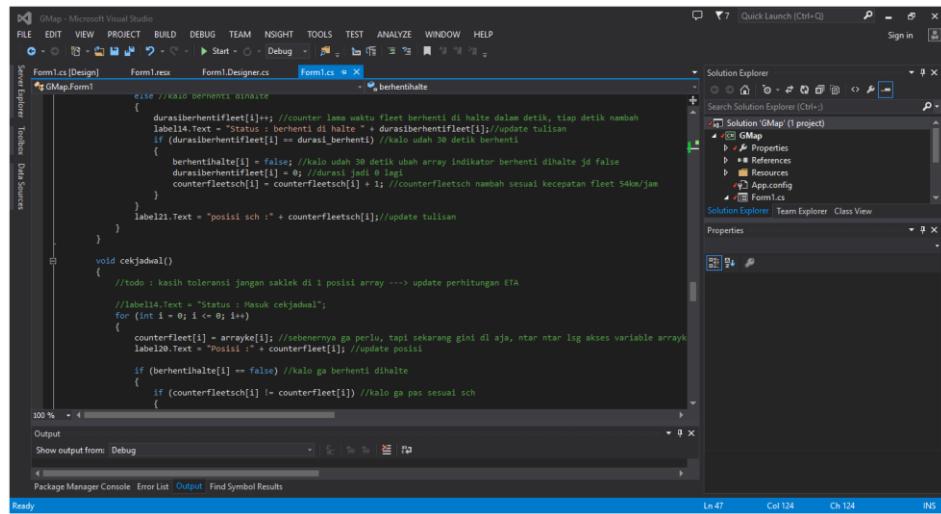


Gambar 42 DFD level 3 agoritma penjadwalan

2.4.2 Environment

Dalam implementasi algoritma penjadwalan digunakan Microsoft Visual Studio. Microsoft Visual Studio merupakan sebuah perangkat lunak yang dapat digunakan untuk pengembangan aplikasi, baik untuk bisnis maupun personal. Visual studio mencakup kompiler, SDK, IDE dan dokumentasi berupa *library*. Pada implementasinya, algoritma

penjadwalan menggunakan Bahasa C# dan .NET pada Microsoft Visual Studio. C# merupakan Bahasa pemrograman yang *object oriented*.



```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Threading;
using System.IO;
using System.Drawing;
using System.Drawing.Imaging;
using System.Drawing.Drawing2D;
using System.Drawing.Text;
using System.Numerics;
using System.Collections;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Threading;
using System.IO;
using System.Drawing;
using System.Drawing.Imaging;
using System.Drawing.Drawing2D;
using System.Drawing.Text;
using System.Numerics;
using System.Collections;

```

Gambar 43 Visual Studio 2013

2.4.3 Prosedur Implementasi

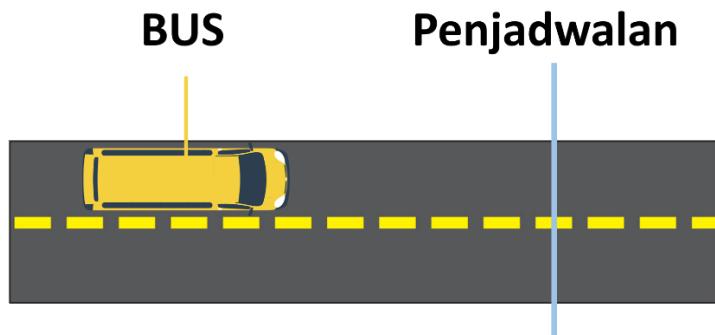
Dalam melakukan implementasi algoritma penjadwalan, dilakukan beberapa tahap implementasi sebagai berikut.

- Mendesain konsep algoritma penjadwalan
- Mendesain dan membuat fungsi dan prosedur pada algoritma penjadwalan.
- Menguji algoritma dengan *dummy* data.

Berikut penjelasan implementasi dan permasalahan yang terjadi selama proses implementasi algoritma penjadwalan. Algoritma penjadwalan menggunakan array trayek yang membagi jalan-jalan menjadi *segment-segment* yang sebelumnya telah dijelaskan pada bagian GUI.

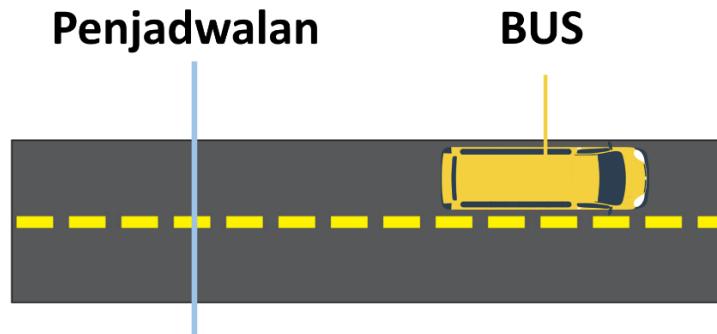
2.4.3.1 Desain Algoritma Penjadwalan

Desain algoritma penjadwalan yang dibuat adalah dengan menetukan jadwal *fleet* tiap waktu. Penjadwalan *fleet* dilakukan dengan menetukan 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*.



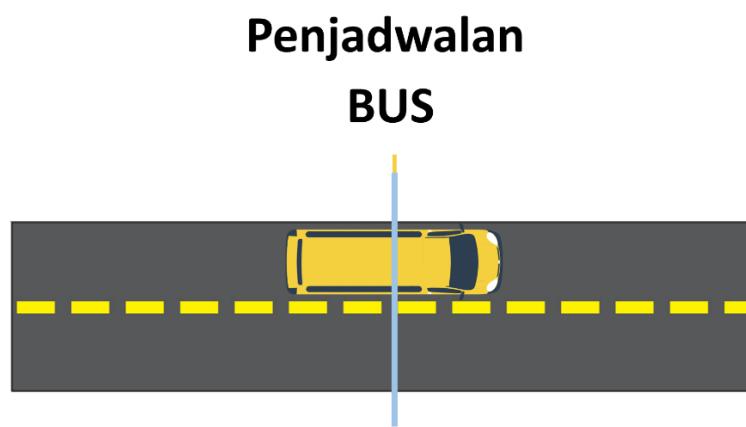
Gambar 44 *fleet* tertinggal dari penjadwalan

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



Gambar 45 *fleet* mendahului penjadwalan

Sebaliknya gambar diatas menunjukan posisi *fleet* mendahului penjadwalan sehingga algortima akan mengurangi kecepatan fleet agar penjadwalan bias menyusul posisi *fleet*.

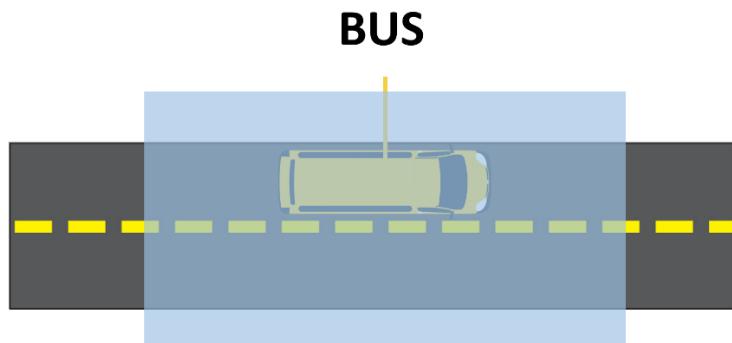


Gambar 46 *fleet* sama dengan penjadwalan

Ketika posisi fleet telah sesuai dengan penjadwalan makan kecepatan fleet dan kecepatan penjadawalan sama.

Kecepatan yang ditentukan oleh algortima apabila terdapat perbedaan posisi akan dikirimkan ke pengemudi *fleet* setelah itu pengemudi harus menaikan/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.

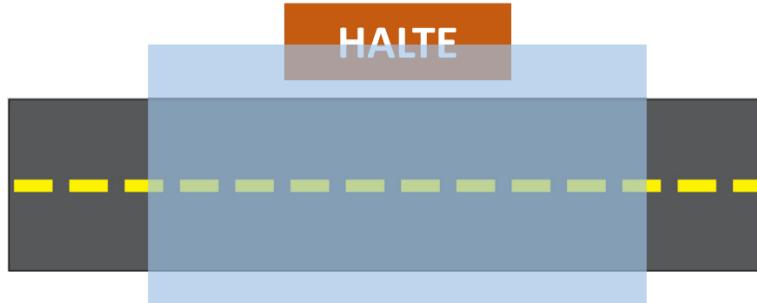
Penjadwalan



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 menetukan posisi selanjutnya.

Penjadwalan



Gambar 48 skema penjadwalan berhenti dalam durasi tertentu ketika berada di halte

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

Algoritma penjadwalan juga dapat menangani kodisi darurat dimana jika salah satu fleet mengalami masalah sehingga tidak bias melanjutkan perjalanan maka semua fleet akan diarahkan untuk berhenti pada halte berikutnya. 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.

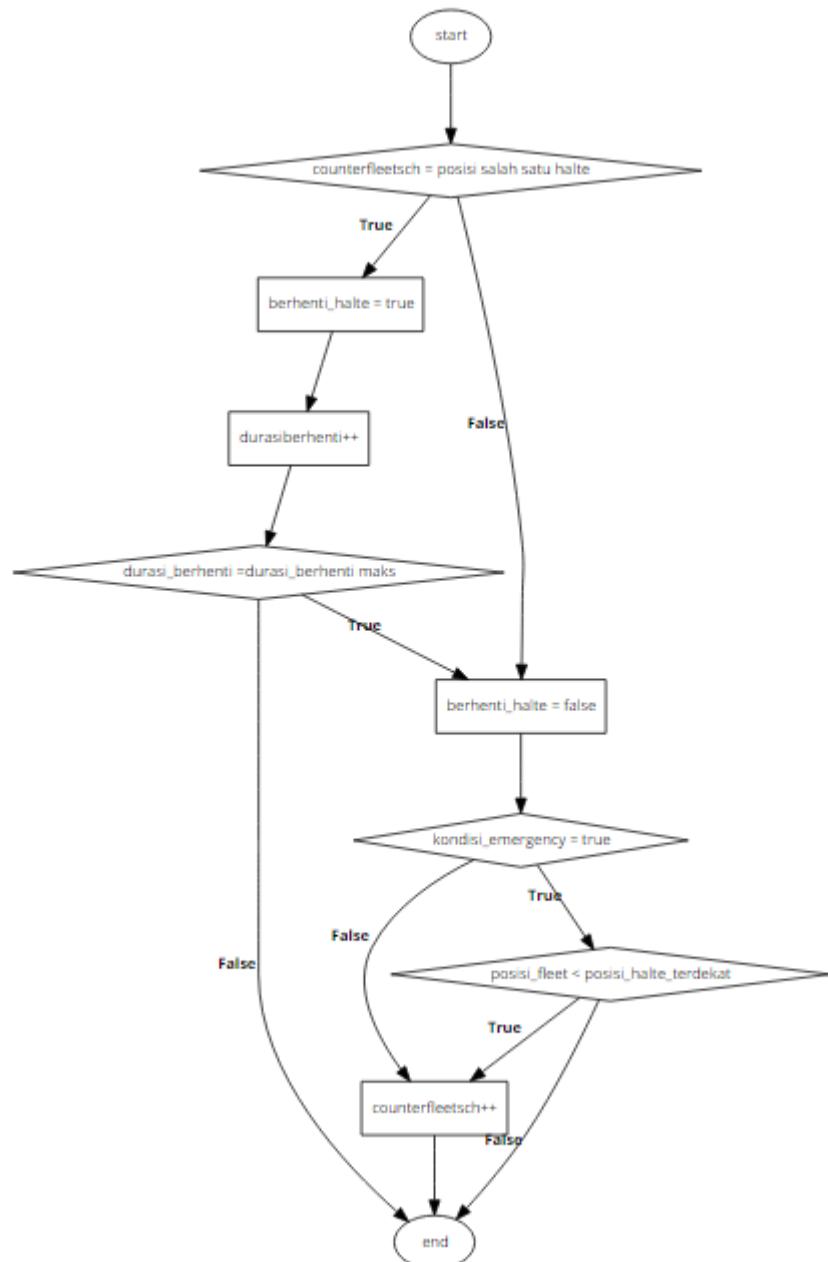
2.4.3.2 Mendesain dan Membuat Fungsi-Fungsi pada Algoritma Penjadwalan.

Dalam algoritma penjadwalan terdiri dari beberapa fungsi dan prosedur yaitu:

- Prosedur penjadwalan()
Prosedur ini akan menambah counter pada variable *counterfleetsch* yang merupakan posisi *fleet* terjadwal. Counter tersebut merupakan representasi posisi *fleet* terjadwal pada array trayek. Frekuensi pemanggilan prosedur penjadwalan bergantung pada

kecepatan normal dari *fleet*. Prosedur ini juga akan mengecek apakah posisi *fleet* seharusnya sama dengan posisi halte jika sama maka counter akan ditahan (tidak ditambah) selama selang waktu tertentu sesuai dengan durasi pemberhentian bis pada tiap halte.

Jika salah satu fleet mengalami kondisi darurat maka penjadwalan tiap fleet akan diarahkan untuk berhenti pada halte terdekat didepannya.



Gambar 39 flowchart algortima penjadwalan

```

void penjadwalan()
{
    for (int i = 0; i <= 5; i++)
    {
  
```

```

        if (posisihalte.Contains(counterfleetsch[i]))
//cek didalam array posisihalte mengandung counterfleetsch[i]
{
    berhentihalte[i] = true;
//counterfleetsch[i] == salah satu array posisihalte -->
indikator berhentihalte[i] jadi true
/*
if (i == 1)
{
    fleetstat.Text = "berhenti di halte";
//update tulisan
}
*/
}
if (berhentihalte[i] == false) //kalo lagi ga
berhenti halte, countefleetsch[i] di tambah
{
    if (fleetcon.Contains(true)) //kalo
bahaya
    {
        if (counterfleetsch[i] <
counterlasthalte[i]) //tambah penjadwalan sampe halte didepannya
        {
            if (counterfleetsch[i] == 12402)
counterfleetsch[i] = 0;
            counterfleetsch[i] =
counterfleetsch[i] + 1; //kecepatan fleet 54km/jam

            if (counterschs[i] == 12402)
counterschs[i] = 0;
            counterschs[i] =
counterschs[i] + 1;
        }
        else if (fleethalth[i])
        {
            if (counterfleetsch[i] == 12402)
counterfleetsch[i] = 0;
            counterfleetsch[i] =
counterfleetsch[i] + 1; //kecepatan fleet 54km/jam

            if (counterschs[i] == 12402)
counterschs[i] = 0;
            counterschs[i] = counterschs[i]
+ 1;
        }
    }
    else //kalo berhenti dihalte
    {
        durasiberhentifleet[i]++;
//counter lama
waktu fleet berhenti di halte dalam detik, tiap detik nambah
//fleetstat.Text = "berhenti di halte "
+ durasiberhentifleet[i]; //update tulisan
        if (durasiberhentifleet[i] ==
durasi_berhenti) //kalo udah 30 detik berhenti
        {
            berhentihalte[i] = false; //kalo
udah 30 detik ubah array indikator berhenti dihalte jd false
        }
    }
}

```

```
        durasiberhentifleet[i] = 0; //durasi  
jadi 0 lagi  
        counterfleetsch[i] =  
counterfleetsch[i] + 1; //counterfleetsch nambah sesuai  
kecepatan fleet 54km/jam  
    }  
}  
}  
}
```

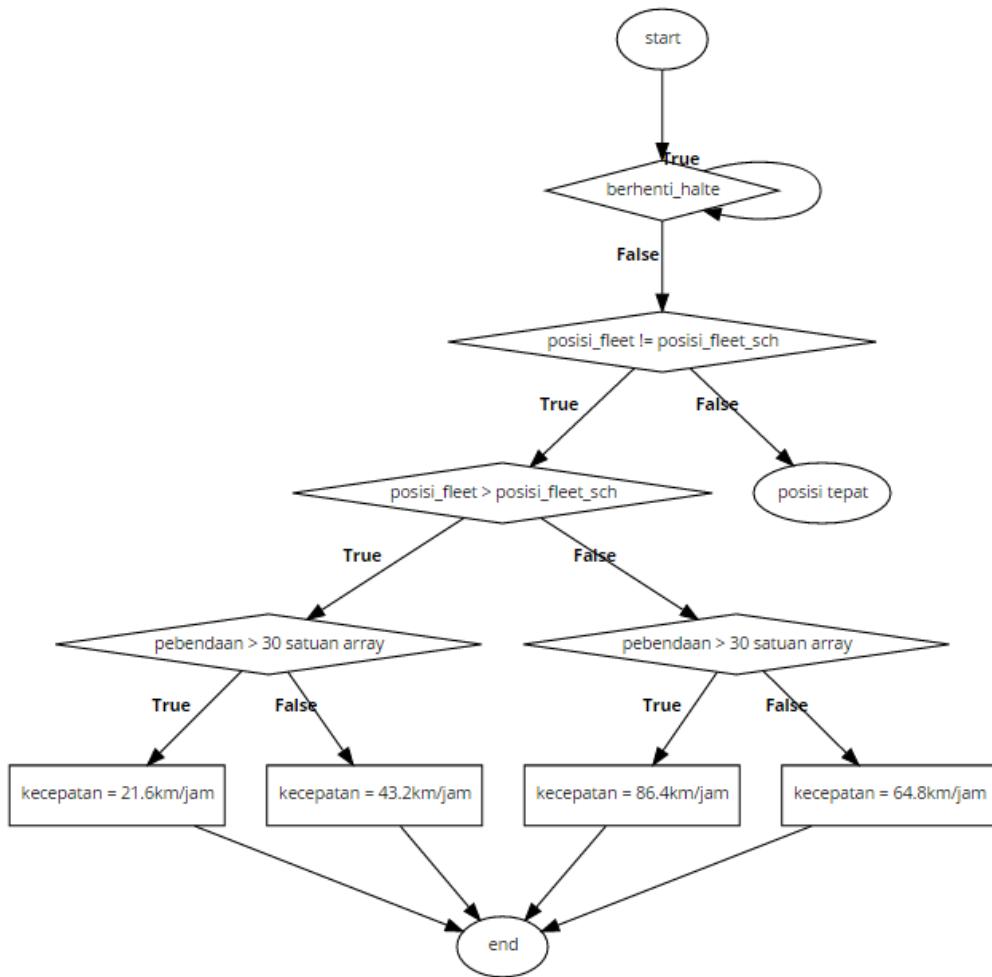
- Prosedur cekjadwal()

Data posisi yang didapatkan dari fleet direpresentasikan 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 7 tabel kecepatan dan representasinya dalam satuan array



Gambar 40 flowchart algoritma cekjadwal()

```

void cekjadwal()
{
    for (int i = 0; i <= 0; i++)
    {
        counterfleet[i] = arrayke[i];
        label20.Text = "Posisi :" + counterfleet[i];
        //update posisi

        if (berhentihalte[i] == false) //kalau tidak berhenti dihalte
        {
            if (counterfleetsch[i] != counterfleet[i]) //kalau tidak pas sesuai sch
            {
                if (counterfleetsch[i] > counterfleet[i])
                //kalau terlalu cepat
                {
                    label14.Text = "Status : Fleet 1 KELAMBATAN";

                    int perbedaan = counterfleetsch[i] -
                    counterfleet[i]; // menghitung perbedaan posisi fleet dan posisi scheduling fleet dalam satuan array
                }
            }
        }
    }
}

```

```

label19.Text = "perbedaan = " + perbedaan;
// menampilkan perbedaan posisi fleet dan posisi scheduling
fleet dalam satuan array

//menentukan kecepatan baru fleet untuk
mengejar ketertinggalan dari jadwal sesuai dengan besar
perbedaan & menghitung ETA
if (perbedaan <= perbedaan_segment)
{
    kecepatanfleet[i] = kecepatan_kenceng1;
// kecepatan fleet i = 64.8km/jam apabila perbedaan <= 30 step
array
    double ETA = ((perbedaan * 3) /
((kecepatanfleet[i] * 1000 / 3600) - (kecepatan_normal * 1000 /
3600));//menghitung estimasi waktu fleet untuk kembali pada
jadwal seharusnya dengan kecepatan_kenceng1
    label18.Text = "ETA : " + ETA;
//tampilkan ETA
}
else
{
    kecepatanfleet[i] = kecepatan_kenceng2;
// kecepatan fleet i = 86.4km/jam apabila perbedaan > 30 step
array
    double ETA = (((perbedaan - 30 +
(perbedaan % 3)) * 3) / ((kecepatanfleet[i] * 1000 / 3600) -
(kecepatan_normal * 1000 / 3600))) + 30 + (perbedaan %
3); //menghitung estimasi waktu fleet untuk kembali pada jadwal
seharusnya dengan kecepatan_keceng2
    label18.Text = "ETA : " + ETA;
//tampilkan ETA
}
label13.Text = "Speedcomm :
" + kecepatanfleet[i];
}
else if (counterfleetsch[i] <
counterfleet[i]) //kalo kelambatan
{
    label14.Text = "Status : Fleet 1
KECEPATAN";

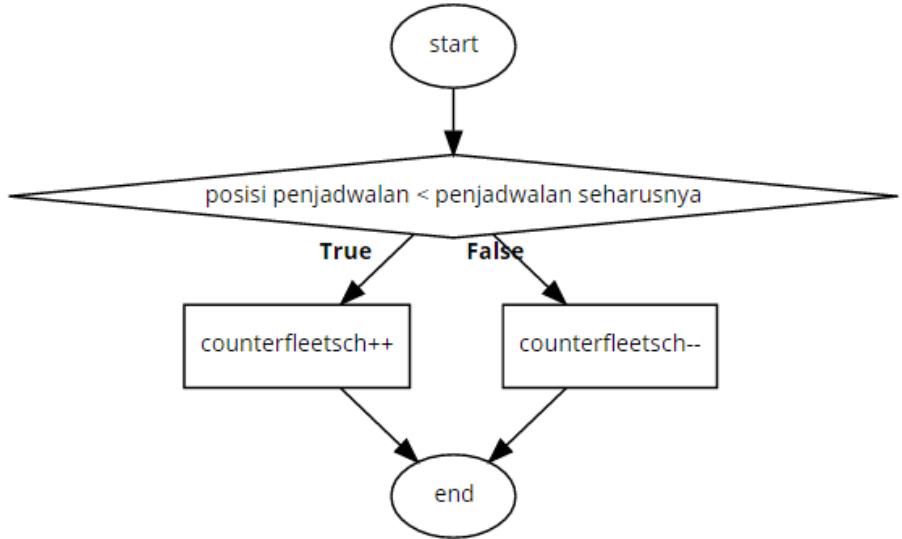
int perbedaan = counterfleet[i] -
counterfleetsch[i]; // menghitung perbedaan posisi fleet dan
posisi scheduling fleet dalam satuan array
label19.Text = "perbedaan = " +
perbedaan; // menampilkan perbedaan posisi fleet dan posisi
scheduling fleet dalam satuan array

//menentukan kecepatan baru fleet untuk
mengejar ketertinggalan dari jadwal sesuai dengan besar
perbedaan & menghitung ETA
if (perbedaan <= perbedaan_segment)
{
    kecepatanfleet[i] =
kecepatan_lambat1; // kecepatan fleet i = 43.2km/jam apabila
perbedaan <= 30 step array
    double ETA = ((perbedaan * 3) / (-
(kecepatanfleet[i] * 1000 / 3600) + (kecepatan_normal * 1000 /
3600))); //menghitung estimasi waktu fleet untuk kembali pada
jadwal seharusnya dengan kecepatan_lambat1
}

```

- Prosedur cek_penjadwalan()

Prosedur `Check_Waiting()` Prosedur ini dipanggil pada frekuensi tertentu untuk mengecek apakah penjadwalan sesuai dengan kondisi yang diinginkan. Posisi penjadwalan dapat berubah ketika terjadi kondisi *emergency*. Apabila posisi penjadwalan tidak sesuai dengan seharusnya maka penjadwalan akan disesuaikan dengan posisi seharusnya. Adapun algoritma dari prosedur ini adalah



Gambar 41 flowchar algoritma cek_penjadwlan()

```

void cek_penjadwalan()
{
    for (int y = 0; y < 6; y++)
    {
        if(counterfleetsch[y] != counterschsch[y])
        {
            pas[y] = false;
        }
        else
        {
            pas[y] = true;
        }
    }

    for (int y = 0; y < 6; y++)
    {
        if (pas[y] == false && berhentihalte[y] ==
false)
        {
            counterschsch_temp[y] += 1; //counter biar
pas 1 detik ditambah
        }

        if (counterschsch_temp[y] == 5)
        {
            if(counterfleetsch[y] < counterschsch[y])
            {
                counterfleetsch[y] += 1;
            }
            else
            {
                counterfleetsch[y] -= 1;
            }

            counterschsch_temp[y] = 0;
        }
    }
}
  
```

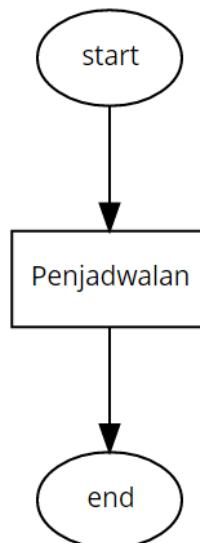
Adapun posisi penjadwalan seharusnya didapatkan dari perhitungan jarak total rute dibagi jumlah fleet yang masih dapat beroperasi.

- Prosedur `itung_tick()`

Prosedur ini akan di eksekusi tiap waktu sesuai dengan kecepatan normal dari fleet sesuai dengan rumus

$$t = \frac{s}{v}$$

Dengan t adalah waktu periode eksekusi, s adalah jarak tiap array, v adalah kecepatan normal. Jika $v = 54 \text{ km/jam} = 15 \text{ m/s}$ dan s adalah 3 m maka $t = 0.2 \text{ s}$. Kami memilih kecepatan 54 km/jam sebagai kecepatan normal dari fleet sehingga prosedur `itung_tick()` akan di panggil tiap 200 ms. Setiap prosedur ini dipanggil maka prosedur penjadwalan akan dieksekusi untuk menambahkan iterasi penjadwalannya sebesar satu kotak array sehingga semua kotak array terlewati oleh penjadawlan.



Gambar 42 flowchart fungsi `itung_tick()`

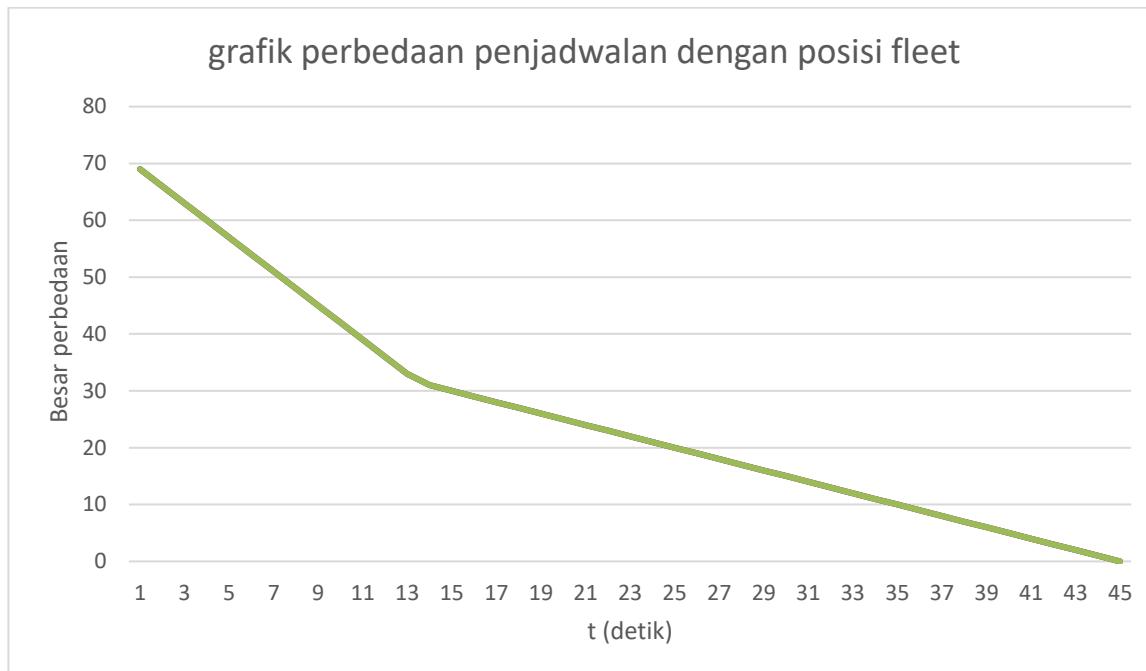
2.4.3.3 Menguji Algoritma dengan *Dummy Data*

Algortima yang telah di implementasikan diuji dengan *dummy* data dan fungsi yang merepresentasikan *fleet*. Fungsi-fungsi tersebut adalah

- Fungsi `kecepatanToCounter(double kecepatan)`
fungsi ini berfungsi untuk menghitung perpindahan kotak tiap detik sesuai dengan kecepatan *fleet*. Kecepatan *fleet* ditentukan melalui *variable* ‘`kecepatanfleet`’. Fungsi ini akan dipanggil tiap detik sehingga seolah-olah *fleet* bergerak tiap detik sesuai dengan kecepatannya.
- Prosedur `fleet()`
Prosedur ini akan memanggil fungsi `kecepatanToCounter` dengan parameter fungsi berupa kecepatan *fleet*. Fungsi tersebut akan mengembalikan nilai perpindahan *fleet*

dalam satuan array posisi *fleet*. Prosedur ini akan dipanggil tiap detik sehingga seolah olah fleet berjalan tiap detiknya sesuai dengan kecepatannya.

Kecepatan yang ditentukan dari algortima penjadwalan akan dimasukkan pada fungsi kecepatanToCounter(). Hasil pengujian algortima penjadwalan menggunakan *dummy* data adalah sebagai berikut:



Grafik 1 grafik perbedaan fleet dengan penjadwalan

Skema yang dijalankan adalah ketika fleet mengalami keterlambatan. Penjadwalan diletakkan pada array ke-69 sedangkan posisi fleet berada pada array ke-0 pada detik ke-0. Perbedaan terus berkurang dengan bertambahnya waktu. Dapat diamati dari grafik diatas dari detik ke-0 sampai dengan detik ke-13 pengurangan perbedaan lebih curam dari detik ke-13 sampai detik ke-45. Hal ini terjadi karena algortima mendekteksi perbedaan lebih dari 30 satuan array sehingga kecepatan yang digunakan adalah 86.4km/jam. Dari detik ke-13 sampai detik ke-45 karena perbedaan kurang 30 maka kecepatan yang digunakan adalah 64.8km/jam. Algortima penjadwalan berhasil meposisikan fleet pada jadwal seharusnya.

3 SIMPULAN

Implementasi sub-sistem elektrik sejauh ini sudah dapat melakukan fungsi-fungsi yang harus dilakukan oleh *ECU monitoring*. Fungsi-fungsi yang dimaksud adalah mengambil data-data *fleet* (posisi dan data baterai), enkripsi data tersebut dan dikirimkan ke server. Kendala yang banyak dialami pada implementasi sub-sistem elektrik adalah saat integrasi sub-sistem ini dengan server. Pengujian yang harus dilakukan pada bagian selanjutnya adalah menguji spesifikasi *ECU monitoring* yang harus dipenuhi yaitu kecepatan pengiriman data (dengan periode minimal 0.67 detik) dan akurasi pembacaan GPS (*error* maksimal 10 meter).

Versi pertama dari implementasi GUI sudah dapat melakukan sebagian besar fungsi-fungsi yang tertulis pada spesifikasinya. Diantaranya adalah menerima data dari server dan

mendekriptnya, menampilkan posisi fleet pada peta geografis, menampilkan data fleet, melakukan pengaturan penjadwalan, serta mengirim komando kepada modul hardware. Adapun fungsi yang belum terlaksana adalah menampilkan marker pada peta diagramatis. Pada tahap selanjutnya akan dilakukan perbaikan metode untuk menampilkan marker pada peta diagramatis serta pengujian spesifikasi GUI.

Server yang di implementasikan pada computer penulis dapat berjalan dengan baik. Server dapat meneruskan data yang diterima sesuai dengan topiknya. Server juga dapat diakses melalui internet.

Algoritma penjadwalan yang diimplementasikan dalam GUI berjalan dengan lancar. Algoritam dapat mendeteksi ketidaksesuai posisi *fleet* dengan penjadwalannya. Apabila *fleet* terlambat dari penjadwalannya maka penjadawlan akan menetukan kecepatan *fleet* untuk mengejar ketertinggalan tersebut dan menghitung perkiraan waktu *fleet* agar sesuai dengan jadwalnya lagi.

Selanjutnya akan dilakukan *packaging ECU monitoring* dan pegujian pada bagian berikutnya dari proses pembuatan FMCS ini.

Ini uwi Fitnah lah aku bahkan baru buka bagian ini:(spiiik

4 LAMPIRAN

Lampiran A – Sub-Sistem Elektrik

Implementasi kode prototipe *hardware*

```
/* 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
 * mempublish "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, 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("  ");
}
}

```

Library koneksi MQTT dengan modul GSM

```

/*
MQTT.h - Library for GSM MQTT Client.
Created by Nithin K. Kurian, Dhanish Vijayan, Elementz Engineers
Guild Pvt. Ltd, July 2, 2016.
Released into the public domain.
*/

```

```

#include "GSM_MQTT.h"
#include "Arduino.h"
#include <SoftwareSerial.h>
#include <avr/pgmspace.h>
extern uint8_t GSM_Response;

extern SoftwareSerial cobac;
extern String MQTT_HOST;
extern String MQTT_PORT;

extern GSM_MQTT MQTT;
uint8_t GSM_Response = 0;
unsigned long previousMillis = 0;
//char inputString[UART_BUFFER_LENGTH];           // a string to hold
incoming data
boolean stringComplete = false; // whether the string is complete
void serialEvent();
GSM_MQTT::GSM_MQTT(unsigned long KeepAlive)
{
    _KeepAliveTimeOut = KeepAlive;
}

void GSM_MQTT::begin(void)
{
    cobac.begin(9600);
    Serial.begin(9600);
    //Serial.write("AT+IPR=9600");
    delay(1000);
    while(Serial.available())
    {
        Serial.write(Serial.read());
    }
    Serial.write("AT\r\n");
    delay(1000);
    while(Serial.available())
    {
        Serial.write(Serial.read());
    }
    _tcpInit();
}
char GSM_MQTT::_sendAT(char *command, unsigned long waitms)
{
    cobac.write(command);
    unsigned long PrevMillis = millis();
    strcpy(reply, "none");
    GSM_Response = 0;
    Serial.write(command);
    unsigned long currentMillis = millis();
    // cobac.println(PrevMillis);
    // cobac.println(currentMillis);
    while ( (GSM_Response == 0) && ((currentMillis - PrevMillis) < waitms) )
    {
        // delay(1);
        serialEvent();
        currentMillis = millis();
    }
    cobac.print("GSM_Response : ");
    cobac.println(GSM_Response);
    return GSM_Response;
}

```

```

}

char GSM MQTT::sendATreply(char *command, char *replystr, unsigned
long waitms)
{
    cobac.write(command);
    strcpy(reply, replystr);
    unsigned long PrevMillis = millis();
    GSM_ReplyFlag = 0;
    Serial.write(command);
    unsigned long currentMillis = millis();

    // cobac.println(PrevMillis);
    // cobac.println(currentMillis);
    while ( (GSM_ReplyFlag == 0) && ((currentMillis - PrevMillis) <
waitms) )
    {
        // delay(1);
        serialEvent();
        currentMillis = millis();
    }
    cobac.print("GSM_ReplyFlag : ");
    cobac.println(GSM_ReplyFlag);
    return GSM_ReplyFlag;
}
void GSM_MQTT::_tcpInit(void)
{
    cobac.print("Modem Status : ");
    cobac.println(modemStatus);
    switch (modemStatus)
    {
        case 0:
        {
            delay(1000);
            Serial.print("+++");
            delay(500);
            if (_sendAT("AT\r\n", 5000) == 1)
            {
                modemStatus = 1;
            }
            else
            {
                modemStatus = 0;
                break;
            }
        }
        case 1:
        {
            if (_sendAT("ATE1\r\n", 2000) == 1)
            {
                modemStatus = 2;
            }
            else
            {
                modemStatus = 1;
                break;
            }
        }
        case 2:
        {
            if (sendATreply("AT+CREG?\r\n", "0,1", 5000) == 1)

```

```

    {
        sendAT("AT+CIPMUX=0\r\n", 2000);
        _sendAT("AT+CIPMODE=1\r\n", 2000);
        if (_sendATReply("AT+CGATT?\r\n", ":", 1, 4000) != 1)
        {
            _sendAT("AT+CGATT=1\r\n", 2000);
        }
        modemStatus = 3;
        _tcpStatus = 2;
    }
    else
    {
        modemStatus = 2;
        break;
    }
}
case 3:
{
    if (GSM_ReplyFlag != 7)
    {
        _tcpStatus = sendATReply("AT+CIPSTATUS\r\n", "STATE", 4000);
        if (_tcpStatusPrev == _tcpStatus)
        {
            tcpATerrorcount++;
            if (tcpATerrorcount >= 10)
            {
                tcpATerrorcount = 0;
                _tcpStatus = 7;
            }
        }
        else
        {
            _tcpStatusPrev = _tcpStatus;
            tcpATerrorcount = 0;
        }
    }
    _tcpStatusPrev = _tcpStatus;
    cobac.print("tcpStatus = ");
    cobac.println(_tcpStatus);
    switch (_tcpStatus)
    {
        case 2:
        {
            _sendAT("AT+CSTT=\\"indosatgprs\\"\\r\\n", 5000);
            break;
        }
        case 3:
        {
            _sendAT("AT+CIICR\\r\\n", 5000) ;
            break;
        }
        case 4:
        {
            sendATReply("AT+CIFSR\\r\\n", ".", 4000) ;
            break;
        }
        case 5:
        {
            Serial.print("AT+CIPSTART=\\"TCP\\", \\"");
        }
    }
}

```

```

        Serial.print(MQTT_HOST);
        Serial.print("\",\"");
        Serial.print(MQTT_PORT);
        if (_sendAT("\r\n", 5000) == 1)
        {
            unsigned long PrevMillis = millis();
            unsigned long currentMillis = millis();
            while ( (GSM_Response != 4) && ((currentMillis - PrevMillis) < 20000) )
            {
                //    delay(1);
                serialEvent();
                currentMillis = millis();
            }
        }
        break;
    }
    case 6:
    {
        unsigned long PrevMillis = millis();
        unsigned long currentMillis = millis();
        while ( (GSM_Response != 4) && ((currentMillis - PrevMillis) < 20000) )
        {
            //    delay(1);
            serialEvent();
            currentMillis = millis();
        }
        break;
    }
    case 7:
    {
        sendATReply("AT+CIPSHUT\r\n", "OK", 4000);
        modemStatus = 0;
        _tcpStatus = 2;
        break;
    }
}
}

void GSM_MQTT::_ping(void)
{
    if (pingFlag == true)
    {
        unsigned long currentMillis = millis();
        if ((currentMillis - _PingPrevMillis) >= _KeepAliveTimeOut * 1000)
        {
            // save the last time you blinked the LED
            _PingPrevMillis = currentMillis;
            Serial.print(char(PINGREQ * 16));
            _sendLength(0);
        }
    }
}
void GSM_MQTT::_sendUTFString(char *string)
{

```

```

int localLength = strlen(string);
Serial.print(char(localLength / 256));
Serial.print(char(localLength % 256));
Serial.print(string);
}
void GSM_MQTT::_sendLength(int len)
{
    bool length_flag = false;
    while (length_flag == false)
    {
        if ((len / 128) > 0)
        {
            Serial.print(char(len % 128 + 128));
            len /= 128;
        }
        else
        {
            length_flag = true;
            Serial.print(char(len));
        }
    }
}
void GSM_MQTT::connect(char *ClientIdentifier, char UserNameFlag, char PasswordFlag, char *UserName, char *Password, char CleanSession, char WillFlag, char WillQoS, char WillRetain, char *WillTopic, char *WillMessage)
{
    ConnectionAcknowledgement = NO_ACKNOWLEDGEMENT ;
    Serial.print(char(CONNECT * 16));
    char ProtocolName[7] = "MQIsdp";
    int localLength = (2 + strlen(ProtocolName)) + 1 + 3 + (2 +
    strlen(ClientIdentifier));
    if (WillFlag != 0)
    {
        localLength = localLength + 2 + strlen(WillTopic) + 2 +
        strlen(WillMessage);
    }
    if (UserNameFlag != 0)
    {
        localLength = localLength + 2 + strlen(UserName);

        if (PasswordFlag != 0)
        {
            localLength = localLength + 2 + strlen(Password);
        }
    }
    _sendLength(localLength);
    _sendUTFString(ProtocolName);
    Serial.print(char(_ProtocolVersion));
    Serial.print(char(UserNameFlag * User_Name_Flag_Mask + PasswordFlag *
    Password_Flag_Mask + WillRetain * Will_Retain_Mask + WillQoS * *
    Will_QoS_Scale + WillFlag * Will_Flag_Mask + CleanSession * *
    Clean_Session_Mask));
    Serial.print(char(_KeepAliveTimeOut / 256));
    Serial.print(char(_KeepAliveTimeOut % 256));
    _sendUTFString(ClientIdentifier);
    if (WillFlag != 0)
    {
        _sendUTFString(WillTopic);
        _sendUTFString(WillMessage);
    }
}

```

```

    }
    if (UserNameFlag != 0)
    {
        _sendUTFString(UserName);
        if (PasswordFlag != 0)
        {
            _sendUTFString(Password);
        }
    }
}
void GSM_MQTT::publish(char DUP, char Qos, char RETAIN, unsigned int MessageID, char *Topic, char *Message)
{
    Serial.print(char(PUBLISH * 16 + DUP * DUP_Mask + Qos * QoS_Scale + RETAIN));
    int localLength = (2 + strlen(Topic));
    if (Qos > 0)
    {
        localLength += 2;
    }
    localLength += strlen(Message);
    _sendLength(localLength);
    _sendUTFString(Topic);
    if (Qos > 0)
    {
        Serial.print(char(MessageID / 256));
        Serial.print(char(MessageID % 256));
    }
    Serial.print(Message);
}
void GSM_MQTT::publishACK(unsigned int MessageID)
{
    Serial.print(char(PUBACK * 16));
    _sendLength(2);
    Serial.print(char(MessageID / 256));
    Serial.print(char(MessageID % 256));
}
void GSM_MQTT::publishREC(unsigned int MessageID)
{
    Serial.print(char(PUBREC * 16));
    _sendLength(2);
    Serial.print(char(MessageID / 256));
    Serial.print(char(MessageID % 256));
}
void GSM_MQTT::publishREL(char DUP, unsigned int MessageID)
{
    Serial.print(char(PUBREL * 16 + DUP * DUP_Mask + 1 * QoS_Scale));
    _sendLength(2);
    Serial.print(char(MessageID / 256));
    Serial.print(char(MessageID % 256));
}
void GSM_MQTT::publishCOMP(unsigned int MessageID)
{
    Serial.print(char(PUBCOMP * 16));
    _sendLength(2);
    Serial.print(char(MessageID / 256));
    Serial.print(char(MessageID % 256));
}

```

```

void GSM_MQTT::subscribe(char DUP, unsigned int MessageID, char
*SubTopic, char SubQoS)
{
    Serial.print(char(SUBSCRIBE * 16 + DUP * DUP_Mask + 1 * QoS_Scale));
    int localLength = 2 + (2 + strlen(SubTopic)) + 1;
    _sendLength(localLength);
    Serial.print(char(MessageID / 256));
    Serial.print(char(MessageID % 256));
    _sendUTFString(SubTopic);
    Serial.print(SubQoS);
    cobac.print("Subscribing : ");
    cobac.print(char(SUBSCRIBE * 16 + DUP * DUP_Mask + 1 * QoS_Scale));
    cobac.print(char(MessageID / 256));
    cobac.print(char(MessageID % 256));
    cobac.println(SubQoS);
}
void GSM_MQTT::unsubscribe(char DUP, unsigned int MessageID, char
*SubTopic)
{
    Serial.print(char(UNSUBSCRIBE * 16 + DUP * DUP_Mask + 1 *
QoS_Scale));
    int localLength = (2 + strlen(SubTopic)) + 2;
    _sendLength(localLength);

    Serial.print(char(MessageID / 256));
    Serial.print(char(MessageID % 256));

    _sendUTFString(SubTopic);
}
void GSM_MQTT::disconnect(void)
{
    Serial.print(char(DISCONNECT * 16));
    _sendLength(0);
    pingFlag = false;
}
//Messages
const char CONNECTMessage[] PROGMEM = {"Client request to connect to
Server\r\n"};
const char CONNACKMessage[] PROGMEM = {"Connect Acknowledgment\r\n"};
const char PUBLISHMessage[] PROGMEM = {"Publish message\r\n"};
const char PUBACKMessage[] PROGMEM = {"Publish Acknowledgment\r\n"};
const char PUBRECMessag[] PROGMEM = {"Publish Received (assured
delivery part 1)\r\n"};
const char PUBRELMessage[] PROGMEM = {"Publish Release (assured
delivery part 2)\r\n"};
const char PUBCOMPMessage[] PROGMEM = {"Publish Complete (assured
delivery part 3)\r\n"};
const char SUBSCRIBEMessage[] PROGMEM = {"Client Subscribe
request\r\n"};
const char SUBACKMessage[] PROGMEM = {"Subscribe
Acknowledgment\r\n"};
const char UNSUBSCRIBEMessage[] PROGMEM = {"Client Unsubscribe
request\r\n"};
const char UNSUBACKMessage[] PROGMEM = {"Unsubscribe
Acknowledgment\r\n"};
const char PINGREQMessage[] PROGMEM = {"PING Request\r\n"};
const char PINGRESPMessage[] PROGMEM = {"PING Response\r\n"};
const char DISCONNECTMessage[] PROGMEM = {"Client is
Disconnecting\r\n"};

```

```

void GSM_MQTT::printMessageType(uint8_t Message)
{
    switch (Message)
    {
        case CONNECT:
        {
            int k, len = strlen_P(CONNECTMessage);
            char myChar;
            for (k = 0; k < len; k++)
            {
                myChar = pgm_read_byte_near(CONNECTMessage + k);
                cobac.print(myChar);
            }
            break;
        }
        case CONNACK:
        {
            int k, len = strlen_P(CONNACKMessage);
            char myChar;
            for (k = 0; k < len; k++)
            {
                myChar = pgm_read_byte_near(CONNACKMessage + k);
                cobac.print(myChar);
            }
            break;
        }
        case PUBLISH:
        {
            int k, len = strlen_P(PUBLISHMessage);
            char myChar;
            for (k = 0; k < len; k++)
            {
                myChar = pgm_read_byte_near(PUBLISHMessage + k);
                cobac.print(myChar);
            }
            break;
        }
        case PUBACK:
        {
            int k, len = strlen_P(PUBACKMessage);
            char myChar;
            for (k = 0; k < len; k++)
            {
                myChar = pgm_read_byte_near(PUBACKMessage + k);
                cobac.print(myChar);
            }
            break;
        }
        case PUBREC:
        {
            int k, len = strlen_P(PUBRECMassage);
            char myChar;
            for (k = 0; k < len; k++)
            {
                myChar = pgm_read_byte_near(PUBRECMassage + k);
                cobac.print(myChar);
            }
            break;
        }
        case PUBREL:
    }
}

```

```

{
    int k, len = strlen_P(PUBRELMessage);
    char myChar;
    for (k = 0; k < len; k++)
    {
        myChar = pgm_read_byte_near(PUBRELMessage + k);
        cobac.print(myChar);
    }
    break;
}
case PUBCOMP:
{
    int k, len = strlen_P(PUBCOMPMessag e );
    char myChar;
    for (k = 0; k < len; k++)
    {
        myChar = pgm_read_byte_near(PUBCOMPMessag e + k);
        cobac.print(myChar);
    }
    break;
}
case SUBSCRIBE:
{
    int k, len = strlen_P(SUBSCRIBEMessag e );
    char myChar;
    for (k = 0; k < len; k++)
    {
        myChar = pgm_read_byte_near(SUBSCRIBEMessag e + k);
        cobac.print(myChar);
    }
    break;
}
case SUBACK:
{
    int k, len = strlen_P(SUBACKMessage );
    char myChar;
    for (k = 0; k < len; k++)
    {
        myChar = pgm_read_byte_near(SUBACKMessage + k);
        cobac.print(myChar);
    }
    break;
}
case UNSUBSCRIBE:
{
    int k, len = strlen_P(UNSUBSCRIBEMessag e );
    char myChar;
    for (k = 0; k < len; k++)
    {
        myChar = pgm_read_byte_near(UNSUBSCRIBEMessag e + k);
        cobac.print(myChar);
    }
    break;
}
case UNSUBACK:
{
    int k, len = strlen_P(UNSUBACKMessage );
    char myChar;
    for (k = 0; k < len; k++)
    {

```

```

        myChar = pgm_read_byte_near(UNSUBACKMessage + k);
        cobac.print(myChar);
    }
    break;
}
case PINGREQ:
{
    int k, len = strlen_P(PINGREQMessage);
    char myChar;
    for (k = 0; k < len; k++)
    {
        myChar = pgm_read_byte_near(PINGREQMessage + k);
        cobac.print(myChar);
    }
    break;
}
case PINGRESP:
{
    int k, len = strlen_P(PINGRESPMessage);
    char myChar;
    for (k = 0; k < len; k++)
    {
        myChar = pgm_read_byte_near(PINGRESPMessage + k);
        cobac.print(myChar);
    }
    break;
}
case DISCONNECT:
{
    int k, len = strlen_P(DISCONNECTMessage);
    char myChar;
    for (k = 0; k < len; k++)
    {
        myChar = pgm_read_byte_near(DISCONNECTMessage + k);
        cobac.print(myChar);
    }
    break;
}
}

//Connect Ack
const char ConnectAck0[] PROGMEM = {"Connection Accepted\r\n"};
const char ConnectAck1[] PROGMEM = {"Connection Refused: unacceptable
protocol version\r\n"};
const char ConnectAck2[] PROGMEM = {"Connection Refused: identifier
rejected\r\n"};
const char ConnectAck3[] PROGMEM = {"Connection Refused: server
unavailable\r\n"};
const char ConnectAck4[] PROGMEM = {"Connection Refused: bad user
name or password\r\n"};
const char ConnectAck5[] PROGMEM = {"Connection Refused: not
authorized\r\n"};
void GSM_MQTT::printConnectAck(uint8_t Ack)
{
    switch (Ack)
    {
        case 0:
        {
            int k, len = strlen_P(ConnectAck0);

```

```

    char myChar;
    for (k = 0; k < len; k++)
    {
        myChar = pgm_read_byte_near(ConnectAck0 + k);
        cobac.print(myChar);
    }
    break;
}
case 1:
{
    int k, len = strlen_P(ConnectAck1);
    char myChar;
    for (k = 0; k < len; k++)
    {
        myChar = pgm_read_byte_near(ConnectAck1 + k);
        cobac.print(myChar);
    }
    break;
}
case 2:
{
    int k, len = strlen_P(ConnectAck2);
    char myChar;
    for (k = 0; k < len; k++)
    {
        myChar = pgm_read_byte_near(ConnectAck2 + k);
        cobac.print(myChar);
    }
    break;
}
case 3:
{
    int k, len = strlen_P(ConnectAck3);
    char myChar;
    for (k = 0; k < len; k++)
    {
        myChar = pgm_read_byte_near(ConnectAck3 + k);
        cobac.print(myChar);
    }
    break;
}
case 4:
{
    int k, len = strlen_P(ConnectAck4);
    char myChar;
    for (k = 0; k < len; k++)
    {
        myChar = pgm_read_byte_near(ConnectAck4 + k);
        cobac.print(myChar);
    }
    break;
}
case 5:
{
    int k, len = strlen_P(ConnectAck5);
    char myChar;
    for (k = 0; k < len; k++)
    {
        myChar = pgm_read_byte_near(ConnectAck5 + k);
        cobac.print(myChar);
    }
}

```

```

        }
        break;
    }
}
unsigned int GSM_MQTT::_generateMessageID(void)
{
    if (_LastMessageID < 65535)
    {
        return ++_LastMessageID;
    }
    else
    {
        _LastMessageID = 0;
        return _LastMessageID;
    }
}
void GSM_MQTT::processing(void)
{
    if (TCP_Flag == false)
    {
        MQTT_Flag = false;
        _tcpInit();
    }
    _ping();
    delay(200);
    serialEvent();
}

bool GSM_MQTT::available(void)
{
    return MQTT_Flag;
}
void serialEvent()
{
    while (Serial.available())
    {
        char inChar = (char)Serial.read();
        if (MQTT.TCP_Flag == false)
        {
            if (MQTT.index < 200)
            {
                MQTT.inputString[MQTT.index++] = inChar;
            }
            if (inChar == '\n')
            {
                MQTT.inputString[MQTT.index] = 0;
                stringComplete = true;
                cobac.print(MQTT.inputString);
                if (strstr(MQTT.inputString, MQTT.reply) != NULL)
                {
                    MQTT.GSM_ReplyFlag = 1;
                    if (strstr(MQTT.inputString, " INITIAL") != 0)
                    {
                        MQTT.GSM_ReplyFlag = 2; //
                    }
                    else if (strstr(MQTT.inputString, " START") != 0)
                    {
                        MQTT.GSM_ReplyFlag = 3; //
                    }
                }
            }
        }
    }
}

```

```

    }
    else if (strstr(MQTT.inputString, "IP CONFIG") != 0)
    {
        _delay_us(10);

        MQTT.GSM_ReplyFlag = 4;
    }
    else if (strstr(MQTT.inputString, " GPRSACT") != 0)
    {

        MQTT.GSM_ReplyFlag = 4; //
    }
    else if ((strstr(MQTT.inputString, " STATUS") != 0) ||
    (strstr(MQTT.inputString, "TCP CLOSED") != 0))
    {

        MQTT.GSM_ReplyFlag = 5; //
    }
    else if (strstr(MQTT.inputString, " TCP CONNECTING") != 0)
    {

        MQTT.GSM_ReplyFlag = 6; //
    }
    else if ((strstr(MQTT.inputString, " CONNECT OK") != 0) ||
    (strstr(MQTT.inputString, "CONNECT FAIL") != NULL) ||
    (strstr(MQTT.inputString, "PDP DEACT") != 0))
    {

        MQTT.GSM_ReplyFlag = 7;
    }
}
else if (strstr(MQTT.inputString, "OK") != NULL)
{
    GSM_Response = 1;
}
else if (strstr(MQTT.inputString, "ERROR") != NULL)
{
    GSM_Response = 2;
}
else if (strstr(MQTT.inputString, ".") != NULL)
{
    GSM_Response = 3;
}
else if (strstr(MQTT.inputString, "CONNECT FAIL") != NULL)
{
    GSM_Response = 5;
}
else if (strstr(MQTT.inputString, "CONNECT") != NULL)
{
    GSM_Response = 4;
    MQTT.TCP_Flag = true;
    cobac.println("MQTT.TCP_Flag = True");
    MQTT.AutoConnect();
    MQTT.pingFlag = true;
    MQTT.tcpATerrorcount = 0;
    //MQTT.MQTT_Flag = true; //INI EDITAN
}
else if (strstr(MQTT.inputString, "CLOSED") != NULL)
{
    GSM_Response = 4;
}

```

```

        MQTT.TCP_Flag = false;
        MQTT.MQTT_Flag = false;
    }
    MQTT.index = 0;
    MQTT.inputString[0] = 0;
}
}
else
{
    uint8_t ReceivedMessageType = (inChar / 16) & 0x0F;
    uint8_t DUP = (inChar & DUP_Mask) / DUP_Mask;
    uint8_t QoS = (inChar & QoS_Mask) / QoS_Scale;
    uint8_t RETAIN = (inChar & RETAIN_Mask);
    if ((ReceivedMessageType >= CONNECT) && (ReceivedMessageType <=
DISCONNECT))
    {
        bool NextLengthByte = true;
        MQTT.length = 0;
        MQTT.lengthLocal = 0;
        uint32_t multiplier=1;
        delay(2);
        char Cchar = inChar;
        while ( (NextLengthByte == true) && (MQTT.TCP_Flag == true))
        {
            if (Serial.available())
            {
                inChar = (char)Serial.read();

                //Serial.println("----");
                //Serial.println(inChar,HEX);
                //Serial.println("-----\r\n");

                if (((Cchar & 0xFF) == 'C') && ((inChar & 0xFF) == 'L')
&& (MQTT.length == 0)) || (((Cchar & 0xFF) == '+') && ((inChar & 0xFF)
== 'P') && (MQTT.length == 0)))
                {
                    MQTT.index = 0;
                    MQTT.inputString[MQTT.index++] = Cchar;
                    MQTT.inputString[MQTT.index++] = inChar;
                    MQTT.TCP_Flag = false;
                    MQTT.MQTT_Flag = false;
                    MQTT.pingFlag = false;
                    cobac.println("Disconnecting");
                }
                else
                {
                    if ((inChar & 128) == 128)
                    {
                        MQTT.length += (inChar & 127) * multiplier;
                        multiplier *= 128;
                        cobac.println("More");
                    }
                    else
                    {
                        NextLengthByte = false;
                        MQTT.length += (inChar & 127) * multiplier;
                        multiplier *= 128;
                    }
                }
            }
        }
    }
}

```

```

        }

MQTT.lengthLocal = MQTT.length;
cobac.println(MQTT.length);
if (MQTT.TCP_Flag == true)
{
    MQTT.printMessageType(ReceivedMessageType);
    MQTT.index = 0L;
    uint32_t a = 0;
    while ((MQTT.length-- > 0) && (Serial.available()))
    {
        MQTT.inputString[uint32_t(MQTT.index++)] =
(char)Serial.read();

        delay(1);

    }
    cobac.println(" ");
    cobac.print("recvdmstype : ");
    cobac.println((char)ReceivedMessageType);
    if (ReceivedMessageType == CONNACK)
    {
        cobac.println("DAPET CONNACK");
        MQTT.ConnectionAcknowledgement = MQTT.inputString[0] * 256
+ MQTT.inputString[1];
        if (MQTT.ConnectionAcknowledgement == 0)
        {
            MQTT.MQTT_Flag = true;
            MQTT.OnConnect();
        }

        MQTT.printConnectAck(MQTT.ConnectionAcknowledgement);
        // MQTT.OnConnect();
    }
    else if (ReceivedMessageType == PUBLISH)
    {
        uint32_t TopicLength = (MQTT.inputString[0]) * 256 +
(MQTT.inputString[1]);
        cobac.print("Topic : ''");
        MQTT.PublishIndex = 0;
        for (uint32_t iter = 2; iter < TopicLength + 2; iter++)
        {
            cobac.print(MQTT.inputString[iter]);
            MQTT.Topic[MQTT.PublishIndex++] =
MQTT.inputString[iter];
        }
        MQTT.Topic[MQTT.PublishIndex] = 0;
        cobac.print("' Message :''");
        MQTT.TopicLength = MQTT.PublishIndex;

        MQTT.PublishIndex = 0;
        uint32_t MessageSTART = TopicLength + 2UL;
        int MessageID = 0;
        if (QoS != 0)
        {
            MessageSTART += 2;
            MessageID = MQTT.inputString[TopicLength + 2UL] * 256 +
MQTT.inputString[TopicLength + 3UL];
        }
        for (uint32_t iter = (MessageSTART); iter <
(MQTT.lengthLocal); iter++)
    }
}

```

```

    {
        cobac.print(MQTT.inputString[iter]);
        MQTT.Message[MQTT.PublishIndex++] =
MQTT.inputString[iter];
    }
    MQTT.Message[MQTT.PublishIndex] = 0;
    cobac.println("''");
    MQTT.MessageLength = MQTT.PublishIndex;
    if (QoS == 1)
    {
        MQTT.publishACK(MessageID);
    }
    else if (QoS == 2)
    {
        MQTT.publishREC(MessageID);
    }
    MQTT.OnMessage(MQTT.Topic, MQTT.TopicLength, MQTT.Message,
MQTT.MessageLength);
    MQTT.MessageFlag = true;
}
else if (ReceivedMessageType == PUBREC)
{
    cobac.print("Message ID :");
    MQTT.publishREL(0, MQTT.inputString[0] * 256 +
MQTT.inputString[1]);
    cobac.println(MQTT.inputString[0] * 256 +
MQTT.inputString[1]);
}

else if (ReceivedMessageType == PUBREL)
{
    cobac.print("Message ID :");
    MQTT.publishCOMP(MQTT.inputString[0] * 256 +
MQTT.inputString[1]);
    cobac.println(MQTT.inputString[0] * 256 +
MQTT.inputString[1]);
}

else if ((ReceivedMessageType == PUBACK) ||
(ReceivedMessageType == PUBCOMP) || (ReceivedMessageType == SUBACK) ||
(ReceivedMessageType == UNSUBACK))
{
    cobac.print("Message ID :");
    cobac.println(MQTT.inputString[0] * 256 +
MQTT.inputString[1]);
}

else if (ReceivedMessageType == PINGREQ)
{
    MQTT.TCP_Flag = false;
    MQTT.pingFlag = false;
    cobac.println("Disconnecting");
    MQTT.sendATreply("AT+CIPSHUT\r\n", ".", 4000);
    MQTT.modemStatus = 0;
}
}

else if ((inChar = 13) || (inChar == 10))
{
}

```

```
        else
        {
            cobac.print("Received :Unknown Message Type :");
            cobac.println(inChar);
        }
    }
}
```