TESIS – TE185401

EVALUASI KINERJA PROTOKOL AIS (AUTOMATIC IDENTIFICATION SYSTEM) DAN MAVLINK PADA JARINGAN AD HOC UNTUK KAPAL NELAYAN

ILA NURMAWATI
07111750032001

DOSEN PEMBIMBING
Dr. Ir. Achmad Affandi, DEA
Dr. Istas Pratomo, ST.MT.

PROGRAM MAGISTER
BIDANG KEAHLIAN TELEKOMUNIKASI MULTIMEDIA
DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI ELEKTRO DAN INFORMATIKA CERDAS
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2020

TESIS – TE185401

EVALUASI KINERJA PROTOKOL AIS (AUTOMATIC IDENTIFICATION SYSTEM) DAN MAVLINK PADA JARINGAN AD HOC UNTUK KAPAL NELAYAN

ILA NURMAWATI
07111750032001

DOSEN PEMBIMBING
Dr. Ir. Achmad Affandi, DEA
Dr. Istas Pratomo, ST. MT.

PROGRAM MAGISTER
BIDANG KEAHLIAN TELEKOMUNIKASI MULTIMEDIA
DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI ELEKTRO DAN INFORMATIKA CERDAS
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2020

LEMBAR PENGESAHAN TESIS

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Teknik (MT)
di
Institut Teknologi Sepuluh Nopember
Oleh:

ILA NURMAWATI
NRP: 07111750032001

Tanggal Ujian: 9 Januari 2020
Periode Wisuda: Maret 2020

Disetujui oleh:
Pembimbing:

1. Dr. Ir. Achmad Affandi, DEA
   NIP: 196510141990021001 .....

2. Dr. Istas Pratomo, ST. MT.
   NIP: 197903252003121001 .....

Penguji:

1. Prof. Dr. Ir. Gamantyo Hendrantoro, M.Eng,. Ph.D
   NIP: 197011111993031002 .....

2. Eko Setijadi, S.T., M.T., Ph.D
   NIP: 197210012003121000 .....

3. Dr. Ir. Wirawan, DEA
   NIP: 196311091989031011 .....

Kepala Departemen Teknik Elektro
Fakultas Teknologi Elektro dan Informatika Cerdas

Dodet Candra R., ST., M.Eng., Ph.D
NIP: 197311192000031001

Halaman ini sengaja dikosongkan

PERNYATAAN KEASLIAN TESIS

Dengan ini saya menyatakan bahwa isi keseluruhan Tesis saya dengan judul "**EVALUASI KINERJA PROTOKOL AIS (AUTOMATIC IDENTIFICATION SYSTEM) DAN MAVLINK PADA JARINGAN AD HOC UNTUK KAPAL NELAYAN**" adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, 26 Desember 2019

Ila Nurmawati

NRP. 07111750032001

EVALUASI KINERJA PROTOKOL AIS (AUTOMATIC IDENTIFICATION SYSTEM) DAN MAVLINK PADA JARINGAN AD HOC UNTUK KAPAL NELAYAN

Nama mahasiswa : Ila Nurmawati
NRP : 07111750032001
Pembimbing : 1. Dr.Ir Achmad Affandi, DEA
2. Dr. Istas Pratomo, ST. MT.

ABSTRAK

Indonesia merupakan negara yang memiliki wilayah laut yang sangat luas sehingga disebut sebagai negara maritime. Untuk mendukung kegiatan nelayan dalam menangkap ikan, pada kapal terpasang sebuah perangkat komunikasi untuk mendapatkan informasi prakiraan cuaca, tinggi gelombang laut dan lain-lain. Teknologi VMS (*Vessel* Monitor*ing System*) dan AIS (*Automatic Identication System*) merupakan sarana komunikasi canggih yang dapat diterapkan pada kapal. VMeS (*Vessel Messaging System*) merupakan komunikasi berbasis radio untuk mengirimkan pesan atau *messages* antara VMeS terminal pada kapal di laut dengan VmeS *gateway* di darat (*ground station*). VMeS terminal pada kapal bersifat *mobile* atau bergerak dan terpasang perangkat GPS (*Global Positioning System*) untuk mengetahui posisi kapal. Pada teknologi VMeS terminal kapal terdiri dari perangkat sensor, gps, modem dan radio *transceiver* , pada sisi *gateway* terdiri dari perangkat modem, radio *transceiver* dan PC. VmeS terhubung melalui saluran kamunikasi radio pada kanal VHF dan dapat berkomunikasi secara dua arah. VMeS dirancang untuk dapat mengirimkan data informasi dari kapal ke *gateway*, informasi mengandung data lokasi kapal, kecepatan kapal, data inersia kapal, data *heading* kapal, informasi muatan kapal dan pesan-pesan lain seperti informasi kecelakaan, kebakaran dan lain-lain. Penelitian yang dilakukan penulis membuat sebuah *prototipe* VmeS berbasis mikrokontroler Arduino dan memanfaatkan saluran komunikasi pada frekuensi 2.4 Ghz. Pada kapal terpasang perangkat VMeS yang berintikan mikrokontroler Arduino Uno yang berfungsi untuk memproses semua perangkat sensor, GPS dan *protocol* komunikasi yang digunakan dalam proses pengiriman data melalui *gateway* sebelum ke *ground station.* Pada sistem *ground station* menggunakan perangkat PC maupun laptop yang berfungsi untuk menampilkan data Monitoring dari masing-masing *client* kapal. *Protocol* komunikasi yang digunakan pada penelitian yang dilakukan menggunakan *Protocol* MAVLink dan AIS. Hasil pengujian diperoleh pengukuran *Quality Of Service* (QOS) jaringan yang diuji pada jarak 30 m hingga 100 meter. Kenaikan jarak komunikasi mengakibatkan kenaikan *end to end delay*, penurunan *throughput* dan potensi *packet loss*.

Kata kunci: *VmeS, AIS,* MAVLink, *Ad-Hoc*

EVALUASI KINERJA *PROTOCOL* AIS (AUTOMATIC IDENTIFICATION SYSTEM) DAN MAVLINK PADA JARINGAN AD HOC UNTUK KAPAL NELAYAN

By : Ila nurmawati
Student Identity Number : 07111750032001
Supervisor(s) :1. Dr.Ir Achmad Affandi, DEA
2. Dr. Istas Pratomo, ST. MT.

ABSTRACT

Indonesia is a maritime country that has wide sea. Communication device is installed in ships to acquire weather information, wave high sea and etc, which is able to support the fisherman's activities and fishing. VMS technology (*Vessel Monitoring System*) and AIS (*Automatic Identification System*) are sophisticated communication that can be applied on ships. VMeS *(Vessel Messaging System)* is a radio-based communication for sending messages between VMeS terminals on ships in sea with VMeS gateways on land *(ground stations).* VMeS terminals on ships is mobile or moving and installed with GPS *(Global Positioning System)* devices to locate ships position. The technology of VMeS of ships terminal consist of sensor device, gps, modem and radio transceiver, the gateway side consists of modem device, radio *transceiver* and PC. VMeS is connected through radio communication channel on the VHF channel and can have two ways communication. VMeS is designed to send information data from ships to *gateway,* the information include ships location data, ships speed, inersia ships data, data ships *heading,* ships loading information and other messages like accident information, fire and etc. This research was conducted by writer to create VMeS prototype microcontroller arduino based and utilize communication channel at frequency 2.4 Ghz. An installed VMeS on ship is the core of Microcontroller Arduino Uno in function to process all sensor devices, GPS and communication protocol that used in sending data process through *gateway* before *ground station. Ground station* system used PC device as well as laptop as function to display the monitoring data from each ships *client.* Communication protocol on this research

used Mavlink and AIS protocol. The test results from measuring Quality Of Service (QOS) network examined at a distance of 30 m to 100 meters. The increase of communication distance inflicted the escalation of end to *end delay*, decreased *throughput* and potential *packet loss.*

Key words: *VmeS, AIS,* MAVLink, *Ad-Hoc*

KATA PENGANTAR

Alhamdulillah dengan nama allah yang maha pengasih lagi maha penyayang. Segala puja dan puji syukur kepada allah swt atas segala rahmat dan karunia yang telah dilimpahkan, sehingga penulisan tesis dengan judul evaluasi kinerja protokol ais (Automatic Identification System) dan MAVLink pada jaringan ad hoc untuk kapal nelayan dapat diselesaikan dengan baik.

Buku tesis ini disusun untuk memenuhi salah satu syarat memperoleh gelar Magister pada Program Studi Teknik Elektro, Bidang Keahlian Telekomunikasi Multimedia, Institut Teknologi Sepuluh Nopember.

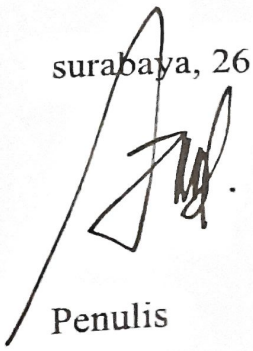
Pada kesempatan ini penulis sampaikan terima kasih yang sedalamdalamnya kepada :

1. Kedua orang tuaku Ayahanda Saridin dan Ibunda sayati tercinta yang telah mendidik penulis dari kecil hingga dewasa.
2. Irdra Bayunanda Yanottama yang selalu memberikan dorongan semangat dalam mengerjakan dan menyelesaikan tesis ini.
3. Bapak Dr. Ir. Achmad Affandi, DEA dan Dr. Istas Pratomo, ST. MT, atas bimbingan, kesabaran dan pendorong semangat dalam menyelesaikan thesis ini.
4. Bapak Prof. Dr. Ir. Gamantyo Hendrantoro, M.Eng,. Ph.D, selaku dosen wali yang telah membimbing selama perkuliahan.
5. Bapak Djoko Suprajitno Rahardjo atas bimbingan, dan motivasi dalam menyelesaikan thesis ini.
6. Windi puspitasari terima kasih atas kebaikan dan kerjasamanya dalam penelitian ini
7. Rekan-rekan S2 dan S1 di lab Jaringan Telekomunikasi B301 terima kasih atas kebaikan dan kerjasamanya dalam penelitian ini.

Penulis menyadari bahwa dalam penulisan tesis ini masih jauh dari sempurna, untuk perbaikan dan penyempurnaan tesis, maka kritik dan saran sangat diharapkan. Besar harapan penulis bahwa buku tesis ini dapat memberikan

informasi dan manfaat bagi pembaca pada umumnya dan mahasiswa Jurusan Teknik Elektro pada khususnya.

surabaya, 26 Desember 2019



Penulis

Halaman ini sengaja dikosongkan

DAFTAR ISI

Halaman ini sengaja dikosongkan

DAFTAR GAMBAR

Halaman ini sengaja dikosongkan

DAFTAR TABEL

Halaman ini sengaja dikosongkan

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Indonesia merupakan negara yang memiliki wilayah laut yang sangat luas sehingga disebut sebagai negara maritim. Luasnya wilayah laut Indonesia memberikan peluang bagi penduduk Indonesia untuk mencari nafkah sebagai nelayan penangkap ikan, sehingga banyak ditemui banyaknya kapal-kapal penangkap ikan di perairan laut Indonesia. Kapal-kapal nelayan penangkap ikan secara umum merupakan kapal berukuran kecil dengan bobot kurang dari 30 GT (*Gross Ton*).

Untuk mendukung kegiatan nelayan dalam menangkap ikan, pada kapal terpasang sebuah perangkat komunikasi untuk mendapatkan informasi prakiraan cuaca, tinggi gelombang laut dan lain-lain. Teknologi yang terpasang bisa dikatakan tertinggal dibanding dengan kapal berukuran lebih besar, yaitu kapal-kapal besar dengan ukuran hingga 300 GT (*Gross Ton*) yang telah menerapkan teknologi satelit. Teknologi VMS (*Vessel Monitoring System*) dan AIS (*Automatic Identication System*) merupakan sarana komunikasi canggih yang dapat diterapkan pada kapal dengan biaya besar. Alternatif lain dapat menggunakan teknologi VMeS (*Vessel Messaging System*) dengan biaya yang jauh lebih murah, sehingga dapat diterapkan pada kapal nelayan berukuran kecil.

VMeS (*Vessel Messaging System*) merupakan komunikasi berbasis radio untuk mengirimkan pesan atau *messages* antara VMeS terminal pada kapal di laut dengan VmeS *gateway* di darat (*ground station*). VMeS terminal pada kapal bersifat *mobile* atau bergerak dan terpasang perangkat GPS (*Global Positioning System*) untuk mengetahui posisi kapal. Pada teknologi VMeS terminal kapal terdiri dari perangkat sensor, gps, modem dan radio *transceiver* , pada sisi *gateway* terdiri dari perangkat modem, radio *transceiver* dan PC. VmeS terhubung melalui saluran kamunikasi radio pada kanal VHF dan dapat berkomunikasi secara dua arah.

VMeS dirancang untuk dapat mengirimkan data informasi dari kapal ke *gateway*, informasi mengandung data lokasi kapal, kecepatan kapal, data inersia kapal, data *heading* kapal, informasi muatan kapal dan pesan-pesan lain seperti informasi kecelakaan, kebakaran dan lain-lain. Informasi ini diperlukan untuk menghindari kapal dengan kapal lainnya terjadi tabrakan, kejadian kecelakaan memungkinkan kapal terdekat untuk memberikan bantuan. Sehingga semua kejadian-kejadian yang tidak diinginkan pada kapal dapat direspon dengan cepat sembari menunggu tindakan dari pihak yang berwenang.

Pada penelitian sebelumnya terminal VMeS menggunakan sebuah SBC (*Single Board Computer*) yang merupakan sebuah komputer mini yaitu Raspberry Pi. Pada penelitian sebelumnya tidak membahas bagaimana *protocol* komunikasi yang diberlakukan, sehingga pada penelitian ini penulis mengangkat tema yang sama dengan menerapkan dan membandingkan beberapa *protocol* komunikasi, yaitu : *protocol* AIS dan *protocol* MAVLink. Penelitian ini akan membuat sebuah *prototype* VmeS berbasis mikrokontroler Arduino dan memafaatkan saluran komunikasi kanal VHF pada frekeuensi 433Mhz.

1.2 Rumusan Masalah

Perumusan masalah dari penelitian yang akan dilakukan adalah sebagai berikut:

a. Bagaimana perancangan terminal komunikasi data VmeS menggunakan ArduinoUno?
b. Bagaimana penerapan *protocol* AIS pada komunikasi data VmeS?
c. Bagaimana penerapan *protocol* MAVLink pada komunikasi data VmeS?
d. Bagaimana perbandingan performa antara *protocol* MAVLink dan AIS pada komunikasi data VmeS?

1.3 Tujuan

Tujuan dari penelitian ini adalah sebagai berikut:

a. Menghasilkan *prototype* terminal komunikasi data VMeS (*Vessel Messaging System*) menggunakan Arduino Uno.

b. Menerapkan *protocol* AIS pada komunikasi data VmeS.
c. Menerapkan *protocol* MAVLink pada komunikasi data VmeS.
d. Membandingan performa antara *protocol* MAVLink dan AIS pada komunikasi data VmeS?

1.4 Batasan Masalah

Batasan masalah dari penelitian terminal komunikasi data VmeS (*Vessel Messaging System*) adalah sebagai berikut:

a. Pada penelitian ini perancangan terminal menggunakan Arduino Uno dan modem NRF24.
b. Menggunakan kanal frekeuensi VHF 433Mhz.
c. Merupakan sebuah *prototype* .
d. Menggunakan *protocol* komunikasi AIS dan *protocol* MAVLink
e. Pengujian akan dibahas pada sisi terminal dan *gateway*.
f. Melakukan pengukuran *throughput*, *packet loss* dan *delay*.

1.5 Kontribusi

Kontribusi yang diharapkan dari hasil penelitian ini dapat memberikan kontribusi keilmiahan dapat memberikan pertimbangan *protocol* komunikasi yang terbaik pada *prototype* terminal komunikasi data VMeS (*Vessel Messaging System*), dan khususnya yang menggunakan *prototype* ini adalah sebagai pertimbangan dan perbandingan dengan *prototype* yang sudah ada.

BAB 2
KAJIAN PUSTAKA

2.1 Perangkat Radio Maritim

2.1.1 Definisi

Perangkat Radio Maritim Non GMDSS (*Global Maritime Distress and Safety System*) adalah perangkat komunikasi yang bekerja pada pita frekuensi maritim yang berfungsi untuk telekomunikasi radio teleponi.

Perangkat Radio Maritim GMDSS adalah perangkat radio yang memenuhi ketentuan sistem komunikasi global dalam dunia pelayaran (maritim) yang berlaku diseluruh dunia baik menggunakan jaringan teristerial (radio) maupun satelit, yang memungkinkan kapal dalam keadaan marabahaya (kecelakaan, tenggelam, dsb) dapat mengirimkan pesan peringatan/marabahaya *(distress)* dalam bentuk teleponi atau DSC (*digital selective calling*) melalui berbagai sistem komunikasi radio secara otomatis ke seluruh kapal dan *Base/Coastal Station* yang berada dekat dengan tempat kejadian atau ke otoritas kelautan yang berwenang [14].

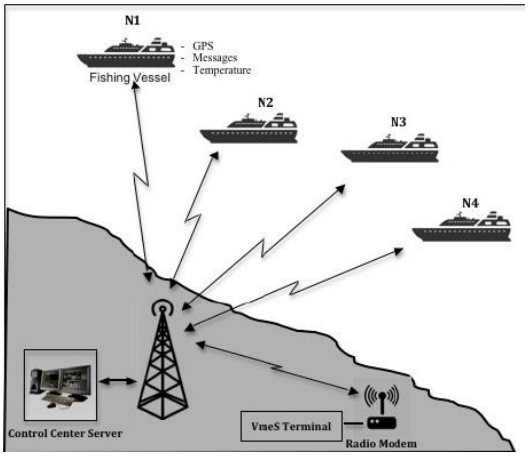
GMDSS adalah suatu paket keselamatan yang disetujui secara internasional yang terdiri dari prosedur keselamatan, jenis-jenis peralatan, *protocol-protocol* komunikasi yang dipakai untuk meningkatkan keselamatan dam mempermudah saat menyelamatkan kapal, perahu, ataupun pesawat terbang yang mengalami kecelakaan.

GMDSS terdiri dari beberapa sistem, beberapa di antaranya baru tetapi kebanyakan peralatan tersebut telah diterapkan selama bertahun-tahun. System tersebut berfungsi untuk : bersiap-siaga (termasuk memantau posisi dari unit yang mengalami kecelakaan), mengkoordinasikan *Search and Rescue*, mencari lokasi (mengevakuasi korban untuk kembali ke daratan), menyiarkan informasi maritim mengenai keselamatan, komunikasi umum, dan komunikasi antar kapal. Radio komunikasi yang spesifik diperlukan sesuai dengan daerah operasi kapal, bukan berdasarkan tonase kapal tersbut. System tersebut juga terdiri dari peralatan

pemancar sinyal berulang sebagai tanda bahaya, serta memiliki sumber power darat untuk menjalankan fungsinya.

VMS (*Vessel Monitoring System*) Sistem pemantauan kapal perikanan/ VMS (*Vessel Monitoring System*) adalah sebuah program pengawasan kegiatan perikanan, yang menggunakan peralatan yang terpasang di kapal perikanan memberikan informasi mengenai kegiatan dan posisi kapal [4].

Menurut peraturan menteri kelautan dan perikanan Nomor PER.05/MEN/2007 tentang penyelenggaraan sistem pemantauan kapal perikanan, sistem pemantauan kapal perikanan adalah salah satu bentuk sistem pengawasan di bidang penangkapan dan/atau pengangkutan ikan, yang menggunakan peralatan pemantauan kapal perikanan yang telah ditentukan. Sistem pemantauan kapal perikanan/ VMS (*Vessel Monitoring System*) adalah sebuah program pengawasan kegiatan perikanan, yang menggunakan peralatan yang terpasang di kapal perikanan memberikan informasi mengenai kegiatan dan posisi kapal [5]. Pada Gambar 2.1 di bawah ini merupakan skema jaringan VMS.



Gambar 2.1 Skema jaringan VMS

Berdasarkan peraturan menteri kelautan dan perikanan Nomor PER.05/MEN/2007, penyelenggaraan sistem pemantauan kapal perikanan/ VMS (*Vessel Monitoring System*)bertujuan untuk:

1. Meningkatkan efektivitas pengelolaan sumberdaya ikan melalui pengendalian dan pemantauan terhadap kapal perikanan;

2. Meningkatkan efektivitas pengelolaan usaha perikanan yang dilakukan oleh perusahaan perikanan;
3. Meningkatkan ketaatan kapal perikanan yang melakukan kegiatan penangkapan dan/atau pengangkutan ikan terhadap ketentuan peraturan perundang-undangan yang berlaku; dan
4. Memperoleh data dan informasi tentang kegiatan kapal perikanan dalam rangka pengelolaan sumberdaya ikan secara bertanggung jawab dan berkelanjutan.

Adapun manfaat sistem pemantauan kapal perikanan bagi pemerintah Indonesia adalah [6]:

1. Dapat melindungi ZEEI Indonesia dari kegiatan-kegiatan kapal perikanan, melacak dan mengidentifikasi tindakan-tindakan illegal *fishing*, dan dengan demikian menegakkan hukum Indonesia dan melindungi kepentingan-kepentingan ekonomi;
2. Dapat menunjukkan penyebaran kapal-kapal di wilayah penangkapan ikan dan membantu penegak hukum terkait untuk memeriksa apakah kapal-kapal tersebut sungguh-sungguh beroperasi di areal penangkapan ikan yang telah ditetapkan; dan
3. Memberikan informasi segera mengenai posisi kapal-kapal yang meminta bantuan sehingga dapat terlacak dan bereaksi secara cepat dan efektif dalam situasi-situasi darurat, seperti perampokan, atau kecelakaan-kecelakaan.

Manfaat sistem pemantauan kapal perikanan bagi pengusaha/pemilik kapal adalah [6]:

1. Dapat memanfaatkan informasi dari *Vessel Monitoring System* untuk memantau keberadaan dan perilaku kapal di laut melalui Website; dan
2. Dapat memanfaatkan informasi *Vessel Monitoring System* untuk keadaan darurat (pembajakan, kebakaran, tenggelam dan lain-lain).

Berdasarkan peraturan menteri kelautan dan perikanan Nomor PER.05/MEN/2007, penyelenggaraan sistem pemantauan kapal perikanan/ VMS (*Vessel Monitoring System*) bertujuan untuk:

1. Meningkatkan efektivitas pengelolaan sumberdaya ikan melalui pengendalian dan pemantauan terhadap kapal perikanan.
2. Meningkatkan efektivitas pengelolaan usaha perikanan yang dilakukan oleh perusahaan perikanan.
3. Meningkatkan ketaatan kapal perikanan yang melakukan kegiatan penangkapan dan/atau pengangkutan ikan terhadap ketentuan peraturan perundang-undangan yang berlaku; dan
4. Memperoleh data dan informasi tentang kegiatan kapal perikanan dalam rangka pengelolaan sumberdaya ikan secara bertanggung jawab dan berkelanjutan [5].

Peruntukan Penggunaan Pita Frekuensi Radio 350 – 438 Mhz [22]

Pasal 2

1. Peruntukan penggunaan pita frekuensi radio 350-438 MHz meliputi: sistem komunikasi radio konvensional;
2. keperluan Kewajiban Pelayanan Universal;
3. sistem komunikasi radio *trunking*;
4. dinas komunikasi radio selain dinas tetap dan dinas bergerak;
5. amatir radio;
6. pengoperasian alat dan perangkat telekomunikasi jarak dekat (*Short Range Device*);
7. dinas satelit eksplorasi bumi (aktif); dan
8. dinas radiolokasi.

Pasal 3

1. Penggunaan pita frekuensi radio untuk sistem komunikasi radio konvensional, keperluan Kewajiban Pelayanan Universal, sistem komunikasi radio *trunking*, dan dinas radiolokasi sebagaimana dimaksud dalam Pasal 2 huruf a, huruf b, huruf c, dan huruf h termasuk kategori primer.
2. Penggunaan pita frekuensi radio untuk amatir radio, pengoperasian alat dan perangkat telekomunikasi jarak dekat (*Short Range Device*), dan dinas satelit

eksplorasi bumi (aktif) sebagaimana dimaksud dalam Pasal 2 huruf e, huruf f, dan huruf g termasuk kategori sekunder.

3. Pengkategorian primer atau sekunder pada penggunaan pita frekuensi radio untuk dinas komunikasi radio selain dinas tetap dan dinas bergerak sebagaimana dimaksud dalam Pasal 2 huruf d diatur di dalam Tabel Alokasi Spektrum Frekuensi Radio Indonesia.
4. Penggunaan pita frekuensi radio yang termasuk kategori sekunder sebagaimana dimaksud pada ayat (2) dan ayat (3) harus memenuhi ketentuan:
   (1) tidak boleh menyebabkan gangguan yang merugikan (*harmful interference*) terhadap penggunaan pita frekuensi radio yang termasuk kategori primer; dan
   (2) tidak mendapatkan perlindungan terhadap adanya gangguan yang merugikan (*harmful interference*) yang

Teknologi VMS memiliki fungsi dalam menyediakan informasi umum dalam laporan data VMS seperti *unit identifier* (ID *transmitter* kapal), tanggal dan waktu, serta garis lintang dan bujur. Penyediaan informasi mengenai posisi kapal menggunakan sistem GPS. GPS yang telah terintegrasi dengan unit dapat menentukan posisi secara langsung termasuk laporan posisinya, atau sistem satelit yang dapat menentukan posisi dengan mengukur pergeseran sinyal *Doppler* yang dikirim dari unit di atas kapal (perubahan frekuensi dari gelombang ketika *emitted electromagnetic* penerima yang berada dalam gerakan relatif terhadap satu sama lain) [6].

Peralatan kapal (*transmitter*) yang mengirimkan laporan posisi dan informasi dalam beberapa cara. Sistem satu arah secara otomatis mengirimkan laporan dalam *pra-interval* yang telah ditetapkan, dan dapat juga mengirimkan informasi tambahan. Sistem dua arah, laporan juga dikirim secara otomatis dalam *pra-interval* yang ditetapkan. pusat pemantauan perikanan memungkinkan untuk meminta informasi dari kapal, termasuk laporan posisi kapal yang terbaru atau status peralatan, dan juga mengubah interval pelaporan. Arah pergerakkan dan kecepatan kapal dapat dihitung secara langsung dan dikirim bersama-sama dengan laporan posisi kapal, atau dapat juga dihitung dengan *Software* di pusat pemantauan perikanan, yang berdasarkan waktu dan jarak antara posisi laporan. Jenis

transmitter yang paling banyak digunakan dalam program ini termasuk VMS Argos *transmitters*, Inmarsat-C dan Inmarsat-D+ *transceiver s*, Qualcomm unit (EutelTRACS dan Boatracs), dan Orbcomm sistem. Kebanyakan dari peralatan ini merupakan integrasi dari GPS untuk mendapatkan posisi [6].

Sistem komunikasi membawa laporan posisi dan pesan lainnya dari peralatan yang berada di atas kapal, melalui ruang angkasa dan jalur darat, menuju pusat pemantauan perikanan. *Provider* yang menggunakan segmen ruang angkasa dalam program VMS di bidang perikanan adalah Argos, dan Inmarsat-C dan Inmarsat D+. Sistem Argos (CLS) memiliki orbit satelit di daerah kutub dan dioperasikan oleh *National Oceanic and Atmospheric Administration USA*. Orbital satelit di daerah kutub memberikan cakupan pengawasan yang baik pada daerah lintang tinggi, dan prosesnya satu arah, dari kapal langsung ke pantai. Inmarsat-C dan Inmarsat D+ menggunakan satelit *geostationary* sepanjang khatulistiwa, memberikan wilayah cakupan pengawasan hampir global dua arah. Karena lokasi satelit di khatulistiwa, maka tidak dapat melakukan cakupan pada wilayah lintang tinggi. Inmarsat menawarkan beberapa jenis layanan komunikasi, tetapi Inmarsat-C dan Inmarsat D+ adalah yang paling sesuai untuk aplikasi VMS karena biaya-efektif untuk pesan *teks* dan paket data [6].

Pusat pemantauan perikanan/ FMC (*Fisheries* Monitor*ing Centre*) adalah sebuah pusat yang memantau dan menerima laporan yang dikirimkan melalui *transmitter* dan kemudian menyimpannya ke dalam *database* dari semua kegiatan kapal penangkap ikan yang telah menggunakan sistem VMS. Pengawas di FMC mengawasi seluruh kegiatan penangkapan dari Monitor dan dianalisis jika terjadi indikasi pelanggaran untuk segera diambil tindakan. FMC merupakan lokasi yang aman dimana hanya personil atau petugas pengawasan yang berwenang yang dapat mengakses data VMS [4].

2.2 Sensor Suhu & Kelembaban (SHT11)

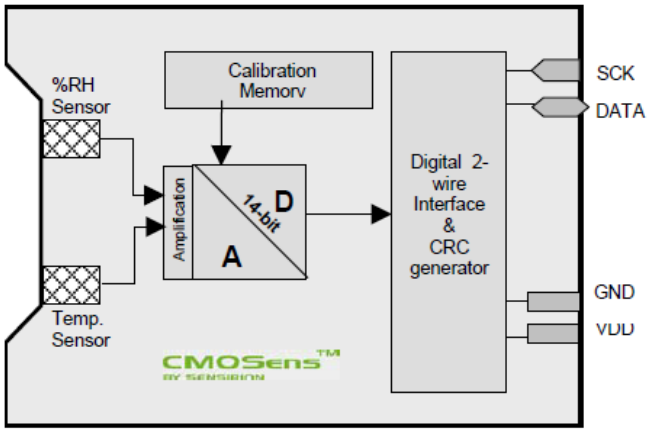
SHT11 adalah sebuah single chip sensor suhu dan kelembaban relatif dengan multi modul sensor yang *output* nya telah dikalibrasikan secara digital. Dibagian dalamnya terdapat kapasitif polimer sebagai elemen untuk sensor

kelembaban relatif dan sebuat pita regangan yang digunakan sebagai sensor temperatur. *Output* kedua sensor digabungkan dan dihubungkan pada ADC 14 bit dan sebuah *interface* serial pada satu chip yang sama. Sensor ini menghasilkan sinyal keluaran yang baik dengan waktu respon yang cepat.

2.2.1 Spesifikasi sensor suhu & kelembaban (SHT11)

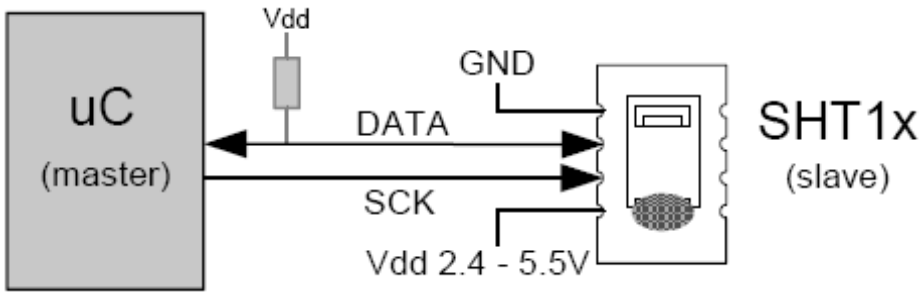
1. Mengukur suhu dari -40°C (-40°F) hingga +123,8°C (+254,8°F) dan kelembaban relatif dari 0% RH hingga 100% RH.
2. Memiliki ketepatan (akurasi) pengukuran suhu hingga ±0,5°C pada suhu 25°C dan ketepatan (akurasi) pengukuran kelembaban relatif hingga ±3,5%RH.
3. Memiliki antarmuka serial s*ynchronous* 2-*wire*, bukan I2C. Jalur antarmuka telah dilengkapi dengan rangkaian pencegah kondisi sensor *lock-up*.
4. Membutuhkan catu daya +5V DC dengan konsumsi daya rendah 30µW. Modul ini memiliki faktor bentuk 8 pin DIP 0,6″ sehingga memudahkan pemasangannya

SHT11 dikalibarasi pada ruangan dengan kelembaban yang teliti menggunakan hygrometer sebagai referensinya. Koefisien kalibrasinya telah diprogramkan kedalam OTP memori. Koefisien tersebut akan digunakan untuk mengkalibrasi keluaran dari sensor selama proses pengukuran. Bidirectional 2-*wire* alat penghubung serial dan regulasi tegangan internal membuat lebih mudah dalam pengintegrasian sistem. Ukurannya yang kecil dan konsumsi daya yang rendah membuat sensor ini adalah pilihan yang tepat, bahkan untuk aplikasi yang paling menuntut. Didalam piranti SHT11 terdapat suatu *surface-mountable* LLC (*Leadless Chip Carrier*) yang berfungsi sebagai suatu *pluggable* 4-pin *single-in-line* untuk jalur data dan clock, blok diagram chip SHT11 dapat dilihat pada gambar 2.2 [12].



Gambar 2.2 Blok diagram pada chip SHT11

Sistem sensor yang digunakan untuk mengukur suhu dan kelembaban adalah SHT11 dengan sumber tegangan 5 Volt dan komunikasi *bidirectonal* 2-*wire*. Sistem sensor ini mempunyai 1 jalur data yang digunakan untuk perintah pengalamatan dan pembacaan data. Pengambilan data untuk masing-masing pengukuran dilakukan dengan memberikan perintah pengalamatan oleh mikrokontroler. Kaki serial Data yang terhubung dengan mikrokontroler memberikan perintah pengalamatan pada pin Data SHT11 “00000101” untuk mengukur kelembaban relatif dan “00000011” untuk pengukuran temperatur. SHT11 memberikan keluaran data kelembaban dan temperatur pada pin Data secara bergantian sesuai dengan clock yang diberikan mikrokontroler agar sensor dapat bekerja. Sensor SHT11 memiliki ADC (*Analog to Digital Converter*) di dalamnya sehingga keluaran data SHT11 sudah terkonversi dalam bentuk data digital dan tidak memerlukan ADC eksternal dalam pengolahan data pada mikrokontroler. Skema pengambilan data SHT11 dapat dilihat pada gambar 2.3 berikut ini. Gambar 2.3 merupakan Cara sensor mengambil data



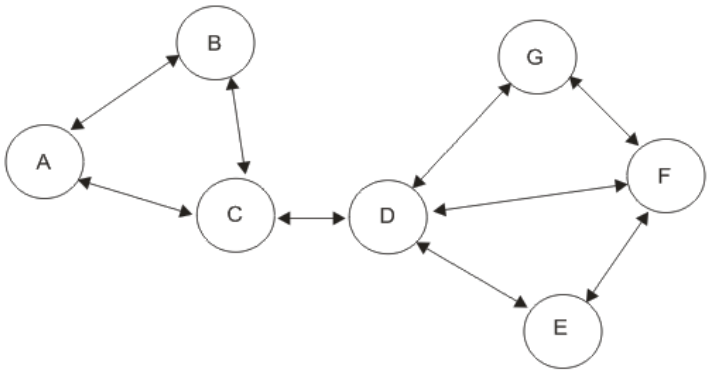
Gambar 2.3 Cara sensor mengambil data

SHT 11 membutuhkan suplai tegangan sebesar 5 VDC. SCK (*serial clock Input*) digunakan untuk menyimgkronkan komunikasi antara mikrokontroller dengan SHT11, data (serial data) digunakan untuk transfer data dari dan ke SHT11[20].

2.3 Jaringan *Wireless* Ad Hoc

Jaringan *Wireless* terdiri dari dua model yaitu *fixed* dan *mobile*.Jaringan Fixed *wireless* tidak mendukung mobility, dan kebanyakan adalah *point to point* (seperti *microwave network dan geostationary satellite network*). Lain halnya dengan jaringan *mobile wireless* yang sangat dibutuhkan oleh pengguna yang bergerak. Jaringan *mobile* dibagi dalam dua kategori utama yaitu jaringan yang memiliki infrastruktur (selular) dan jaringan yang tidak memiliiki infrastruktur. Jaringan yang tak memiliki infrastruktur ini yang yang biasanya disebut dengan jaringan ad hoc.

Jaringan ad hoc untuk suatu tujuan diartikan sebagai suatu jaringan tanpa infrastruktur dimana masing-masing *node* adalah suatu *router* bergerak yang dilengkapi dengan *transceiver wireless* . Pesan yang dikirim dalam lingkungan jaringan ini akan terjadi antara dua *node* dalam cakupan transmisi masing-masing yang secara tidak langsung dihubungkan oleh multiple hop melalui beberapa *node* perantara [3].



Gambar 2.4 Struktur dasar jaringan Ad Hoc

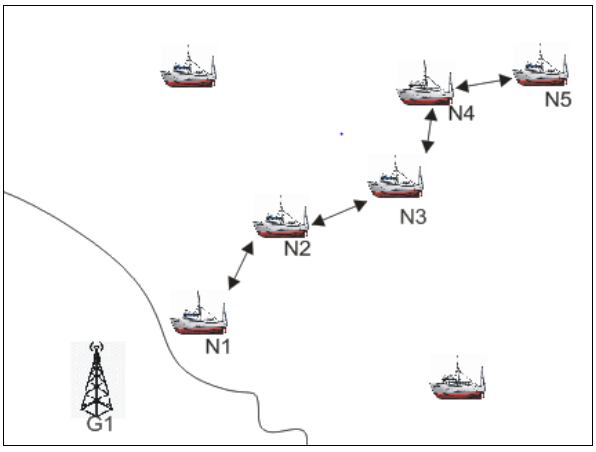
Sebuah *protocol routing* untuk jaringan *wireless* ad hoc sangat diperlukan pada proses komunikasi antara beberapa *node*, untuk mengirimkan paket data melalui satu atau beberapa *node* menuju alamat tujuan dimana topologi jaringan selalu berubah. *Protocol* rute yang dibangun harus dapat mencari rute alternatif untuk mengatasi masalah ketika terjadi rute *error* sehingga *node* tidak memulai proses pencarian rute dari awal. Selain itu, sistem komunikasi kapal laut yang menggunakan kanal VHF memiliki jumlah kanal terbatas dan *bitrate* yang rendah(1200 bps) [9].

Sistem jaringan ad hoc pada umumnya memiliki kemampuan untuk membangun rute secara mandiri. Namun dengan beberapa pertimbangan teknis sistem ad hoc bisa juga dijadikan sistem yang komunikasi datanya terkontrol dari pusat. Untuk membangun *protocol* komunkasi data maka perlu mempertimbangkan beberapa aspek seperti kecepatan pengiriman data (*bitrate*), jumlah trafik yang ditawarkan, ketersediaan kanal komunikasi, jarak antar masing-masing titik *client* yang ada di dalam jaringan, dan proses untuk menangani *multiple acces*.

Dasar dari pengembangan ini karena kecepatan komunikasi data melalui kanal radio terbatas yaitu sebesar 1200 bps atau sekitar 120 karakter per detik sehingga proses '*flooding* paket data' untuk memperbaharui tabel *routing* sebaiknya dihindari. Di atas kapal pada saat ini sudah dirasa perlu untuk memasang GPS dimana GPS selain untuk navigasi pada saat berada di atas kapal juga dapat pula dimanfaatkan untuk memperbarui tabel *routing* dan untuk Monitoring posisi.

Pada komunikasi data kesalahan satu bit dalam satu paket data dapat mengakibatkan dibuangnya seluruh paket data sehingga paket data yang dikirimkan dibuat tidak terlalu panjang supaya bila ada kesalahan data yang dibuang juga tidak banyak. Namun paket yang terlalu pendek juga tidak akan efisien karena satu paket juga memerlukan rangkaian bit sinkronisasi dan overhead yang berupa pengalamatan dan untuk kontrol kesalahan paket.

Pada proses *multiple access* digunakan metode TDMA (*Time Division Multiple Access*) dengan menyisipkan slot-slot waktu tertentu untuk registrasi *client* disela-sela waktu untuk menginterogasi *client -client* yang sudah terdaftar. Fungsi dari registrasi *client* adalah untuk menyisipkan *client* baru pada daftar *pooling* TDMA dan untuk membuang *client* dari tabel *routing* apabila sudah kembali ke daratan. *Client -client* yang sedang berlayar diharapkan dapat selalu terpantau sampai mereka kembali ke daratan. Dalam proses *pooling* ini juga dapat diterapkan metode prediksi posisi dari laporan posisi, arah dan kecepatan terakhir. Serta dapat juga didasarkan pada laporan rencana rute pada saat sebelum melepas jangkar. Yang diilustrasikan pada gambar 2.5[9].



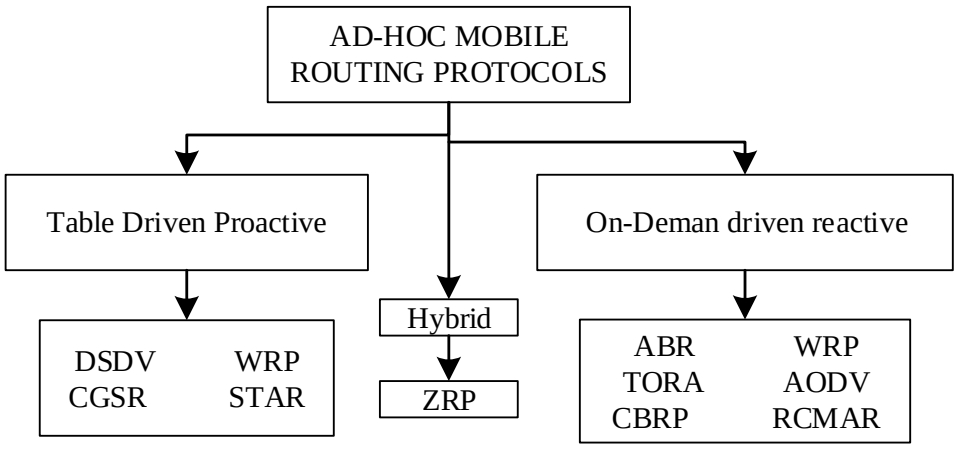
Gambar 2.5 Ilustrasi contoh kasus komunikasi data melalui jaringan Ad Hoc

Pada gambar 2.5 diasumsikan bahwa jarak jangkauan maksimum antar *client* adalah sebagai berikut: G1 hanya bisa berkomunikasi dengan N1, N1 hanya bisa berkomunikasi dengan N2, N2 hanya bisa berkomunikasi dengan N3, begitu pula dengan N3, N4, dan N5. Pada kasus ini G1 untuk menjangkau N2 tidak dapat

dilakukan secara langsung sehingga memanfaatkan N1 sebagai perantara untuk menyampaikan paket data ke N2. Begitu pula untuk mengirimkan paket data ke N5 dari G1 maka diperlukan perantara N1, N2, N3, dan N4 [7].

2.3.1 *Routing*

Routing adalah hal yang sangat penting didalam jaringan ad hoc. Pada Gambar 2.6 digambarkan klasifikasi *protocol routing*. *Protocol routing* proaktif bersifat table driven dimana setiap *node* menyimpan tabel yang berisi informasi rute semua *node* yang diketahuinya. Informasi rute di-*update* secara berkala. *Protocol routing* reaktif adalah on-demand yang berbasis pada sebuah rute yang dibentuk selama permintaan (*request*). *Hybrid routing Protocol* adalah kombinasi darikedua *routing protocol* antara proaktif dan reaktif. Penggunaan *protocol routing* proaktif secara mendasar memberikan solusi terpendek *end-to-end delay*, karena informasi *routing* selalu tersedia dan *up-to-date* jika dibandingkan dengan *protocol routing* reaktif. Kekurangan dari dari *protocol routing* proaktif adalah terlalu banyak penggunaan sumber daya (*resource*), seperti overhead disaat meng-*update* informasi *routing* [10].



Gambar 2.6 Klasifikasi *Protocol Routing* Ad hoc

2.4 *Protocol* MAVLink

MAVLink atau *Micro Air Vehicle Link* adalah *protocol* untuk berkomunikasi dengan kendaraan kecil tanpa awak. *Protocol* ini dirancang sebagai pustaka *header-only message marshaling*. MAVLink pertama kali dirilis awal 2009

oleh Lorenz Meier di bawah lisensi LGPL. *Protocol* ini Ini digunakan sebagian besar untuk komunikasi antara GCS(*Ground Control Station*) dan kendaraan tak berawak, dan dalam komunikasi antar subsistem kendaraan. Hal ini dapat digunakan untuk mengirimkan orientasi kendaraan, lokasi dan kecepatan GPS-nya.

MAVLINK adalah *protocol* layer 2 (merujuk pada OSI *Layer Reference*) yaitu data *Link Layer*. Sebagai *protocol* layer 2 MAVLINK bertanggungjawab untuk menbangun *link connection*, menyediakan prosedur logic untuk *information transfer*, dan *link disconnection*. Sehingga MAVLINK cukup lengkap untuk dijadikan contoh implementasi sebuah *protocol*. Untuk membangun hubungan antara dua buah terminal melalui *physical layer* dan lapisan *data link*. *Protocol* ini akan bekerja pada dua kondisi transmisi yaitu *half duplex* dan *full duplex*. Selanjutnya dua lapisan yang ada pada *protocol* ini yaitu *physical layer* dan lapisan *data link* dapat dibagi lagi ke dalam beberapa status keadaan. Keadaan yang dimasudkan adalah mendefinisikan keadaan suatu *link* komunikasi radio untuk *multi link* [11].

2.4.1 Struktur Paket

Ini digunakan sebagian besar untuk komunikasi antara GCS (*Ground Control Station*) dan kendaraan tak berawak, dan dalam komunikasi antar subsistem kendaraan. Hal ini dapat digunakan untuk mengirimkan orientasi kendaraan, lokasi dan kecepatan GPS-nya. Dalam versi 1.0 struktur paket dapat dilihat pada tabel 2.1 adalah sebagai berikut:

Tabel 2.1Struktur Paket MAVLink

Nama bidang	Indeks (*Byte*)	Tujuan
Mulai dari bingkai	0	Menunjukkan awal transmisi frame (v1.0: 0xFE)
Panjang *payload*	1	panjang *payload* (n)
Urutan paket	2	Setiap komponen menghitung urutan pengirimannya. Mengizinkan deteksi paket yang hilang.

Tabel lanjutan		
ID sistem	3	Identifikasi sistem pengiriman. Memungkinkan untuk membedakan sistem yang berbeda pada jaringan yang sama.
ID komponen	4	Identifikasi komponen pengiriman. Memungkinkan untuk membedakan berbagai komponen dari sistem yang sama, misalnya IMU dan autopilot.
ID pesan	5	Identifikasi pesan - id menentukan apa arti "muatan" *payload* dan bagaimana seharusnya diterjemahkan dengan benar.
Muatan	6 hingga (n + 6)	Data ke dalam pesan, tergantung pada id pesan.
CRC	(n + 7) hingga (n + 8)	Periksa-jumlah seluruh paket, tidak termasuk tanda mulai paket (LSB ke MSB)

2.4.2 Bidang CRC

Untuk memastikan integritas pesan, pemeriksaan CRC (redundansi siklik) dihitung untuk setiap pesan menjadi dua *bytes* terakhir. Fungsi lain dari bidang CRC adalah untuk memastikan pengirim dan penerima sepakat dalam pesan yang sedang ditransfer. Ini dihitung menggunakan hash ITU X.25 / SAE AS-4 dari *bytes* dalam paket, tidak termasuk indikator *Start-of-Frame* (jadi 6 + n + 1 *bytes* dievaluasi, ekstra +1 adalah nilai *seed*) .

Selain itu nilai *seed* ditambahkan ke bagian akhir data saat menghitung CRC. *Seed* dihasilkan dengan setiap set pesan baru dari *protocol*, dan di-*hash* dengan cara yang sama seperti paket-paket dari setiap spesifikasi pesan. Sistem yang menggunakan *protocol* MAVLink dapat menggunakan *array* yang sudah dikomputasi untuk tujuan ini [19].

2.5 AIS messages (*Automatic Identication System*)

AIS (*Automatic Identification System*) adalah sebuah *Transceiver* yang bekerja pada frekuensi maritim 161,975 MHz dan 162,025 MHz sesuai regulasi IMO (*Internasional Maritime Organization*). Perangkat ini secara otomatis mengirimkan AIS Message kesemua arah, Message yang dikirimkan antara lain berisi MMSI (*Mobile Maritime System Identification*) atau ID Kapal, Kecepatan Kapal, Posisi Kapal, Arah Kemudi Kapal, dan seterusnya sehingga kapal lain di sekitar kapal tersebut yang sudah dilengkapi dengan perangkat AIS *Transceiver* dapat mengetahui secara terus menerus situasi lalulintas disekelilingnya yang ditampilkan pada layar ECDIS (*display Monitor Electronic Chart Display Information System*) / SENC (System Electronic Navigation Chart) atau ENC (*Electronic Navigation Chart*).

Dengan terpasangnya AIS *Transceiver* di kapal maka Perangkat Monitoring Pelayaran di Darat / VTS (*Vessel Tracking System*) dapat meMonitor situasi lalu lintas yang berada di area pengamatan mereka dengan baik dan dapat memberikan arahan atau petunjuk jika terjadi situasi yang berbahaya setiap saat.
Jika kapal sudah keluar dari jangkauan AIS *Base Station* maka selanjutnya AIS Message yang dikirimkan oleh AIS *Transceiver* tersebut dapat diterima oleh Perangkat AIS *Receiver Satellite* yang kemudian mengirimkan AIS *Message* tersebut ke VTS sehingga posisi kapal tersebut dapat selalu dimonitor pada VTS [20].

Transponder AIS secara otomatis menyiarkan informasi, seperti posisi, kecepatan, dan status navigasi kapal, secara berkala melalui pemancar VHF yang terpasang pada transponder. Informasi tersebut berasal dari sensor navigasi kapal, biasanya penerima dan GNSS (*gyrocompass* sistem navigasi satelit global). Informasi lain, seperti nama kapal dan tanda panggilan VHF, diprogram saat memasang peralatan dan juga dikirimkan secara teratur. Sinyal diterima oleh transponder AIS yang dipasang di kapal lain atau pada sistem berbasis darat, seperti sistem VTS. Informasi yang diterima dapat ditampilkan pada layar atau *plotter* bagan, yang menunjukkan posisi kapal lain dengan cara yang sama seperti tampilan radar. Standar AIS terdiri dari beberapa 'Standar' sub-standar yang menentukan

jenis produk secara individual. Spesifikasi untuk setiap jenis produk memberikan spesifikasi teknis terperinci yang memastikan integritas keseluruhan sistem AIS global di mana semua jenis produk beroperasi. Standar sistem AIS adalah :

1. Kelas A, *Transceiver* AIS yang dipasang di kapal (kirim dan terima) yang beroperasi menggunakan SOTDMA. Ditargetkan pada kapal komersial besar, SOTDMA membutuhkan *transceiver* untuk mempertahankan peta slot yang terus diperbarui dalam memorinya sehingga memiliki pengetahuan sebelumnya tentang slot yang tersedia untuk ditransmisikan. *Transceiver* SOTDMA kemudian akan mengumumkan pengiriman mereka, secara efektif memesan slot pengiriman mereka. Oleh karena itu transmisi SOTDMA diprioritaskan dalam sistem AIS. Ini dicapai melalui 2 penerima dalam operasi berkelanjutan. Kelas A harus memiliki tampilan terintegrasi, mentransmisikan pada 12,5 W, kemampuan antarmuka dengan beberapa sistem kapal, dan menawarkan pilihan fitur dan fungsi yang canggih. Tingkat pengiriman *default* adalah setiap beberapa detik. Perangkat yang memenuhi standar tipe AIS menerima semua jenis pesan AIS.
2. Kelas B, *Transceiver* AIS yang dipasang di kapal (mentransmisikan dan menerima) yang beroperasi menggunakan akses ganda pembagian waktu (CSTDMA) atau SOTDMA operator-sense; sekarang ada 2 spesifikasi IMO terpisah untuk Kelas B. Ditujukan untuk pasar komersial dan rekreasi yang lebih ringan. *Transceiver* CSTDMA mendengarkan peta slot segera sebelum mentransmisikan dan mencari slot di mana '*noise'* dalam slot adalah sama atau mirip dengan noise latar belakang, dengan demikian menunjukkan bahwa slot tidak sedang digunakan oleh perangkat AIS lain. Kelas B mentransmisikan pada 2 W dan tidak diharuskan memiliki tampilan terintegrasi: Kelas B dapat dihubungkan ke sebagian besar sistem tampilan di mana pesan yang diterima akan ditampilkan dalam daftar atau ditindih pada grafik.Laju pengiriman standar biasanya setiap 30 detik, tetapi ini dapat bervariasi sesuai dengan kecepatan kapal atau instruksi dari stasiun pangkalan. Standar tipe Kelas B memerlukan GPS terintegrasi dan indikator LED tertentu. Peralatan Kelas B menerima semua jenis pesan AIS [18].

2.5.1 Penerima AIS

Sejumlah pabrikan menawarkan penerima AIS, yang dirancang untuk memantau lalu lintas AIS. Ini mungkin memiliki dua penerima, untuk memantau kedua frekuensi secara bersamaan, atau mereka dapat beralih di antara frekuensi (sehingga tidak ada pesan di saluran lain, tetapi dengan harga lebih murah). Secara umum mereka akan menampilkan data RS232 , NMEA , USB atau UDP untuk ditampilkan pada komplotan bagan elektronik atau komputer.

2.5.2 Karakteristik RF

AIS menggunakan saluran Marine Band 87 & 88 yang dialokasikan secara global. AIS menggunakan sisi tinggi dupleks dari dua "saluran" radio VHF (87B) dan (88B)

- Saluran A 161.975 MHz (87B)
- Saluran B 162.025 MHz (88B)

Saluran simpleks 87A dan 88A menggunakan frekuensi yang lebih rendah sehingga mereka tidak terpengaruh oleh alokasi ini dan masih dapat digunakan sebagaimana ditentukan untuk rencana frekuensi seluler maritim.

Sebagian besar transmisi AIS terdiri dari semburan beberapa pesan. Dalam kasus ini, di antara pesan, pemancar AIS harus mengubah saluran.
Sebelum dikirim, pesan AIS harus dikodekan NRZI . Pesan AIS ditransmisikan menggunakan modulasi GMSK . Modulator GMSK produk-BT yang digunakan untuk transmisi data harus 0,4 maksimum (nilai nominal tertinggi).

Data kode GMSK harus memodulasi frekuensi pemancar VHF. Indeks modulasi harus 0,5. Kecepatan bit transmisi adalah 9600bit /s, Penerima VHF biasa dapat menerima AIS dengan penyaringan dinonaktifkan (penyaringan menghancurkan data GMSK). Namun, *output* audio dari radio perlu didekodekan. Ada beberapa aplikasi PC yang dapat melakukan ini [17].

2.5.3 Pesan dikirim dan diterima melalui udara.

Semua pesan AIS mengirimkan 3 elemen dasar informasi:

1. Nomor MMSI kapal atau peralatan yang memegang pemancar (stasiun pangkalan, pelampung, dll.)
2. Identifikasi pesan yang sedang dikirim (Lihat tabel di bawah)
3. Indikator berulang yang dirancang untuk digunakan untuk mengulangi pesan melalui rintangan oleh perangkat relai.

Tabel berikut ini memberikan ringkasan dari semua pesan AIS yang saat ini digunakan [15].

Tabel 2.2 Pesan dikirim dan diterima pada komunikasi AIS

Pesan ais	Pemakaian	Komentar
Pesan 1, 2, 3: Laporan Posisi Kelas A	Melaporkan informasi navigasi	Pesan ini mentransmisikan informasi yang berkaitan dengan navigasi kapal: Bujur dan lintang, waktu, arah, kecepatan, status navigasi kapal
Pesan 4: Laporan Stasiun Base	Digunakan oleh BTS untuk menunjukkan keberadaannya	Pesan melaporkan posisi dan waktu yang tepat. Ini berfungsi sebagai referensi statis untuk kapal lain
Pesan 5: Data Terkait Statis dan Pelayaran	Memberikan informasi tentang kapal dan perjalanannya	Salah satu dari beberapa pesan yang datanya dimasukkan dengan tangan. Informasi ini termasuk data statis seperti panjang, lebar kapal, konsep, serta tujuan kapal yang dituju
Pesan 6: Pesan Biner Ditujukan	Pesan point-to-point yang ditujukan dengan muatan biner yang tidak ditentukan.	

Pesan 7: Pesan Pengakuan Biner	Dikirim untuk mengetahui penerimaan pesan 6	
Pesan 8: Pesan Siaran Biner	Pesan siaran dengan muatan biner yang tidak ditentukan.	
Pesan 9: Laporan Standar Pencarian dan Penyelamatan Posisi Pesawat	Digunakan oleh pesawat terbang (helikopter atau pesawat terbang) yang terlibat dengan operasi pencarian dan penyelamatan di laut (yaitu pencarian dan pemulihan korban yang selamat dari kecelakaan di laut).	Mengirim lokasi (termasuk ketinggian) dan informasi waktu
Pesan 10: Permintaan UTC / Tanggal	Dapatkan waktu dan tanggal dari stasiun pangkalan	Permintaan informasi UTC / Tanggal dari stasiun pangkalan AIS. Digunakan ketika perangkat tidak memiliki waktu dan tanggal secara lokal, biasanya dari GPS
Pesan 11: Respons waktu / tanggal universal yang	Tanggapan dari pesan 10	Identik dengan pesan 4.

terkoordinasi terkoordinasi		
Pesan 12: Pesan Terkait Keamanan	Digunakan untuk mengirim pesan teks ke kapal yang ditentukan	Pesan teks mungkin dalam bahasa Inggris, kode komersial atau bahkan dienkripsi
Pesan 13: Pengakuan yang terkait dengan keselamatan	Tanggapan dari pesan 12	
Pesan 14: Pesan siaran terkait keamanan	Identik dengan pesan 12, tetapi disiarkan	
Pesan 15: Interogasi	Digunakan oleh stasiun pangkalan untuk mendapatkan status hingga 2 perangkat AIS lainnya	
Pesan 16: Perintah mode yang ditetapkan	Digunakan oleh stasiun pangkalan untuk mengelola slot AIS	
Pesan 17: Pesan biner sistem navigasi	Digunakan oleh *Base Station* untuk menyiarkan koreksi diferensial untuk GPS	

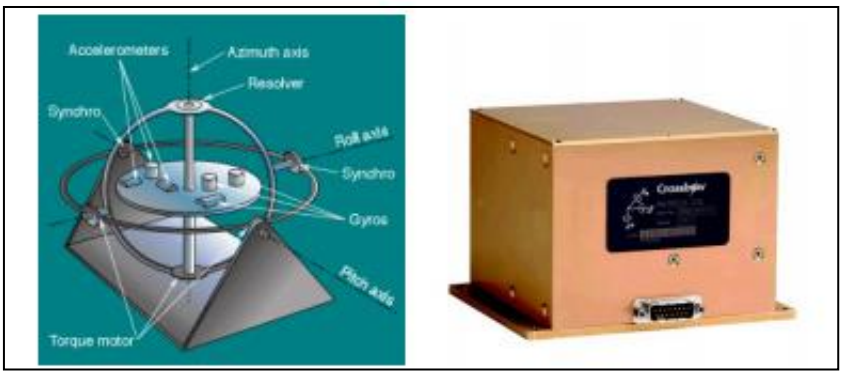
global-satelit menyiarkan		
Pesan 18: Laporan posisi peralatan kelas B standar	Laporan yang kurang rinci dibandingkan tipe 1-3 untuk kapal yang menggunakan pemancar Kelas B	Tidak termasuk status navigasi atau laju belokan
Pesan 19: Laporan posisi peralatan kelas B yang diperluas	Untuk peralatan kelas B legacy	Digantikan oleh pesan 18
Pesan 20: Pesan manajemen tautan data	Digunakan oleh stasiun pangkalan untuk mengelola slot AIS	Pesan ini digunakan untuk pra-alokasikan slot TDMA dalam jaringan stasiun pangkalan AIS
Pesan 21: Laporan bantuan-ke-navigasi	Digunakan oleh bantuan (AtN) untuk perangkat navigasi (pelampung, mercusuar ..)	Mengirimkan waktu dan lokasi yang tepat serta karakteristik AtN
Pesan 22: Manajemen saluran	Digunakan oleh stasiun pangkalan untuk mengelola tautan VHF	
Pesan 23: Perintah	Digunakan oleh stasiun pangkalan untuk	

penugasan grup	mengelola stasiun AIS lainnya	
Pesan 24: Laporan data statis	Setara dengan pesan Tipe 5 untuk kapal yang menggunakan peralatan Kelas B	
Pesan 25: Pesan biner slot tunggal	Digunakan untuk mengirimkan data biner dari satu perangkat ke perangkat lainnya	
Pesan 26: Pesan biner multi slot dengan status komunikasi	Digunakan untuk mengirimkan data biner dari satu perangkat ke perangkat lainnya	
Pesan 27: Pesan siaran sistem identifikasi otomatis jarak jauh	Pesan ini digunakan untuk deteksi jarak jauh kapal AIS Kelas A dan Kelas B (biasanya oleh satelit).	Sama seperti pesan 1, 2 dan 3

2.6 Inertial Measurement Unit (IMU)

IMU (*Inertial Measurement Unit*) merupakan alat yang memanfaatkan sistem pengukuran seperti gyroskop dan akselerometer untuk memperkirakan posisi relatif, kecepatan, dan akselerasi dari gerakan motor. IMU adalah bagian dari navigasi system yang dikenal sebagai Inertial Navigation System atau INS. Pertama kali didemonstrasikan oleh C.S. Draper tahun 1949, IMU menjadi komponen navigasi umum dari bidang dan kapal. Ada beberapa macam IMU yang biasa

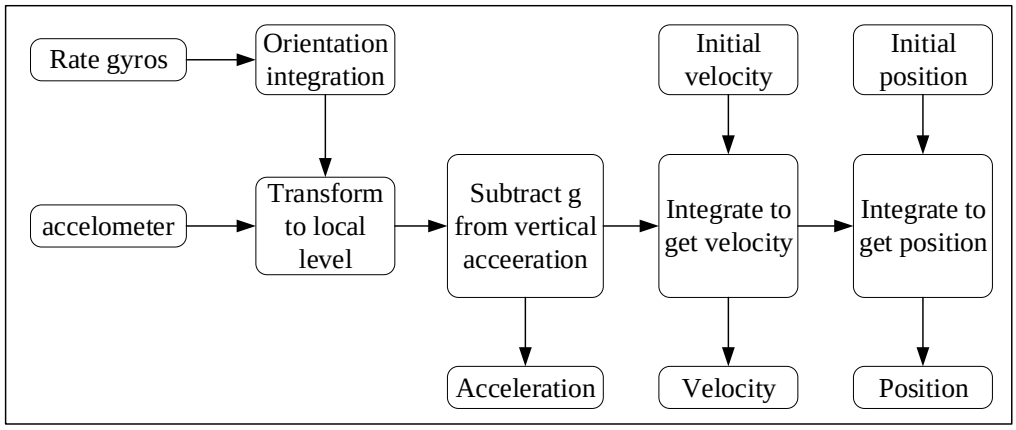
digunakan yaitu IMU gimbaled (Gambar 2.7(a)) dan IMUstrap-down (Gambar 2.7 (b)). IMU strap-downlebih umum dipakai saat ini. IMU mempertahankan 6-degree-of-freedom (DOF) yang memperkirakan gerakan yaitu posisi (X Y Z) dan orientasi (*roll*, *pitch*, *yaw*). Sistem seperti IMU hanya mempertahankan perhitungan terus menerus dari orientasi yang dikenal sebagai AHRS (Atitude and Heading Reference Sistem) dan dipergunakan dalam cara yang sama sebagai IMU tetapi mempertahankan representasi tidak menyeluruh. Sebagai tambahan untuk mempertahankan sikap motor 6-DOF, komersial IMU juga secara khas mempertahankan perkiraan dari kecepatan dan akselerasi.



(a) (b)

Gambar 2.7 (a) Inertial measurement unit gimbaled. (b) Inertial measurement unit strap-down

Data gyroskop (ω) diintegrasikan untuk mempertahankan perkiraan orientasi motor (θ) secara terus menerus. Di waktu yang sama, tiga akselerometer digunakan untuk memperhitungkan akselerasi motor (a) dengan segera. Data ini kemudian ditransformasikan melalui perhitungan dari orientasi motor relatif terhadap gravitasi, sehingga vector gravity dapat dihitung dan diekstrak dari pengukuran. Hasil akselerasi kemudian diintegrasi untuk mendapatkan kecepatan motor dan kemudian diintegrasi lagi untuk mendapatkan posisi (r).



Gambar 2.8 Diagram block inertial measurement unit [10]

IMU sangat sensitif untuk mengukur kesalahan di gyroskop dan akselerometer yang mendasar. Penyimpangan gyroskop mengarahkan pada kesalahan perhitungan dari orientasi motor, relatif terhadap gravitasi, menghasilkan penggagalan yang tidak tepat dari vektor gravitasi,seperti data akselerometer yang diintergasi dua kali, sisa vektor gravitasi akan menghasilkan kesalahan *quadratic* dalam posisi [19].

2.7 GPS

GPS adalah sistem radio navigasi satelit yang dikembangkan oleh DOD (*the U.S Dept. of Defense*) untuk keperluan navigasi global segala cuaca dimuka bumi pada sembarang waktu. Sistem ini memungkinkan pemakai GPS menentukan posisi, kecepatan gerak dalam koordinat tiga dimensi dan waktu dengan teliti. Sistem radio navigasi satelit ini terdiri dari tiga bagian yaitu : Space Segment, Control Segment, dan User Segment. Penentuan posisi GPS digambarkan dengan menggunakan nilai koordinat X dan Y atau garis bujur dan garis lintang (*longitude*/*latitude*).

GPS minimal harus memiliki 3 sinyal satelit untuk menghitung posisi 2D dan dibutuhkan 4 atau lebih sinyal satelit untuk menghitung 3 posisi (*longitude*, *latitude* dan *altitude*). Dengan informasi posisi, GPS dapat menghitung data-data lain, seperti : receptacle, arah, lintasan, jarak tempuh, matahari terbit & terbenam. Apabila dibandingkan dengan sistem dan metode penentuan posisi lainnya, GPS

mempunyai banyak kelebihan dan menawarkan lebih banyak keuntungan baik dalam segi operasional maupun dalam penentuan posisi [17].

2.7.1 GPS UBLOX NEO-8M

NEO-8M adalah salah satu modul GPS yang masuk dalam salah satu seri GPS UBLOX NEO-8 yang memiliki kinerja tinggi, *receiver* yang fleksibel, murah, dan menawarkan berbagai pilihan konektivitas hanya dalam miniatur 16 x 12,2 x 2,4 mm. Dengan arsitektur yang compact dan pilihan memori membuat NEO-6M ideal untuk dioperasikan dengan baterai perangkat *mobile*. Mesin 50-channel u-blox 8 menawarkan TTFF (*Time-To-FirstFix*) di bawah 1 detik. Mesin akuisisi yang memiliki 2 juta correlators ini memungkinkan untuk menemukan satelit secara langsung. Serta dengan desain dan teknologi yang inovatif menjadikan NEO-8M sebuah navigasi yang paling baik bahkan di lingkungan yang ekstrim.

Gambar 2.9 GPS U-blox neo8m
(Sumber : http://www.senith.lk/shop/item/1121/ublox-neo-8m-gps-module)

2.8 Arduino Uno

Arduino adalah sebuah kit elektronik *open Source* yang dirancang khusus untuk memudahkan bagi para seniman, desainer, dan siapapun yang tertarik dalam menciptakan objek atau mengembangkan perangkat elektronik yang dapat berinteraksi dengan bermacam-macam sensor dan pengendali.

Arduino UNO merupakan sebuah *board* mikrokontroler yang dikontrol penuh oleh ATmega328. Seperti yang ditunjukan pada gambar 1 dibawah, Arduino UNO mempunyai 14 pin digital *input*/*output* (6 di antaranya dapat digunakan

sebagai *output* PWM), 6 *input* analog, sebuah osilator Kristal 16 MHz, sebuah koneksi USB, sebuah power jack, sebuah ICSP *header*, dan sebuat tombol reset. Arduino UNO memuat semua yang dibutuhkan untuk menunjang mikrokontroler, mudah menghubungkannya ke sebuah computer dengan sebuah kabel USB atau mensuplainya dengan sebuah adaptor AC ke DC atau menggunakan baterai untuk memulainya.

Gambar 2.10 Konfigurasi pin ATMega 328 Arduino uno R3

2.8.1 Skematik Arduino

Skematik arduino *board* yang telah disederhanakan seperti pada gambar 2 *Shield* merupakan sebuah papan yang dapat dipasang diatas arduino *board* untuk menambah kemampuan dari arduino *board*. Bahasa pemograman yang dipakai dalam Arduino bukan bahasa assembler yang relatif sulit, melainkan bahasa pemograman mirip dengan bahasa pemrograman C++ yang disederhanakan dengan bantuan pustaka-pustaka (*libraries*) Arduino [16].

Adapun spesifikasi data teknis yang terdapat pada *board* Arduino UNO R3 adalah sebagai berikut:

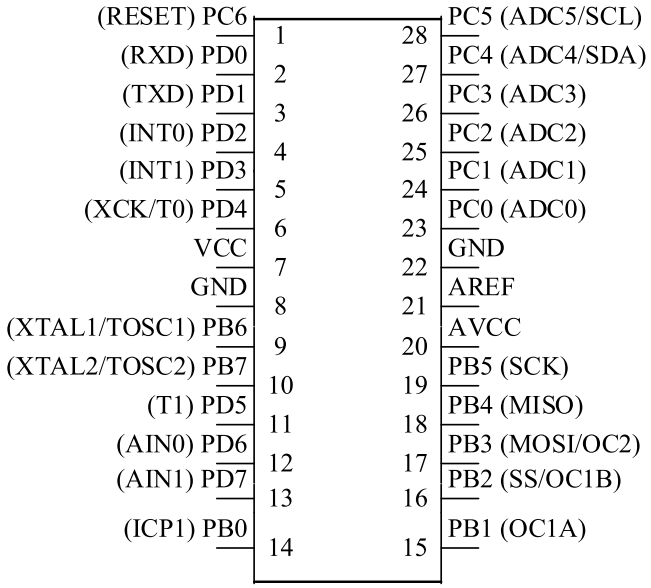
Tabel 2.3 Spesifikasi Arduino UNO

Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM *output*)

PWM Digital I/O Pins	6
Analog *Input* Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock *Speed*	16 MHz

2.8.2 Mikrokontroler Atmega 328P

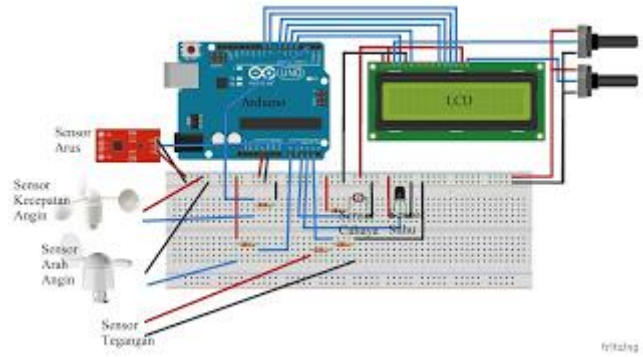
Arduino Uno R3 menggunakan mikrokontroler yang dikontrol secara penuh oleh mikroprosesor ATmega328P. Mikroprosesor yang digunakan ini sudah dilengkapi dengan konverter sinyal analog ke digital (ADC) sehingga tidak diperlukan penambahan ADC eksternal. Pada Gambar 2.11 dibawah ini merupakan penjelasan melalui gambar mengenai konfigurasi pin-pin yang merupakan bagian dari mikrokontoller ATMega328 yang digunakan didalam modul *board* arduino, sebagai berikut ini:



Gambar 2.11 Konfigurasi pin ATMega 328P

2.8.3 Kegunaan atau Fungsi Arduino

Arduino yang dikontrol penuh oleh mikrokontroler ATmega328, banyak hal yang bisa dilakukan itu semua tergantung kreatifitas anda. Arduino dapat disambungkan dan mengontrol led, beberapa led, bahkan banyak led, motor DC, relay, servo, modul dan sensor-sensor, serta banyak lagi komponen lainnya. Platform Arduino sudah sangat populer sekarang ini, sehingga tidak akan kesulitan untuk memperoleh informasi, tutorial dan berbagai eksperimen yang menarik yang tersedia banyak di internet. Dengan Arduino, dunia *Hardware* bisa bekerja sama dengan dunia *Software*. Anda bisa mengontrol *Hardware* dari *Software*, dan *Hardware* bisa memberikan data kepada *Software*. Semuanya bisa dilakukan dengan relatif mudah, murah, dan menyenangkan [16].



Gambar 2.12 Arduino yang digunakan untuk membaca sensor yang ditampilkan ke LCD

BAB 3
METODE PENELITIAN

3.1 Rancangan Penelitian

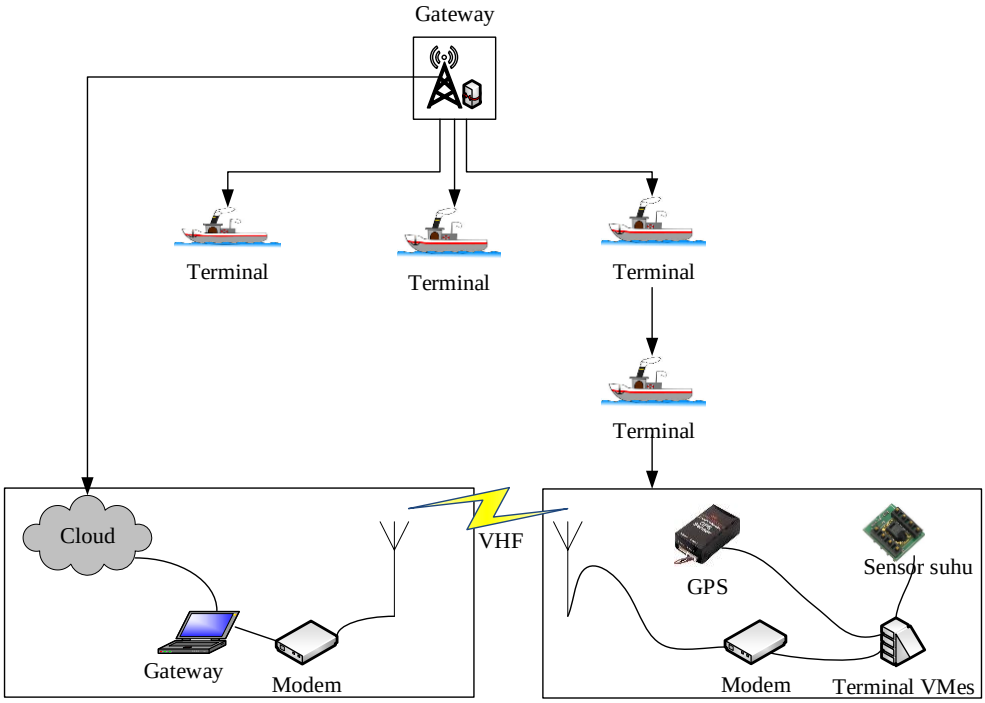
Pada penelitian ini untuk dapat mewujudkan sistem komunikasi yang menerapkan teknologi VMeS (*Vessel Messaging System*), diperlukan beberapa persyaratan. Persyaratan sebagai pokok utama kebutuhan sistem dalam menentukan bagaimana sistem bekerja dan fitur apa yang tersedia pada sistem. Ada dua jenis persyaratan, yaitu persyaratan perangkat keras, dan persyaratan perangkat lunak.

Sistem komunikasi VMeS yang dirancang pada penelitian ini menggunakan gelombang radio pada frekuensi 2.4 Ghz sebagai *channel* komunikasi dengan tipe komunikasi adalah *half duplex*. Pada kapal terpasang perangkat VMeS yang berintikan mikrokontroler Arduino Uno yang berfungsi untuk memproses semua perangkat sensor, GPS dan *protocol* komunikasi yang digunakan dalam proses pengiriman data melalui *gateway* sebelum ke *ground station*.

Perangkat *gateway* menggunakan sebuah radio komunikasi pada kanal frekuensi VHF 433Mhz yang berfungsi untuk meneruskan data ke *ground station* maupun untuk menangani sejumlah *client* kapal yang terhubung. Pada sistem *ground station* menggunakan perangkat PC maupun laptop yang berfungsi untuk menampilkan data *Monitoring* dari masing-masing *client* kapal. Data ditampilkan dalam bentuk *user interface* lokasi kapal (*map*), *heading* kapal (*compass*), suhu dan kelembaban, sikap kapal terhadap bumi (*yaw, pitch, roll*) dan kecepatan kapal dalam knot.

Penelitian rancang bangun dan pembuatan *prototype* dilaksanakan dalam beberapa tahapan yang dimulai dari perancangan hingga pengujian *prototype*. Pada gambar 3.1 menjelaskan aliran kegiatan penelitian dimulai dari perancangan, pembuatan dan pengujian sistem, serta analisa dan validasi sistem sehingga diperoleh sistem yang handal dan bekerja sesuai dengan perancangan dan tujuan awal pembuatan. Pokok utama adalah pengujian masing-masing modul dan pengujian beberapa *protocol* komunikasi digunakan seperti yang telah disebutkan

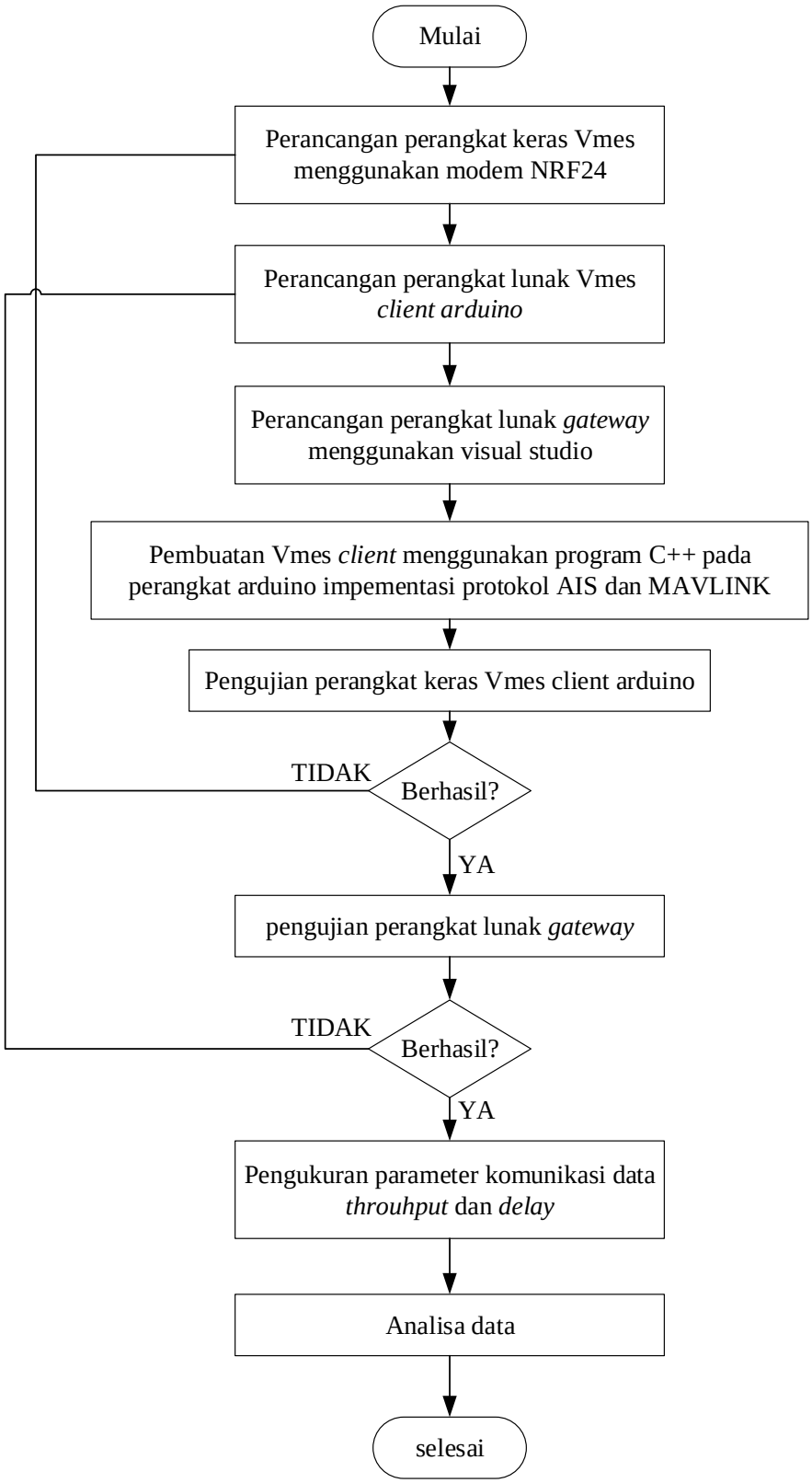
sebelumnya. Hasil akhir adalah sebuah kajian dan analisa untuk mendapatkan *protocol* komunikasi terbaik dalam menerapkan teknologi VMeS pada penelitian ini.



Gambar 3.1 Aplikasi rancangan system

Dari aplikasi perancangan system yang sudah di buat seperti gambar 3.1 dapat dibuat alur tahapan penelitian seperti gambar 3.2. *flowchart* tahapan penelitian menjelaskan langkah-langkah yang dilakukan dalam penelitian mulai dari *gateway* sampai dengan terminal yang akan dibahas dalam penelitian ini.

Perancangan perangkat lunak *gateway* menggunakan visual studio

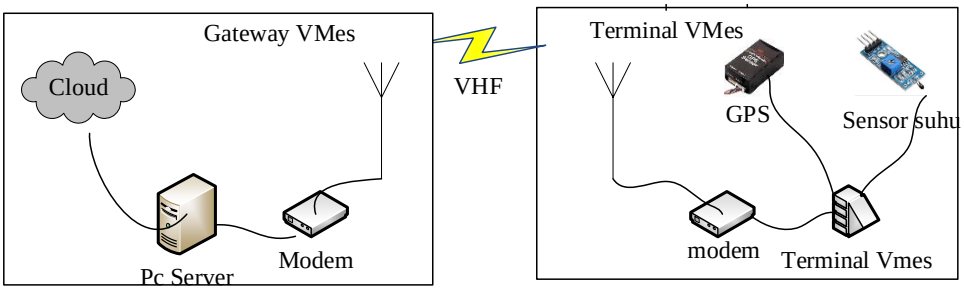


Gambar 3.2 *Flowchart* Tahapan Penelitian

Dari gambar 3.2 *flowchart* tahapan penelitian diatas menjelaskan langkah-langkah yang dilakukan pada penelitian pembuatan *prototype* pengembangan teknolgi VMeS menggunakan *protocol* komunikasi AIS, dan MAVLink, sebagai berikut :

3.2 Perancangan Sistem

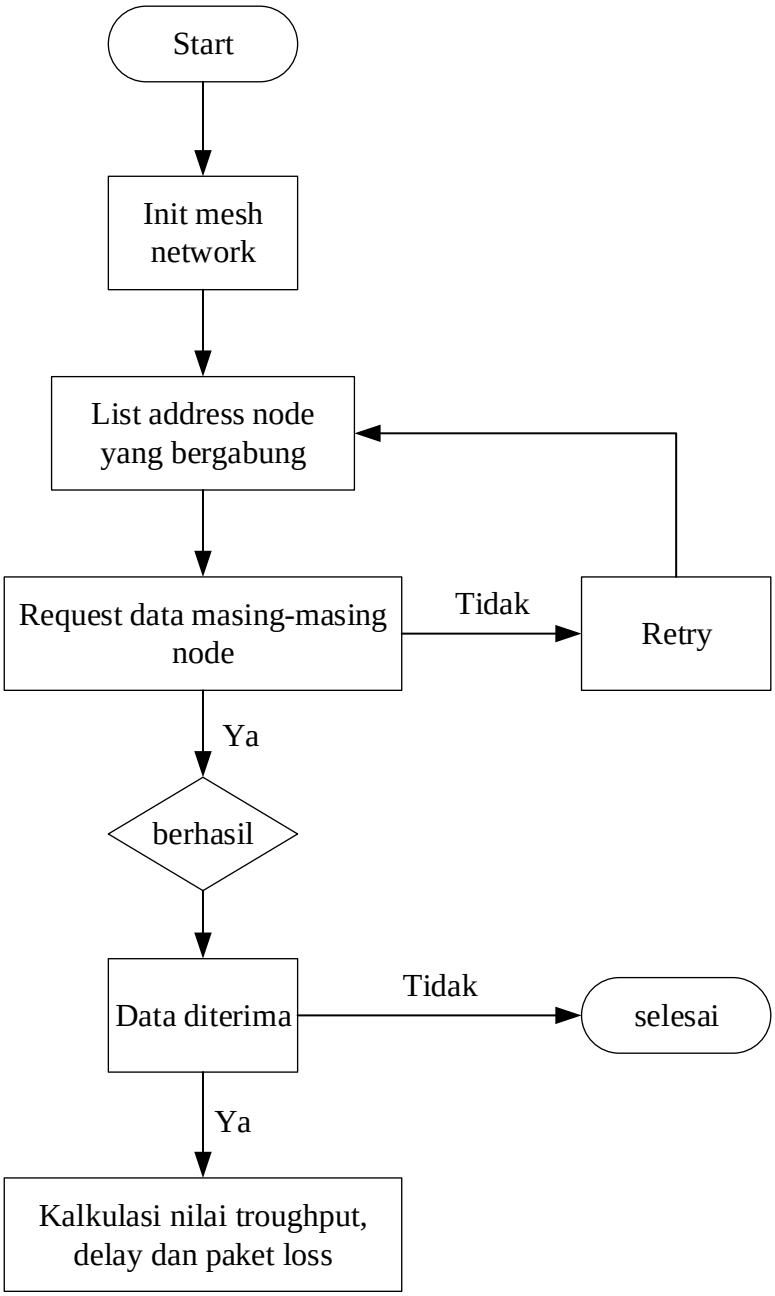
Sub bab perencangan sistem akan dijelaskan mengenai perencangan sistem yang akan dibuat. Perancangan sistem merupakan tahapan yang menjelaskan sistem secara keselurahan baik dalam bentuk diagram blok ataupun *flowchart*. Diagram blok sistem secara keseluruhan ditunjukkan dalam Gambar 3.3.



Gambar 3.3 Diagram blok sistem secara keseluruhan

3.2.1 Sistem pada *Gateway*

Flowchart sistem pada *Gateway* ditujukkan dalam Gambar 3.4. Gambar 3.4 memperlihatkan *flowchart* keseluruhan pada system *gateway*. *Flowchart* ini menjalaskan cara kerja sensor mulai dari menetukan tujuan *node* yang akan melakukan perintah pembacaan suhu, kelembaban udara dan sampai mengirimkan data kembali. Selanjutnya dikirimkan perintah untuk melakukan pembacaan data sensor. Setelah *node* menerima perintah baca data, jika keputusan yang diambil "ya" maka akan dilakukan pembacaan suhu dan pembacaan kelembaban maupun udara oleh *node* Sedangkan untuk keputusan yang diambil "tidak" maka *gateway* akan meminta data kembali.

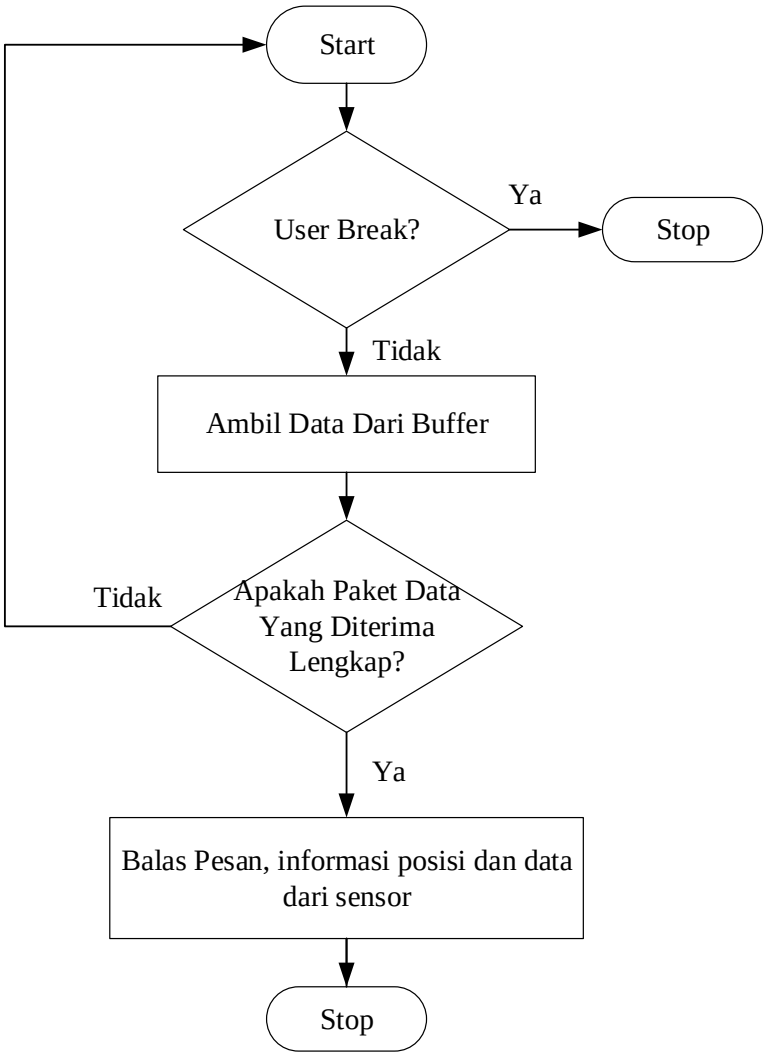


Gambar 3.4 Diagram Alir Sistem Pada *Gateway*

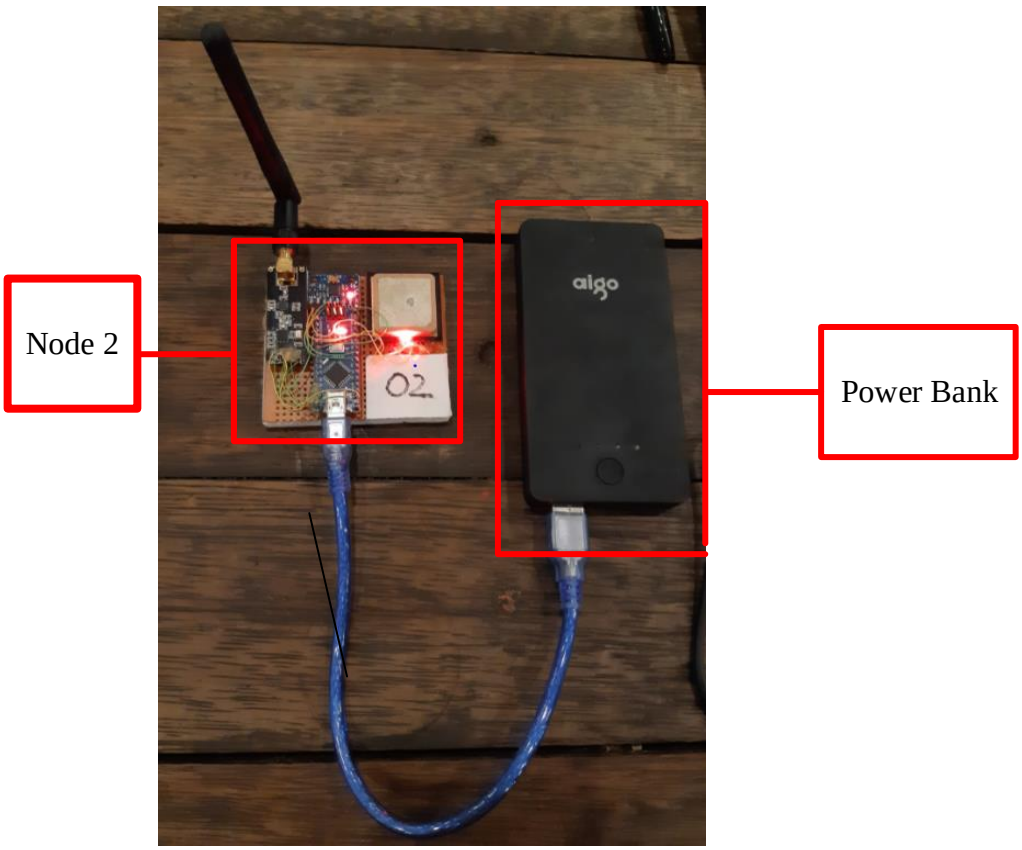
Pada program ini apabila ada data yang masuk maka akan terlebih dahulu diperiksa apakah data yang dikirim sudah lengkap atau belum dengan membandingkan antara FCS data yang dikirim oleh *gateway* atau terminal lain dengan FCS yang dihitung pada terminal lokal. Selanjutnya bisa mengirim balik pesan dan bisa menambahkan data-data dari GPS dan sensor-sensor yang ada di kapal.

3.2.2 Sistem pada Terminal

Aplikasi pada terminal VMeS yang dirancang pada tesis ini dikembangkan menggunakan bahasa pemrograman C++ digunakan pada Arduino UNO. Terminal komunikasi yang dirancang harus dapat berfungsi sebagai pengirim sekaligus juga sebagai penerima.



Gambar 3.5 Diagram Alir *Software* Pada terminal



Gambar 3.6 Rangkaian *node* terminal

3.3 Perancangan perangkat keras VmeS menggunakan NRF24

Perangkat keras yang dirancang berdasarkan pada kebutuhan data yang akan diolah yaitu : data suhu dan kelembaban, lokasi kapal dalam koordinat *latitude* dan *longitude*, sikap kapal dalam sumbu *yaw-pitch-roll*, *heading* kapal terhadap bumi. Kebutuhan perangkat keras akan transmisi data melalui saluran komunikasi 2.4 Ghz, untuk mentransmisikan data dari *client* kapal ke *ground station*. *Client* - *client* yang lain akan berfungsi sebagai *node* perantara, untuk meneruskan data dari *client* kapal ke *ground station* atau sebaliknya. Perancangan perangkat lunak diharapkan dapat menerapkan sistem VMeS yang dibuat, perangkat lunak dirancang agar dapat memproses data sensor-sensor pada *client* kapal.

3.3.1 Terminal VMes

Terminal Vmes merupakan terminal jaringan menggunakan modem NRF24L01, Terminal berlaku sebagai Master *node* yang bertugas mengelola jaringan, pada penelitian ini terdapat 2 *node child* yang merupakan *node* kapal 1 dan *node* kapal 2. Topologi jaringan yang digunakan merupakan jaringan *Mesh*. Setiap *node child* yang bergabung pada jaringan akan diidentifikasi berdasarkan *node address* nya, sehingga Master *node* setiap saat melakukan *update* jaringan untuk mendapatkan informasi *node* yang bergabung atau yang meninggalkan jaringan. Penerapan Ad-Hoc pada jaringan bergerak ini, berguna untuk mendapatkan *routing* karena setiap *node* yang bergabung dapat menjadi perantara atau *node* yang dilewati jika *node* tujuan diluar jangkauan atau melewati *node* perantara merupakan *routing* terpendek.

3.3.1.1 Sensor & GPS

Sensor-sensor yang digunakan pada penelitian ini dapat disebut satu-persatu sebagai berikut :

1. Sensor Imu MPU6050

   Merupakan sensor inersia untuk mendeteksi orientasi pada sumbu *yaw*, *pitch* dan *roll*. Sensor ini digunakan untuk mengukur tingkat kemiringan kapal yang bisanya menggunakan sensor *inclinometer*. Sensor IMU-MPU5060 merupakan sensor yang menggunakan komunikasi I2C agar dapat melakukan transaksi data dengan mikrokontroler Arduino.

2. Sensor BMP280

   Merupakan sensor barometer, berfungsi untuk mendapatkan informasi tekanan udara dan temperatur. Menggunakan I2C komunikasi untuk berkomunikasi dengan Arduino, melalui pin sda dan scl.

3. Sensor compass QMC5833

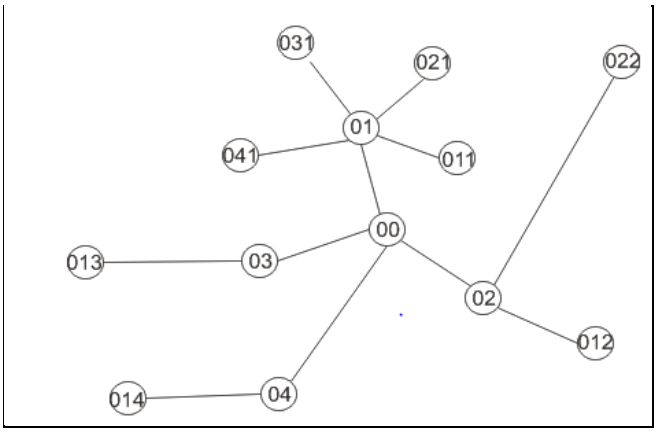
   Merupakan sensor untuk mengukur haluan kapal terhadap bumi, Menggunakan I2C komunikasi untuk berkomunikasi dengan Arduino, melalui pin sda dan scl.

4. GPS NEO-M8

Merupakan sensor untuk mendapatkan titik kooordinat lokasi kapal, sensor gps menghasilkan data dalam format NMEA. Setiap kalimat NMEA yang diterima dilakukan proses parse untuk mendapatkan variabel *latitude*, *longitude*, *speed*, *altitude*, jumlah satelit dan hdop.

3.3.1.2 Modul NRF24

Modem berupa modul NRF24 dirancang pada konfigurasi *tree topology*, sesuai dengan masing-masing alamat *node*. *Node* hanya dapat berkomunikasi secara langsung dengan *parent node* dan *children node*. Jaringan akan secara otomatis mengirim pesan ke tempat yang tepat. Sebagai ilustrasi *node* 00 adalah simpul 'basis'. *Node* 01-05 langsung berkomunikasi dengan *Node* 00, tetapi tidak dengan satu sama lain. Jadi untuk *Node* 01 untuk mengirim pesan ke *Node* 02, ia akan melakukan perjalanan melalui *Node* 00. *Node* 011, 021, 031 dan seterusnya yang merupakan *children* dari *Node* 01. *Node* 011 mengirim ke 02, maka akan mengirim melalui 01, melalui 00, dan ke 02. Oleh karena itu, jika meletakkan *Node* 011 pada jaringan, pastikan bahwa ada *node* 01 berada di jaringan.



Gambar 3.7 Detail Alamat *Node*

3.3.2 *Gateway* VMes

Gateway Vmes merupakan aplikasi *Ground station* yang dibuat menggunakan bahasa pemrograman C# dengan editor Visual Studio 2012. Sistem

Gateway merupakan *Software* Monitor untuk menampilkan semua informasi yang dikirimkan oleh *node child*. *Gateway* memiliki tugas utama untuk melakukan proses *decode Protocol* AIS dan MAVLink. *Protocol* AIS merupakan *Protocol* yang digunakan oleh *node* 2 dan *Protocol* MAVLink digunakan oleh *node* 1. Fitur-fitur yang direnacanakan pada *Software Gateway* dapat disebutkan sebagai berikut :

1. Monitor data yang diterima melalui terminal Vmes.
2. Identifikasi data apakah data merupakan *Protocol* AIS atau MAVLink.
3. Melakukan *decode* masing-masing *Protocol*.
4. Menampilkan informasi data dari masing-masing *Protocol*.
5. Menampilkan lokasi kapal 1 dan kapal 2 pada peta elektronik.
6. Menghitung jarak masing-masing kapal dengan lokasi *Gateway*.
7. Melakukan perhitungan *delay, throughput* dan *packet loss*.
8. Menampilkan grafik analisa jaringan pada antar muka sistem *Gateway*.

3.4 Arsitektur sistem *protocol* AIS

VMeS terdiri dari dua blok sistem yaitu VMeS *gateway* dan VMeS terminal yang terhubung dengan melalui gelombang radio pada kanal VHF. Arsitektur sistem untuk VMeS *gateway* dan VMeS terminal secara umum sama yaitu terdiri dari antenna VHF, modem VHF, terminal. VMeS terminal berperan sebagai VMeS *mobile station* dimana VMeS terminal akan bergerak dan dilengkapi dengan GPS (*Global Positioning System*) untuk mengetahui posisi *mobile station*. VMeS *Gateway* berfungsi sebagai VmeS *Base Station* untuk mengontrol sisi VMeS terminal dimana ada tambahan *database* untuk mendata seluruh VMeS terminal yang ada dalam jangkauan VMeS BS.[3]

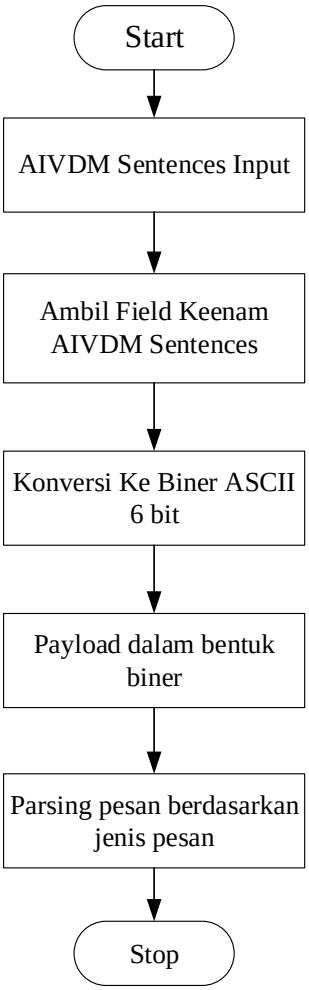
Sistem AIS terdiri dari AIS BS yang fungsinya mengkodekan pesan yang diterima dari AIS *receiver* untuk dimasukkan dalam *database*. Dalam penelitian ini AIS BS digantikan fungsinya oleh PC (*Personal Computer*) yang berfungsi mengolah data AIS dan terhubung dengan *database* AIS. *Database* AIS ini nantinya akan diintegrasikan dengan *database* VMeS yang dikonversi terlebih dahulu dalam format AIS yang nantinya akan ditampilkan dalam *database* untuk kapal dari Monitoring sistem.

3.4.1 Format Data AIS

Tabel 3.1AIS kelas A tipe pesan 1,2,3 [4]

Parameter	Panjang (Bit)
Message ID	6
Repeat indicator	2
User ID	30
Navigational status	4
Rate of turn ROTAIS	8
SOG	10
Position accuracy	1
Longitude	28
Latitude	27
COG	12
True Heading	9
Time stamp	6
Special manuvre indicator	2
Spare	3
RAIM-flag	1
Communication state	19
Jumlah bit	168

AIVDM terdiri dari tujuh bit *field* seperti yang dijelaskan dalam *protocol* data AIVDM. Data AIS berada pada *field* keenam. Pada proses *decoding* data yang dilustrasikan pada gambar 3.8 field keenam akan diambil. Data ini merupakan pesan dalam bentuk NMEA 0183 yang selanjutnya diubah ke biner [5]. Data biner ini nantinya akan diubah ke tipe pesan yang sesuai berdasarkan 6 bit pertama dari data biner.



Gambar 3.8 NMEA data *decoding*

Contoh untuk paket data AIS tipe 1 yaitu !AIVDM,1,1,,A,13HOI:0P0000VOHLCnHQKwvL05Ip,0*23. Data ini akan didekodekan untuk *Protocol* AIVDM terlebih dahulu yaitu diambil data pada *field* keenam yang berisi 13HOI:0P0000VOHLCnHQKwvL0\5Ip. Data ini kemudian diubah ke biner kemudian dipisah sesuai dengan field yang telah didefinisikan menurut ITU-R M.1371.

Pada *node client* 02 menggunakan *Protocol* AIS. Saat pertama kali *node client* dinyalakan maka *client* melakukan inisialisa sensor-sensor yang diintegrasikan ke perangkat. Kemudian perangkat *node client* melakukan konfigurasi modem dan menjalankan perangkat agar terhubung ke jaringan *Mesh*.

Pada proses ini *node client* akan menunggu perintah dari *gateway*, perintah ini diperoleh dari melalui jaringan. Jika data yang diterima dari *gateway* adalah *request* data maka *node client* akan mempersiapkan data sensor untuk di-*generate* kedalam *payload Protocol* AIS dan *packet* data sebesar 47 *byte* yang telah terbentuk akan dikirimkan ke *gateway*. Pada pembentukan *payload Protocol* AIS maka data yang disusun mengikuti format *Protocol* AIS dengan diakhiri dengan CRC untuk pengecekan *bit error* pada sisi *gateway*.

Jika data yang diterima oleh *node client* adalah data untuk perhitungan *packet delay*, maka *node client* akan mengirimkan kembali data yang telah diterima ke *gateway*. Data yang diterima pada proses ini merupakan data *timer* dalam format *microsecond*. Jika data yang diterima oleh *node client* 02 dan dikirimkan kembali ke *gateway*, maka pada *gateway* dapat melakukan perhitungan *delay* dengan mengurangi waktu sekarang *gateway* dengan data *timer* yang dikirim oleh *node client* . Untuk format protocol AIS dapat dilihat pada gambar 3.9.

Preamble	Start flag	Payload	CRC	Stop flag	Buffer
24 bit	8 bit	168 bit	16 bit	8 bit	32 bit

Gambar 3.9 format frame protokol AIS

FCS menggunakan *cyclic redundancy check* (CRC) polinomial 16-bit untuk menghitung *checksum* sebagaimana didefinisikan dalam ISO / IEC 3309: 1993. Bit-bit CRC harus disetel sebelumnya ke satu (1) pada awal perhitungan CRC. Hanya bagian data yang harus dimasukkan dalam perhitungan CRC. CRC ditambahkan ke segmen terakhir dari datagram. CRC dihitung atas semua fragmen datagram.

CRC didefinisikan sebagai sisa dari pembagian (modulo 2) dari polinom yang dibentuk oleh data dan yang disebut generator polinom, yang derajatnya sama dengan panjang CRC ditambah satu. Dalam beberapa kasus, beberapa nol dapat dimasukkan sebelum sisanya untuk mendapatkan panjang CRC yang terafiks. Pada sisi penerima, kesalahan dideteksi dengan membandingkan CRC yang dihitung dari data yang diterima dengan CRC yang terkandung dalam data frame. Definisi CRC dalam sistem AIS, CRC dimasukkan tepat setelah bit informasi di mana ia dihitung. Dengan definisi ini, tidak ada kesalahan yang terdeteksi di sisi penerima ketika

CRC bersama adalah nol, yang dapat dinyatakan sebagai CRC ([Data, CRC (Data)]).

3.5 Arsitektur sistem *protocol* MAVLink

Paket MAVLink dibuat oleh *header*, *payload* dan *Cyclic Redundancy* Check (CRC). *Header* berisi informasi penting untuk identifikasi sistem itu menerima paket. Oleh karena itu, ini disusun oleh tanda awal, panjang paket, nomor urut, ID sistem dan komponen dari sistem pengiriman dan ID dari pesan yang masuk. Muatan berisi data yang dianggap dikirim. Itu bervariasi, tergantung pada jenis pesan, misal Lokasi. CRC adalah kesalahan mendeteksi kode, digunakan untuk mengkonfirmasi integritas pesan. Informasi di atas dilanjutkan pada tabel 3.2[6].

Tabel 3.2 MAVLINK *message composition*

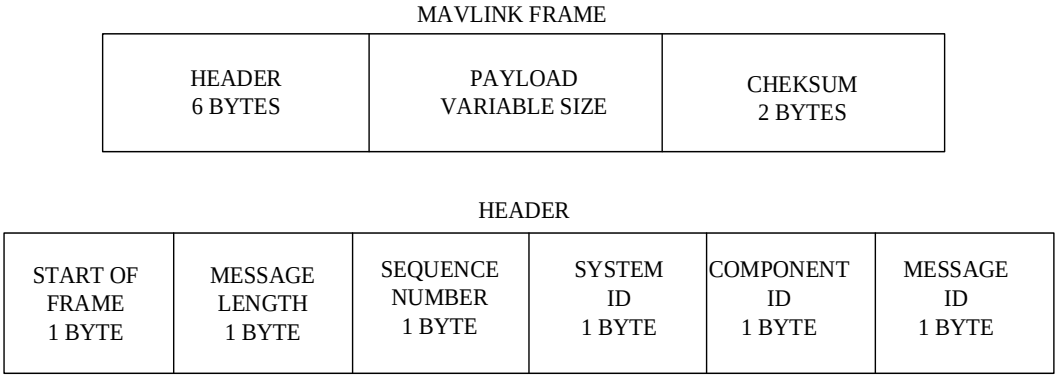
Bytes	Parameter	Explanation
0	Start sign	Announces the start of the *packet*
1	*Payload* length	Indicates length of the *payload*
2	Sequence Number	Used to detect *packet loss*
3	System ID	ID of the sending system, in order to differentiate different MAVs
4	Component ID	ID of the sending component, e.g. autopilot
5	Message ID	Identifies the type of message
6 (n+6) to	Data Actual	data of the message, depending on the above field
7 (n+7) to (n+8)	CRC	Allows to detect errors in the *packet*

Saat pertama kali *node client* 01 dinyalakan maka *client* melakukan inisialisa sensor-sensor yang diintegrasikan ke perangkat. Kemudian perangkat

node client melakukan konfigurasi modem dan menjalankan perangkat agar terhubung ke jaringan *Mesh*.

Pada proses ini *node client* akan menungu perintah dari *gateway*, perintah ini diperoleh dari melalui jaringan. Jika data yang diterima dari *gateway* adalah *request* data maka *node client* 01 akan mempersiapkan data sensor untuk di-*generate* kedalam *payload Protocol* MAVLink dan *packet* data yang telah terbentuk dalam format MAVLink sebesar 32 *byte* akan dikirimkan ke *gateway.* Pada pembentukan *payload Protocol* MAVLink maka data yang disusun mengikuti format *Protocol* MAVLink dengan diakhiri dengan CRC untuk pengecekan bit error pada sisi *gateway*.

Jika data yang diterima oleh *node client* adalah data untuk perhitungan *packet delay*, maka *node client* 01 akan mengirimkan kembali data yang telah diterima ke *gateway*. Data yang diterima pada proses ini merupakan data *timer* dalam format *microsecond*. Jika data *timer* yang diterima oleh *node client* 01 dan dikirimkan kembali ke *gateway*, maka pada *gateway* dapat melakukan perhitungan *delay* dengan mengurangi waktu sekarang *gateway* dengan data *timer* yang dikirim oleh *node client* 01.



Gambar 3.10 format frame protokol MAVLINK

MAVLink menggunakan satu CRC tambahan yang ditambahkan ke pesan CRC untuk mendeteksi ketidakcocokan dalam spesifikasi pesan. Hal ini berguna mencegah dua perangkat menggunakan versi pesan yang berbeda dari salah pada proses *encoding* pesan dengan panjang yang sama.

Ketika generator kode MAVLink dijalankan, dibutuhkan checksum dari struktur XML untuk setiap pesan dan membuat array yang menentukan

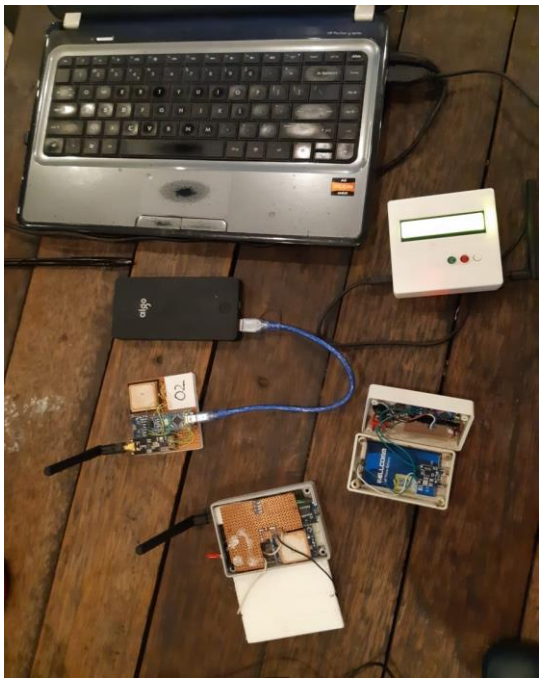
MAVLINK_MESSAGE_CRCS. Hal ini digunakan untuk menginisialisasi array mavlink_message_crcs [] dalam implementasi. Ketika checksum pada pesan dihitung, byte tambahan ini ditambahkan pada akhir data setelah checksum dihitung. Hasilnya adalah jika XML berubah maka pesan akan ditolak oleh penerima karena memiliki checksum yang salah. Hal ini memastikan bahwa hanya pesan-pesan di mana pengirim dan penerima menggunakan struktur pesan yang sama akan dilewatkan.

BAB 4
HASIL DAN PEMBAHASAN

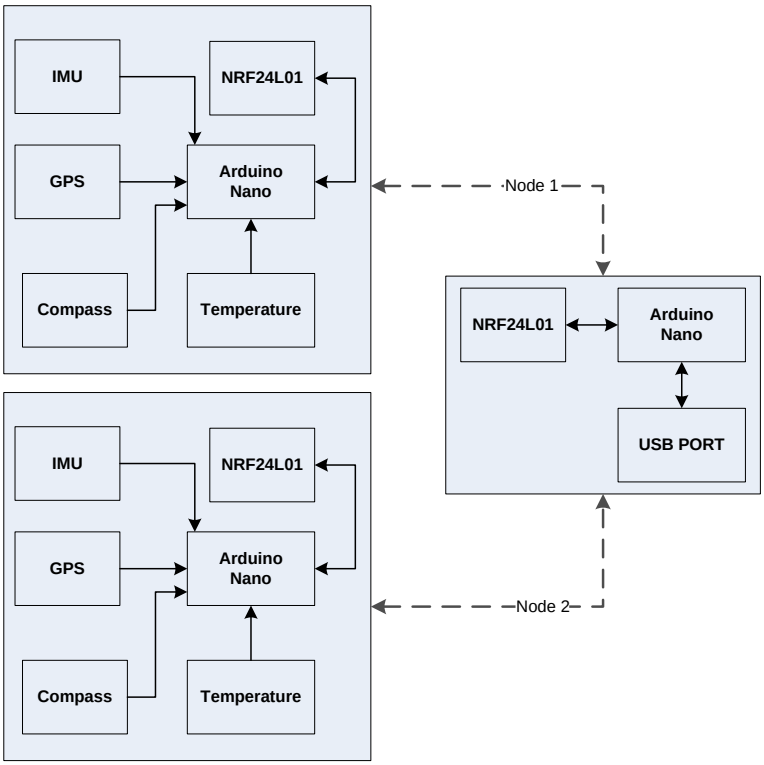
Pada bab ini akan dijelaskan mengenai analisa dan pengujian sistem yang telah dibuat berdasarkan perancangan pada bab sebelumnya. Pengujian dilakukan melalui beberapa tahap pengujian pada masing-masing perangkat sensor, mikrokontroler dan perangkat komunikasi data untuk mendapatkan hasil keluaran yang sesuai dengan tujuan penelitian ini. Pada penelitian ini, *Mesh* dan *node* membentuk jaringan *Mesh* secara nirkabel. Seperangkat *wireless Mesh* sensor *node* membentuk infrastruktur yang bersifat bergerak atau *mobile*. *Wireless sensor node* dapat ditambahkan maupun dihapus dari jaringan secara ad-hoc.

4.1. Implementasi *Hardware*

Hasil implementasi *Hardware* akan menjelaskan mengenai rangkaian *Hardware* yang akan digunakan sebagai sistem Monitoring. Pada gambar 4.1 adalah gambar yang menunjukan *hardware* secara keseluruhan.



Gambar 4.1 Implementasi *Hardware* Secara Keseluruhan



Gambar 4.2 Blok Diagram Hardware Secara Keseluruhan

Pada sisi terminal menggunakan arduino nano, GPS NEO-M8, sensor kompas QMC5883, modul NRF24L01 dan sensor BMP280. Dan pada sub bab ini akan dijelaskan tentang rangkaian proses pengiriman data dari *node* ke *gateway* sistem Monitoring.

4.2. Implementasi Pada Sisi *Gateway*

Implementasi perangkat pada sisi *gateway* jaringan WSN merupakan perangkat nirkabel yang bertindak layaknya modem, yaitu modul NRF24L01. Unit proses pada sisi *gateway* menggunakan Arduino Nano Atmega 328. Perangkat ini terhubung ke PC maupun laptop melalui saluran komunikasi USB. Tugas utama *gateway* WSN adalah mengelola jaringan *Mesh*, menyediakan sistem pengalamatan dan perutean untuk modul radio RF24, dalam proses pembentukan jaringan WSN. *Node* Master dalam hal ini *gateway* melacak *node*ID unik dan alamat RF24Network yang telah ditentukan. Ketika sebuah *node* dipindahkan secara fisik, atau kehilangan koneksi ke jaringan, dapat diatur untuk secara otomatis bergabung kembali dengan *Mesh*, dan mengkonfigurasi ulang dalam jaringan.

Alamat jaringan dapat dilihat sebagai alamat MAC, dan *node* ID dapat dilihat pula sebagai alamat IP statis. Saat bergabung ke jaringan, *node* akan meminta alamat dan diidentifikasi melalui *node* ID. Penerapan pada mikrokontroler Arduino semua *node* lain, daftar alamat tidak disimpan. Untuk memaksa konvergensi jaringan, harus melakukan *restart gateway*. Jika *node* dikonfigurasikan untuk memverifikasi koneksi pada interval yang ditentukan, *gateway* harus dibiarkan *offline* selama selang waktu tertentu.

Node ID adalah identitas yang bersifat unik, alamat RF24Network berubah secara dinamis dalam struktur yang ditentukan secara statis. Karena struktur ini, setiap *node* dapat berkomunikasi dengan Master *node*, karena alamat *gateway* atau Master *node* selalu bernilai (00). Master *node* menyimpan daftar setiap *node* pada jaringan. Komunikasi dari *node*-ke-*node* selalu memerlukan permintaan alamat untuk dikirim ke Master *node*, karena *node* secara independen dapat mengubah alamat setiap saat.

4.3. Implementasi Pada Sisi Terminal

Implementasi perangkat sensor GPS NEO-M8, kompas QMC5883, modul dan sensor BMP280 pada sisi terminal *node*, merupakan sumber data yang akan dikirimkan ke Master *node*. Sensor dan modem NRF24 terhubung ke perangkat Arduino Nano 328 pada terminal *node*, dapat dijelaskan sebagai berikut :

1. Gps terhubung secara serial ke Arduino Nano, melalui pin 10 dan 11 yang dikonfigurasi sebagai port serial menggunakan pustaka program *Software* Serial.
2. Sensor kompas QMC5833 terhubung ke Arduino Nano, melalui saluran komunikasi I2C.
3. Sensor kompas IMU MPU6050 terhubung ke Arduino Nano, melalui saluran komunikasi I2C.
4. Sensor kompas BMP280 terhubung ke Arduino Nano, melalui saluran komunikasi I2C.
5. Modem NRF24L01 terhubung ke Arduino Nano, melalui saluran komunikasi SPI.

4.3.1 Implementasi Pada *Node* 1

Pada *node* 1 perangkat sensor yang terpasang adalah sensor GPS, sensor kompas, IMU MPU6050 dan sensor BMP280. Sensor-sensor dihubungkan seperti penjelasan sub bab 4.3.

1. Sensor gps berfungsi untuk mendapatkan data lokasi dalam format *latitude* dan *longitude*, data jumlah satelit, data hdop *(horizontal dilution of precision*).
2. Sensor kompas berfungsi untuk mendapatkan data haluan perangkat terhadap bumi dalam rotasi *yaw*.
3. Sensor BMP280, berfungsi untuk mendapatkan data tekanan udara, *altitude* dan suhu.
4. Sensor IMU MPU6050, berfungsi untuk mendapatkan data orientasi perangkat pada sumbu *yaw, pitch* dan *roll*.

Perangkat modem NRF24L01 terinstal pada perangkat *node* 1 dengan *node* ID 1.

4.3.2 Implementasi Pada *Node* 2

Pada *node* 2 perangkat sensor yang terpasang adalah sensor GPS, sensor MPU 6050. Sensor-sensor dihubungkan seperti penjelasan sub bab 4.3.

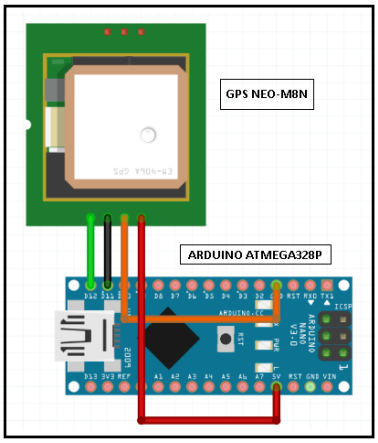
1. Sensor gps berfungsi untuk mendapatkan data lokasi dalam format *latitude* dan *longitude*, data jumlah satelit, data hdop *(horizontal dilution of precision*).
2. Sensor IMU MPU6050, berfungsi untuk mendapatkan data orientasi perangkat pada sumbu *yaw, pitch, roll* dan suhu.

Perangkat modem NRF24L01 terinstal pada perangkat *node* 2 dengan *node* ID 2.

4.4. Pengujian Sensor *Wireless Mesh node* 01

4.4.1 Pengujian GPS NEO-M8

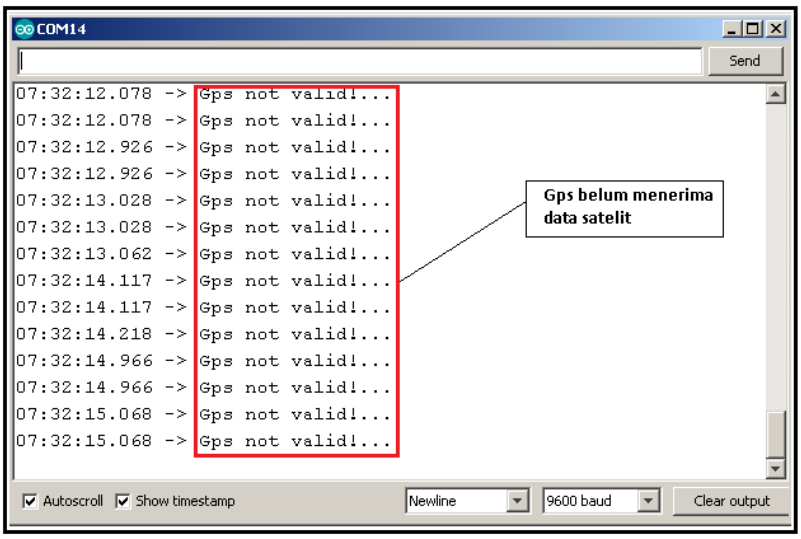
Untuk melakukan pengujian modul GPS Ublox Neo-08M perlu menghubungkan GPS dengan perangkat Arduino seperti *wiring* diagram gambar 4.3 berikut.



Gambar 4.3 *Wiring* Diagram Pengujian GPS

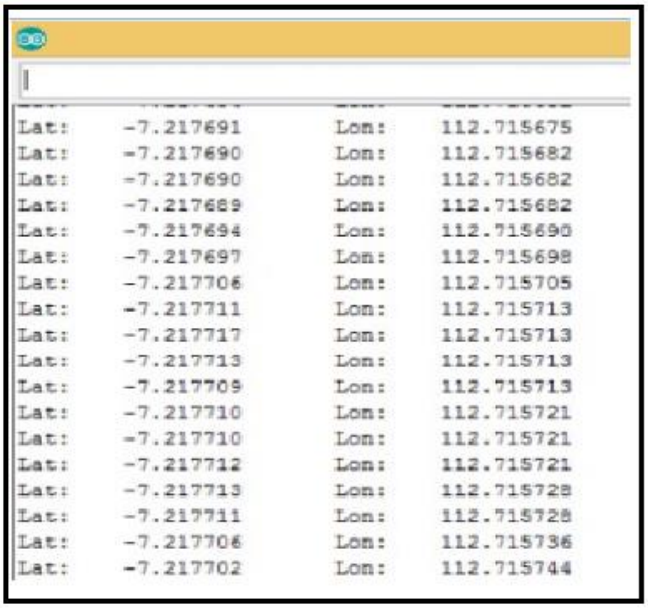
Pada Gambar 4.3 transaksi data antar GPS dengan mikrokontroler melalui komunikasi data serial Arduino Nano, secara *default baudrate* GPS adalah 9600, sehingga pada pengaturan *baudrate* program menyesuaikan dengan *baudrate* GPS.

Hasil pengujian dapat dijelaskan dalam bentuk gambar-gambar pengujian, pada pengujian ini peneliti menggunakan fitur terminal Monitor pada arduino ide untuk menampilkan hasil pembacaan data lokasi GPS dalam format koordinat *latitude* dan *longitude*. Pada pengujian GPS pertama kali data gps tidak teridentifikasi sebagai data NMEA yang valid hal ini disebabkan gps belum menerima data NMEA dari satelit. Gambar 4.4 berikut merupakan hasil pengujian gps saat belum menerima data NMEA dari satelit.



Gambar 4.4 Hasil Pengujian GPS Ublox Neo-8M Saat GPS Belum Menerima Data Satelit.

Untuk mendapatkan data GPS yang valid sesuai dengan format NMEA, maka dibutuhkan waktu beberapa menit hingga indikator pada modul GPS menyala. Indikator ini akan berkedip dengan interval waktu tertentu. Pada modul GPS neo-M8N indikator led berwarna biru.

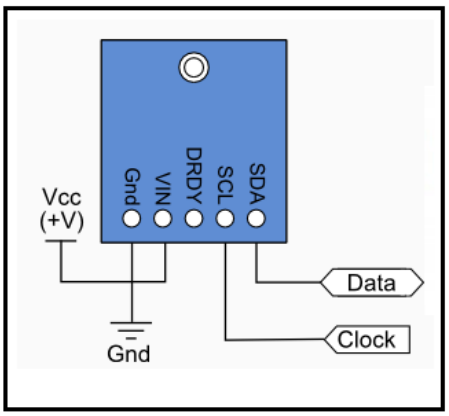


Gambar 4.5 Hasil Pengujian GPS Ublox Neo-8M

- Latitude = garis lintang mengarah dari khatulistiwa (0) ke kutub selatan, atau khatulistiwa ke kutub utara (sudut 0-90 dan 0 -90)
- Longitude = garis bujur adalah garis horizontal seperti dari khatulistiwa. Sudut 0 (*Greenwich*) ke arah Hawai adalah 0-180, sedangkan kebalikannya dari 0 ke -180

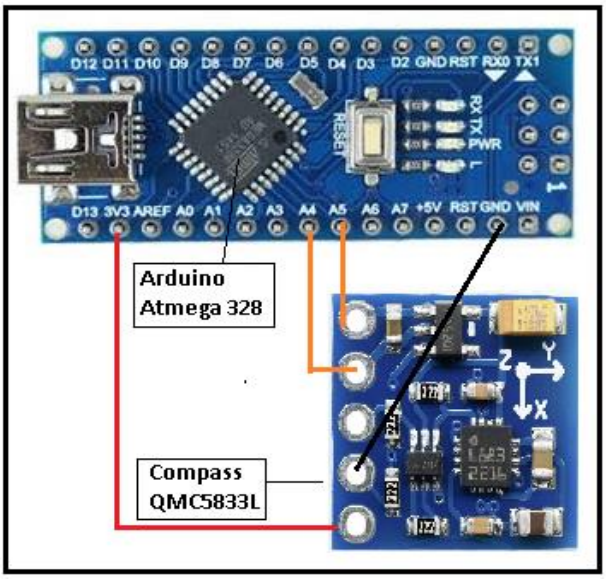
4.4.2 Pengujian Kompas QMC5883

Modul kompas dapat dihubungkan ke perangkat Arduino melalui pin-pin kompas seperti yang ditunjukkan pada gambar 4.6 di bawah ini.



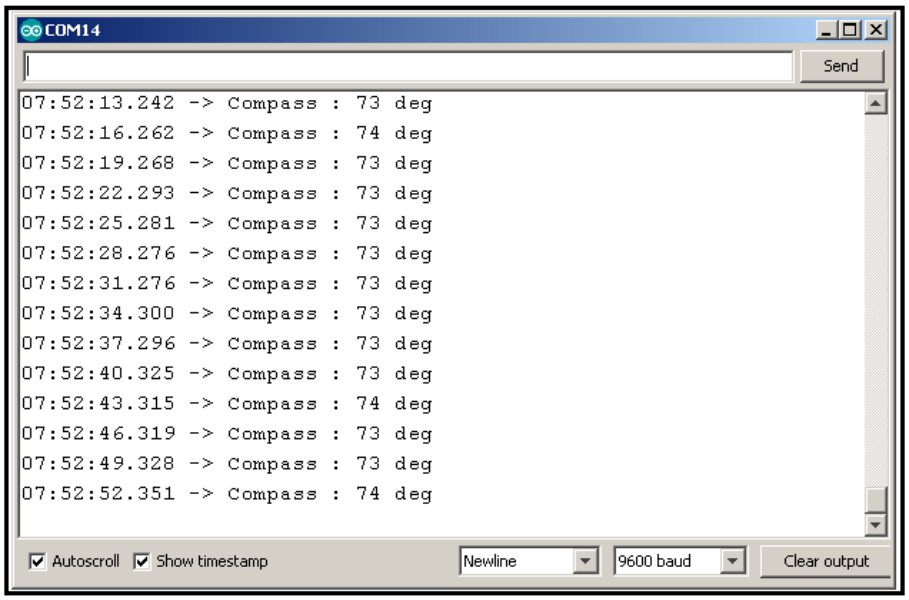
Gambar 4.6 Pin-pin Kompas

Modul kompas QMC5883L terhubung ke Arduino seperti ditunjukkan pada gambar 4.7 berikut ini.



Gambar 4.7 *Wiring* Diagram Pengujian Kompas

Hasil pengujian dapat dijelaskan dalam bentuk gambar-gambar pengujian, pada pengujian ini peneliti menggunakan fitur terminal Monitor pada arduino ide untuk menampilkan hasil pembacaan sensor kompas dalam sumbu *yaw*. Gambar 4.8 berikut merupakan hasil pengujian pembacaan sensor kompas dan ditampilkan pada terminal Monitor.



Gambar 4.8 Pengujian Kompas QMC5883L

Hasil keluaran yang ditunjukkan pada hasil pengujian kompas merupakan sudut orientasi perangkat terhadap bumi. Sehingga rentang pembacaan sudut adalah 0-360 derajat.

4.4.3 Pengujian Sensor BMP280

BMP280 mengukur tekanan dan temperatur, karena suhu mengubah densitas gas seperti udara. Pada suhu yang lebih tinggi, udara tidak sepadat dan berat, jadi itu mengurangi tekanan pada sensor. Pada suhu yang lebih rendah, udara lebih padat dan berat lebih banyak, sehingga memberikan lebih banyak tekanan pada sensor. Sensor menggunakan pengukuran suhu secara *real time* untuk mengkompensasi pembacaan tekanan pada perubahan dalam densitas udara. BMP280 berkomunikasi dengan Arduino melalui pin I2C, pin SDA dan pin SCL. Sebagai suplai daya dc pin daya sensor bmp280 dihubungkan pada pin vcc 3.3Vdc dan GND arduino Nnao.

Pada tahap pengujian sensor BMP280, langkah-langkah dalam pengujian dapat dijabarkan sebagai berikut:

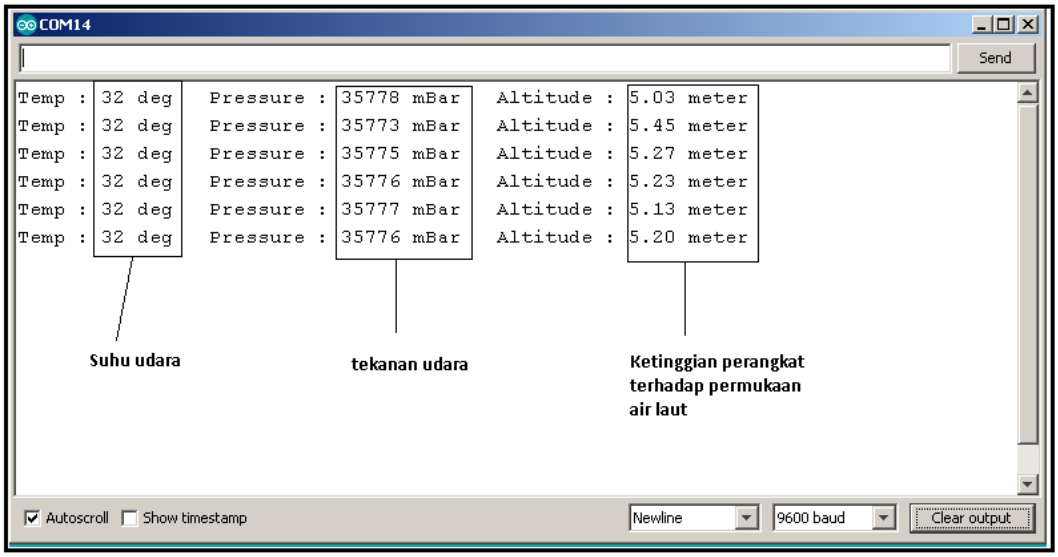
1. Menyiapkan perangkat sensor altimeter yang berintikan chip sensor BMP280.

2. Menghubungkan sensor BMP280 pada pin IC2 komunikasi arduino nano pada pin SDA (data), SCL(*clock*), *ground* dengan *ground* arduino dan pin sumber daya 3.3Vdc.
3. Menyiapkan kode program yang ditulis menggunakan bahasa C++ pada arduino ide 1.8.5 dan melakukan uji pembacaan pada terminal Monitor arduino ide.

Pada penelitian program, peneliti menambahkan rumus untuk mendapatkan ketinggian berdasarkan pembacaan suhu dan tekanan udara. Berikut potongan kode program untuk mendapatkan nilai ketinggian sensor terhadap permukaan air laut.

```
Po = 1013.25 //(Average sea-level pressure)
p = bmp.readPressure();
T = bmp.readTemperature();
altitude = (((po/p)^(1/15.257))*(T+273.15))/0.0065
```

Pada gambar 4.9 terlihat dari hasil pengujian didapatkan suhu udara sebesar 32 deg, tekanan udara 35776 mBar dan ketinggian perangkat terhadap permukaan air laut sebesar 5.20 meter.



Gambar 4.9 Pengujian BMP280

4.5. Pengujian Perangkat Nirkabel

Pada pengujian perangkat nirkabel, penulis menggunakan NRF24L01 2.4GHz yang terdiri dari antena *pigtail*, beroperasi pada pita 2,4 GHz ISM dengan lebar saluran 1 MHz yang terdapat 125 saluran yang dapat digunakan. Karena perangkat Wi-Fi menggunakan sebagian besar saluran yang rendah sehingga diatur menggunakan saluran tertinggi pada perangkat ini untuk menghindari interferensi dengan saluran WIFI.

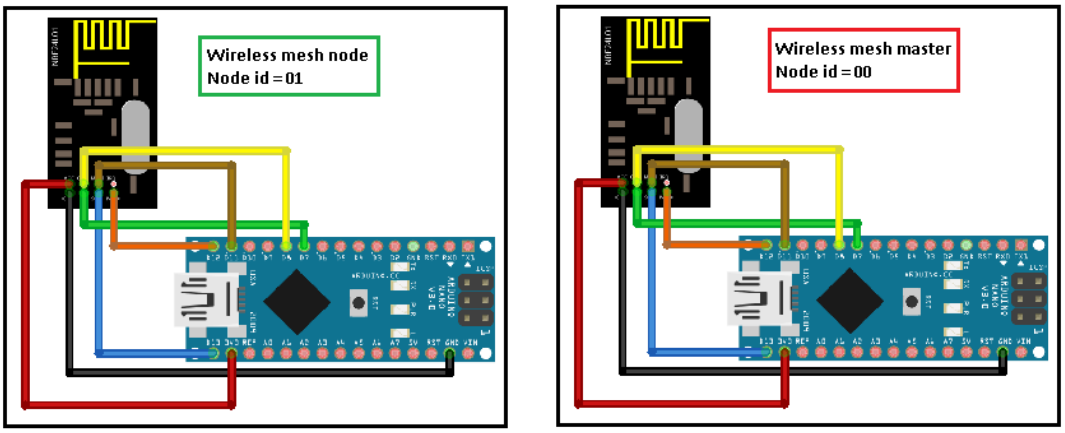
Modul NRF24L01 dihubungkan dengan unit mikrokontroler menggunakan komunikasi SPI. Penulis memanfaatkan pustaka program RF24 yang dikembangkan oleh TMRH20 untuk Arduino dan Raspberry Pi, yang secara efektif membangun TCP / IP *suite* (Gambar 4.10). Untuk percobaan ini, penulis menggunakan pustaka jaringan RF24 dan RF24 *Mesh* untuk Arduino.

TMRH20 RF24 Library	TCP/IP Suite Equivalent
RF24Mesh	Application Layer(DHCP)
RF24Ethernet	Transport Layer
RF24Network	Network/Internet Layer
RF24	Link Layer

Gambar 4.10 Persamaan TMRH20 libraries dan TCP/IP

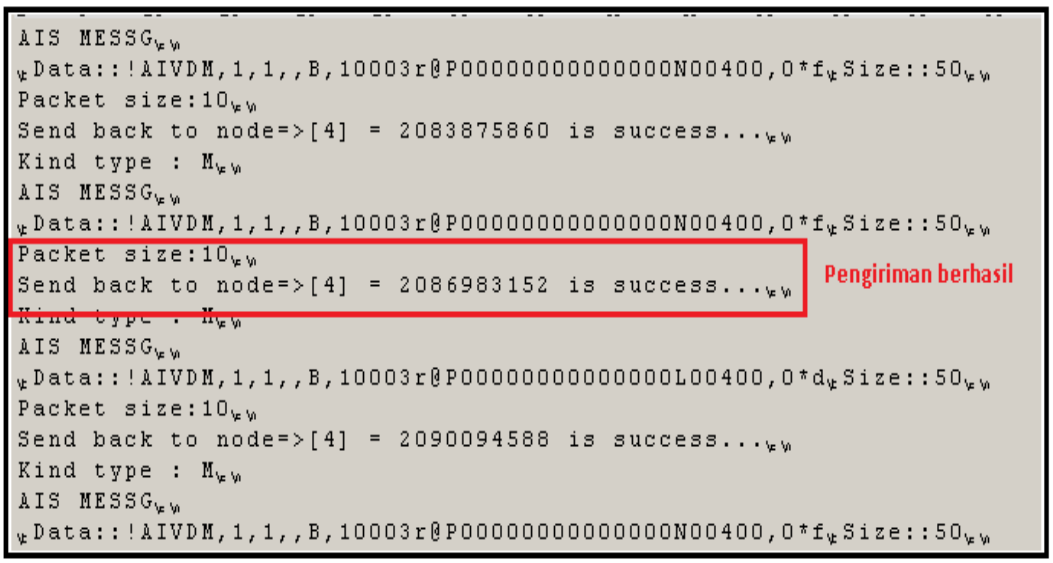
Untuk melakukan percobaan ini, penulis menggunakan 2 perangkat yang berfungsi sebagai *Wireless Mesh node* dan *Wireless Mesh Master*. Hal penting yang perlu dilakukan adalah mengatur *Node* id masing-masing perangkat. Untuk *Wireless Mesh node*, seperti yang telah disebutkan diatas didefiniksan pada *Node* id 01 dan *Wireless Mesh Master* didefiniskan pada *Node* id 00.

Berikut gambar *wiring* diagram pengujian modul nirkabel NRF24L01



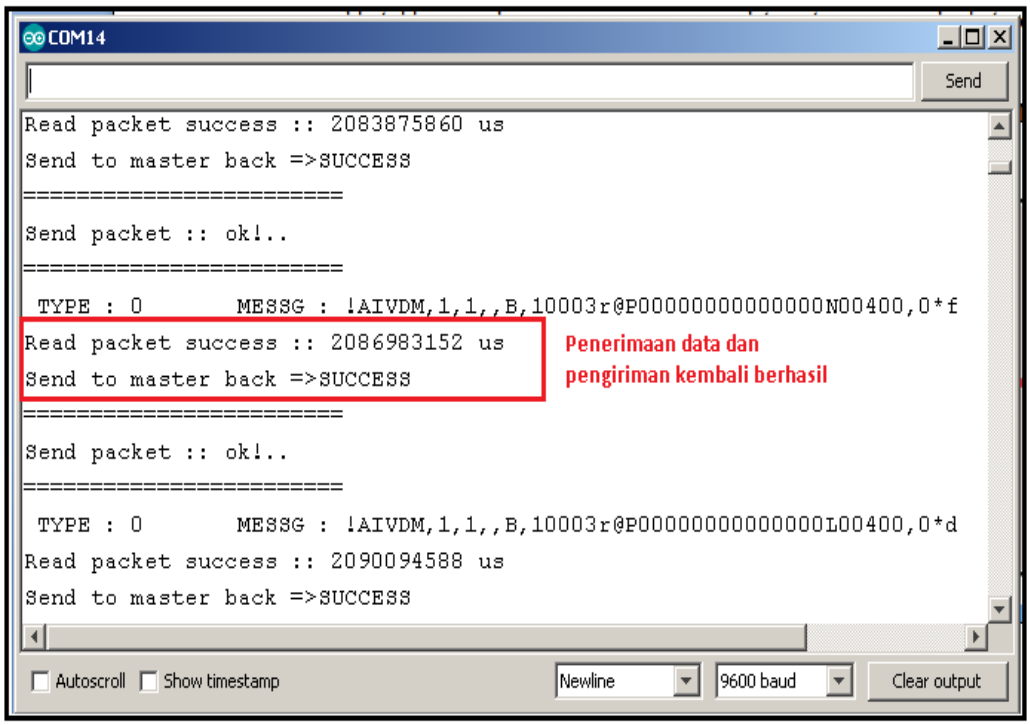
Gambar 4.11 *wiring* diagram pengujian modul nirkabel NRF24L01

Pada pengujian yang dilakukan *Wireless Mesh Master* akan mengirimakan data *timer* yang terintegrasi pada chip mikrokontroler arduino Nano 328p ke *Wireless Mesh node 00. Wireless Mesh node* akan me-*replay* data yang diterima ke *Wireless Mesh Master* kembali. Hal ini bertujuan untuk mengetahui apakah komunikasi dapat berjalan dengan baik dan untuk mengukur waktu yang dibutuhkan pada proses pengiriman data atau yang disebut *end delay measurement*. Gambar 4.12 merupakan hasil pengujian pengiriman data dari Master *node* 00 ke *node* 01.



Gambar 4.12 Pengujian pengiriman data dari Master ke *node*

Gambar 4.13 merupakan hasil pengujian penerimanan data dari Master *node* 00 ke *node* 01 dan proses pengiriman kembali ke Master *node*.



Gambar 4.13 Pengujian penerimaan data dari Master dan pengiriman kembali ke Master

Gambar 4.14 menunjukan bahwa pengujian pada modul nirkabel NRF24L01 data telah diterima dan pengiriman data kembali berhasil dengan melihat hasil yang ditampilkan pada Arduino ide.

Berdasarkan proses pengujian yang telah dilakukan dapat diperoleh hasil pengujian keberhasilan pengiriman dan penerimaaan serta pengukuran *delay* pengiriman. Pada gambar hasil pengujian dapat dianalisa *delay* pengiriman sebagai berikut :

Master *node* mengirimkan data *timer* yang diambil dari *internal timer* mikrokontroler dalam satuan *microsecond*, menggunakan fungsi *micros*(). Berikut gambar kode program yang menjelaskan lebih detail proses ini.

```
if(trySend){
  TX_ON();
  uint32_t timer_ = micros();
  int timer_size = String(timer_).length();
  PACKET_SIZE[NODE_INDEX] = timer_size;
  Serial.print("Packet size:");
  Serial.println(PACKET_SIZE[NODE_INDEX]);
  RF24NetworkHeader header_(node_target,OCT);
  if(network.write(header_,&timer_,sizeof(timer_))){
     Serial.print("Send back to node=>[");
     Serial.print(node_target,OCT);
     Serial.print("] = ");
     Serial.print(timer_);
     Serial.println(" is success...");
  }else{
     Serial.print("Send back to node=>[");
     Serial.print(node_target);
     Serial.println("] failed");
  }
  TX_OFF();
  trySend = false;
```

Gambar 4.14 kode program pengiriman data *timer* dari Master ke *node*

Saat proses pengujian diperoleh angka *timer* T = 2086983152 us yang dikirimkan oleh Master ke *node*, dan dikrimkan kembali ke Master. Master akan menerima data *timer* kembali jika proses pengiriman berhasil dilakukan. Data *timer* yang diterima oleh Master akan digunakan sebagai nilai pengurang bagi nilai *timer* yang ter-*update* sehingga akan diperoleh nilai *delay* pengiriman dan penerimaan. Nilai *delay* dapat dilakukan perhitungan sebagai berikut :

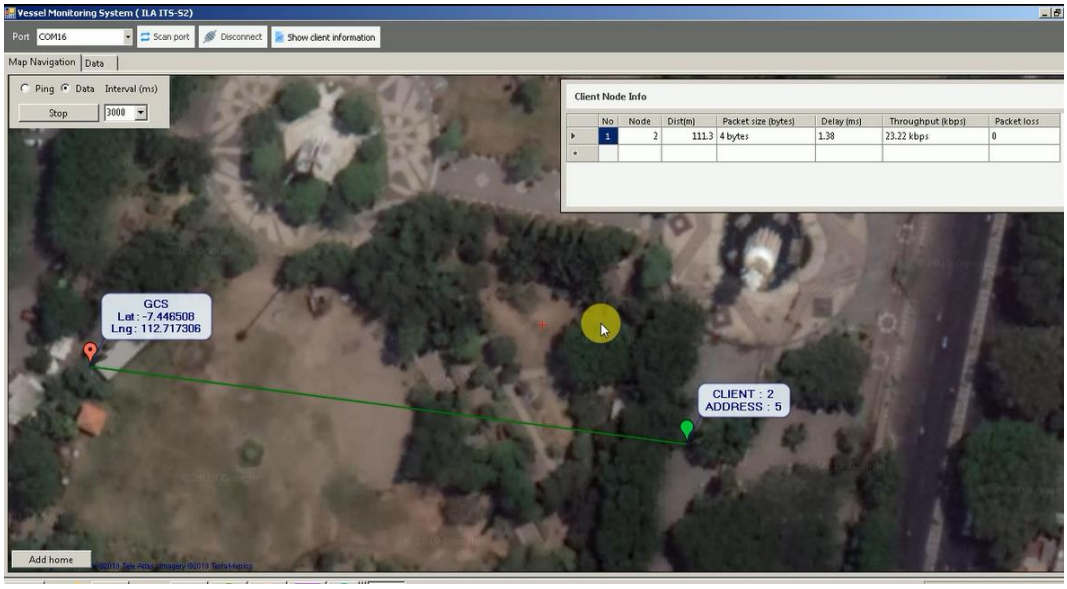
$$delay = (waktu\ paket\ diterima - waktu\ paket\ dikirim)/2$$
$$delay = (208816312\ us - 2086983152\ us)/2$$
$$delay = 615800\ us$$
$$delay = 61.58\ ms$$

Berikut Gambar 4.15 pengukuran *delay* yang ditampilkan pada LCD Master dan hasil pada LCD hasilnya sama dengan hasil perhitungan.

Gambar 4.15 Pengukuran *delay*

4.5.1 Pengujian Jarak Jangkuan Modul *Wireless* NRF24L01

Untuk pengujian pengiriman pesan antar sensor *node* dilakukan bertujuan untuk mengetahui jarak maksimal pengiriman data antar modul *wireless* NRF24L01. Jarak yang diukur adalah jarak dari setiap daya pancar modul *wireless* NRF24L01. Seperti yang ditunjukkan dalam Gambar 4.16 pengujian jarak jangkau modul *wireless* NRF24L01 Gambar 4.14 memperlihatkan pengujian jarak jangkau modul *wireless* NRF24L01. Pada pengujian ini menggunakan dua modul *wireless* NRF2401. Satu modul berfungsi sebagai pemancar dan satu modul berfungsi sebagai penerima. Modul *wireless* NRF24L01 memiliki 3 jenis daya pancar, yaitu : 0 dBm, -12 dBm. Dan -18 dBm. Pengujian jarak jangkau modul NRF24L01 diukur dengan menggunakan baud rate diatur sebesar 9600 *bit per second* (bps) . Pengukuran jarak dilakukan pada masing-masing perubahan daya dalam kondisi LOS (*Line of Sight*) antar perangkat dengan keadaan antenna mikrostrip modul *wireless* NRF24L01 lurus berhadapan 180° dengan menggunakan *Software* visual studio. Seperti yang ditunjukkan dalam Gambar 4.16.



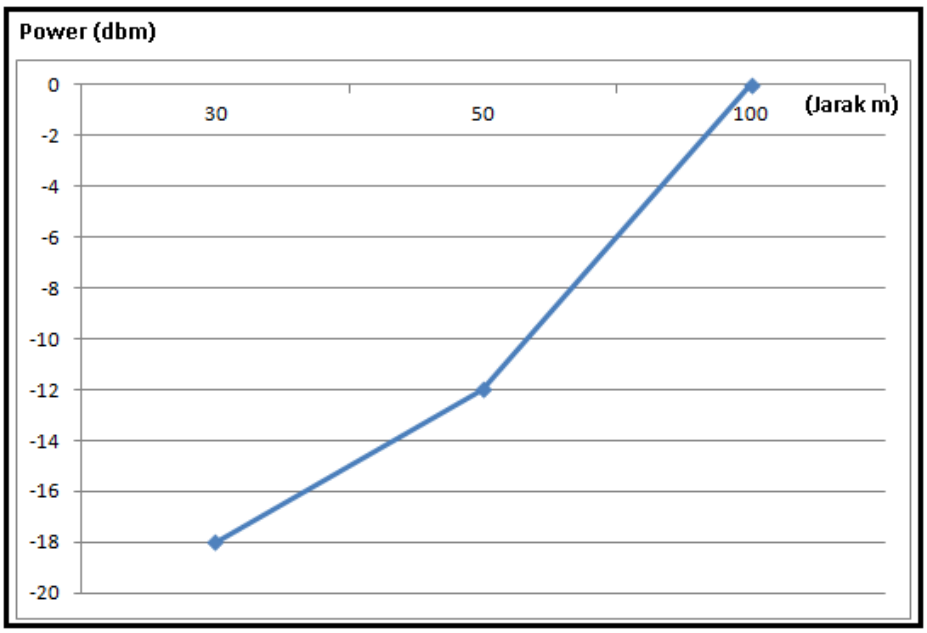
Gambar 4.16 Pengujian jarak modul *wireless* NRF24L01

Saat perangkat pengirim mengirim informasi dan tidak dapat diterima oleh penerima maka, jarak pengujian dengan merubah posisi lebih dekat dengan penerima hingga lampu indikator *gateway* menyala berkedip sebagai tanda dapat menerima data. Jika lampu indikator menyala tanpa kedip hal ini menunjukkan *gateway* mengalami kegagalan dalam mengirim data ke terminal *node*. Dari hasil pengujian dan pengukuran diperoleh data maksimal jarak transmisi. Seperti yang terlihat pada Gambar 4.16. Dari pengujian yang telah dilakukan, didapatkan jarak jangkau modul *wireless* NRF4L01 yang ditunjukkan pada Tabel 4.1.

Tabel 4.1 Hasil pengukuran jarak modul *wireless* NRF24L01

No	Daya pancar (dbm)	Jarak (m)
1	-18	30
2	-12	50
3	0	100

Dari Tabel 4.1 pengukuran jarak modul NRF24L01 dapat diplot dalam, bentuk grafik untuk menggambarkan hubungan daya pancar dengan jarak jangkauan.



Gambar 4.17 Grafik hubungan daya pancar dengan jarak jangkauan

Pada umumnya, satuan dari daya transmisi atau power adalah Watt. Namun pada implementasinya, satuan yang digunakan adalah dBm. Agar terlihat pola dari setiap perubahan daya maka nilai dBm tersebut dikonversi menjadi satuan mW (miliWatt). Proses konversi daya dalam satuan dBm menjadi satuan mW dihitung melalui rumus persamaan berikut.

P(dBm) : 10 log ($P.mWatt\ mW$).....(1)

Keterangan persamaan : P(dBm) : Daya transmisi dalam satuan decibel (dBm)

P.mWatt : Daya transmisi dalam satuan Watt

mW : Satuan daya dalam mili Watt

Hasil perhitungan konversi daya modul *wireless* NRF24L01 dari satuan dBm menjadi satuan mW ditunjukkan dalam Tabel 4.2

Tabel 4.2 Konsumsi daya pancar modul *wireless* NRF24L01

No	Daya (dbm)	Daya (mW)
1	-18	0.015
2	-12	0.063
3	0	1

4.5.2 Pengujian Komunikasi Antar *Node* Menggunakan *Protocol* MAVLink

Pada proses pembentukan *payload Protocol* MAVLink, penulis menyusun data dengan panjang data tidak melebihi batasan maksimal paket. Pada pengujian *Protocol* MAVLink yang telah dilakukan penulis menyusun data pengujian karakter 27 pada lokasi ke 1 dan ke dua. Karakter 2 dan karakter 7 diubah dalam bentuk 8 bit data, sehingga menjadi 0x02H dan 0x07H. Sebelum data dikirimkan data *bytes* dilakukan swap 2 *bytes* data, sehingga awal nilai 0x02H

0x02H = 0000 0010 -> swap -> 0000 0100 = 0x04H

0x07H = 0000 1011 -> swap -> 0000 1101 = 0x0DH

Data 0x04H diletakkan pada *array payload* indek ke 0 dan nilai 0x0DH diletakkan pada *array payload* indek ke 1, *payload* kemudian dapat dikirimkan pengirim ke penerima.

Proses pembentukan *payload* dapat dijelaskan sebagai berikut :

1. Data *latitude* gps menempati lokasi 4 *byte* dan dilakukan swap masing-masing *byte* nya. Proses swap adalah proses menukar posisi *byte*.
   Byte ke 0 ditukar ke *byte* ke 3
   Byte ke 1 ditukar ke *byte* ke 2
   Byte ke 2 ditukar ke *byte* ke 1
   Byte ke 3 ditukar ke *byte* ke 0

Sebelum proses *swap byte* langkah awal yang dilakukan adalah melakukan konversi nilai *latitude* ke *byte array*, karena variabel *double* merupakan variabel dengan panjang data 4 *byte*, maka *latitude* dilakukan konversi ke 4 *byte array*.

Berikut kode program proses konversi dan swap data

```
mav_put_4.val = latitude;
buf[0] = mav_put_4.bytes[3];
buf[1] = mav_put_4.bytes[2];
buf[2] = mav_put_4.bytes[1];
buf[3] = mav_put_4.bytes[0];
```

2. Data *longitude* gps menempati lokasi 4 *byte* dan dilakukan swap masing-masing *byte* nya.

   Byte ke 4 ditukar ke *byte* ke 7

   Byte ke 5 ditukar ke *byte* ke 6

   Byte ke 6 ditukar ke *byte* ke 5

   Byte ke 7 ditukar ke *byte* ke 4

Sebelum proses *swap byte* langkah awal yang dilakukan adalah melakukan konversi nilai *longitude* ke *byte array*, karena variabel *double* merupakan variabel dengan panjang data 4 *byte*, maka *latitude* dilakukan konversi ke 4 *byte array*.

Berikut kode program proses konversi dan swap data

```
mav_put_4.val = longitude;
buf[4] = mav_put_4.bytes[3];
buf[5] = mav_put_4.bytes[2];
buf[6] = mav_put_4.bytes[1];
buf[7] = mav_put_4.bytes[0];
```

3. Data SOG (*Speed over ground*) gps menempati lokasi 4 *byte* dan dilakukan swap masing-masing *byte* nya.

   Byte ke 8 ditukar ke *byte* ke 11

   Byte ke 9 ditukar ke *byte* ke 10

   Byte ke 10 ditukar ke *byte* ke 9

   Byte ke 11 ditukar ke *byte* ke 8

Sebelum proses *swap byte* langkah awal yang dilakukan adalah melakukan konversi nilai SOG ke *byte array*, karena variabel *double* merupakan variabel dengan panjang data 4 *byte*, maka SOG dilakukan konversi ke 4 *byte array*.

Berikut kode program proses konversi dan swap data

```
mav_put_4.val = SOG;
buf[8] = mav_put_4.bytes[3];
buf[9] = mav_put_4.bytes[2];
buf[10] = mav_put_4.bytes[1];
buf[11] = mav_put_4.bytes[0];
```

4. Data COG (*Course over ground*) kompas menempati lokasi 4 *byte* dan dilakukan swap masing-masing *byte* nya.
   Byte ke 12 ditukar ke *byte* ke 15
   Byte ke 13 ditukar ke *byte* ke 14
   Byte ke 14 ditukar ke *byte* ke 13
   Byte ke 15 ditukar ke *byte* ke 12

Sebelum proses *swap byte* langkah awal yang dilakukan adalah melakukan konversi nilai COG ke *byte array*, karena variabel *double* merupakan variabel dengan panjang data 4 *byte*, maka COG dilakukan konversi ke 4 *byte array*.

Berikut kode program proses konversi dan swap data

```
mav_put_4.val = COG;
buf[12] = mav_put_4.bytes[3];
buf[13] = mav_put_4.bytes[2];
buf[14] = mav_put_4.bytes[1];
buf[15] = mav_put_4.bytes[0];
```

5. *Altitude* atau ketinggian terhadap permukaan air laut gps, menempati lokasi 4 *byte* dan dilakukan swap masing-masing *byte* nya.
   Byte ke 16 ditukar ke *byte* ke 19
   Byte ke 17 ditukar ke *byte* ke 18
   Byte ke 18 ditukar ke *byte* ke 17
   Byte ke 19 ditukar ke *byte* ke 16

Sebelum proses *swap byte* langkah awal yang dilakukan adalah melakukan konversi nilai *Altitude* ke *byte array*, karena variabel *double* merupakan variabel dengan panjang data 4 *byte*, maka *altitude* dilakukan konversi ke 4 *byte array*.

Berikut kode program proses konversi dan swap data

```
mav_put_4.val = altitude;
buf[16] = mav_put_4.bytes[3];
buf[17] = mav_put_4.bytes[2];
buf[18] = mav_put_4.bytes[1];
buf[19] = mav_put_4.bytes[0];
```

6. Hdop atau *horizontal delution* merupakan faktor akurasi lokasi gps terhadap bujur, menempati lokasi 4 *byte* dan dilakukan swap masing-masing *byte* nya.

   Byte ke 20 ditukar ke *byte* ke 23

   Byte ke 21 ditukar ke *byte* ke 22

   Byte ke 22 ditukar ke *byte* ke 21

   Byte ke 23 ditukar ke *byte* ke 20

Sebelum proses *swap byte* langkah awal yang dilakukan adalah melakukan konversi nilai hdop ke *byte array*, karena variabel *double* merupakan variabel dengan panjang data 4 *byte*, maka hdop dilakukan konversi ke 4 *byte array*.

Berikut kode program proses konversi dan swap data

```
mav_put_4.val = altitude;
buf[16] = mav_put_4.bytes[3];
buf[17] = mav_put_4.bytes[2];
buf[18] = mav_put_4.bytes[1];
buf[19] = mav_put_4.bytes[0];
```

7. Jumlah satelit,merupakan jumlah satelit yang dapat diterima oleh perangkat gps, menempati lokasi 1 *byte* dan tidak perlu dilakukan swap *byte*.

```
buf[24] = numSatelite;
```

8. Data temperatur memiliki panjang variabel 2 *byte* dan dilakukan swap *byte*

   Byte ke 25 ditukar ke *byte* 26

   Byte 26 ditukar ke *byte* 25

```
mav_put_2.val = (int)temperature;
 buf[25] = mav_put_4.bytes[1];
 buf[26] = mav_put_4.bytes[0];
```

9. *Pressure*, merupakan data tekanan udara, menempati lokasi 4 *byte* dan dilakukan swap masing-masing *byte* nya.

   Byte ke 27 ditukar ke *byte* ke 30

   Byte ke 28 ditukar ke *byte* ke 29

   Byte ke 29 ditukar ke *byte* ke 28

   Byte ke 30 ditukar ke *byte* ke 27

Sebelum proses *swap byte* langkah awal yang dilakukan adalah melakukan konversi nilai *pressure* ke *byte array*, karena variabel *double* merupakan variabel dengan panjang data 4 *byte*, maka *pressure* dilakukan konversi ke 4 *byte array*.

Berikut kode program proses konversi dan swap data

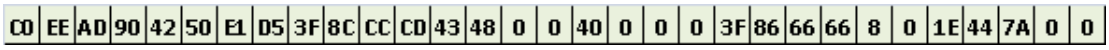
```
mav_put_4.val = pressure;
buf[27] = mav_put_4.bytes[3];
buf[28] = mav_put_4.bytes[2];
buf[29] = mav_put_4.bytes[1];
buf[30] = mav_put_4.bytes[0];
```

Sebagai contoh proses pembentukan *payload Protocol* MAVLink, dapat dijelaskan berikut :

Latitude	=	-7.4586869
Konversi ke 4 *byte*	=	0x 90 AD EE C0 H
Swap *byte*	=	0x C0 EE AD 90 H
Longitude	=	112.657878
Konversi ke 4 *byte*	=	0x D5 50 E1 42 H
Swap *byte*	=	0x 42 50 E1 D5 H
SOG	=	1.1 km/jam
Konversi ke 4 *byte*	=	0x CD CC 8C 3F H
Swap *byte*	=	0x 3F 8C CC CD H
COG/kompas	=	200 deg
Konversi ke 4 *byte*	=	0x 0 0 48 43 H
Swap *byte*	=	0x 43 48 0 0 H
Altitude	=	2.0 meter
Konversi ke 4 *byte*	=	0x 0 0 0 40 H
Swap *byte*	=	0x 40 0 0 0 H
Hdop	=	1.05
Konversi ke 4 *byte*	=	0x 66 66 86 3F H
Swap *byte*	=	0x 3F 86 66 66 H
Satelite	=	8
Konversi ke 1 *byte*	=	0x 08 H

Temperatur	=	30 deg
Konversi ke 2 *byte*	=	0x 1E 0 H
Swap *byte*	=	0x 0 1E H
Tekanan	=	1000 mBar
Konversi ke 4 *byte*	=	0x 0 0 7A 44 H
Swap *byte*	=	0x 44 7A 0 0 H

Payload yang terbentuk berdasarkan *bytes* data yang diperoleh sebagai berikut :



Pada sisi penerima akan melakukan proses *decode* data MAVLink dengan proses kebalikan dari sisi pengirim. Data yang diperoleh dari *payload* penerima dilakukan proses swap *bytes* untuk mendapatkan data informasi asli. Proses diawali untuk mendapatkan data *payload*, menyusun data dan melakukan *swap byte* data dan melakukan konversi dari *array byte* ke variabel *float* maupun *double*.

Berikut proses decode *Protocol* MAVLink pada penerima :

Sebagai contoh proses pembentukan *payload Protocol* MAVLink, dapat dijelaskan berikut :

Data 4 *byte*	=	0x C0 EE AD 90 H
Swap 4 *byte*	=	0x 90 AD EE C0 H

Konversi 4 *byte* ke *double*

Latitude	=	-7.4586869
Data 4 *byte*	=	0x 42 50 E1 D5 H
Swap 4 *byte*	=	0x D5 50 E1 42 H

Konversi 4 *byte* ke *double*

Longitude	=	112.657878
Data *byte*	=	0x 3F 8C CC CD H
Swap 4 *byte*	=	0x CD CC 8C 3F H

Konversi 4 *byte* ke float

SOG	=	1.1 km/jam
Data 4 *byte*	=	0x 43 48 0 0 H

Swap 4 *byte*	=	0x 0 0 48 43 H
Konversi 4 *byte* ke float		
COG/kompas	=	200 deg
Data 4 *byte*	=	0x 40 0 0 0 H
Swap 4 *byte*	=	0x 0 0 0 40 H
Konversi 4 *byte* ke float		
Altitude	=	2.0 meter
Data *byte*	=	0x 3F 86 66 66 H
Swap 4 *byte*	=	0x 66 66 86 3F H
Konversi 4 *byte* ke float		
Hdop	=	1.05
Data *byte*	=	0x08 H
Konversi ke karakter		
Satelite	=	8
Data 2 *byte*	=	0x 0 1E H
Swap 2 *byte*	=	0x 1E 0 H
Konversi 2 *byte* ke integer		
Temperatur	=	30 deg
Data 4 *byte*	=	0x 44 7A 0 0 H
Swap 4 *byte*	=	0x 0 0 7A 44 H
Konversi 4 *byte* ke float		
Tekanan	=	1000 mBar

1) Pengujian 1

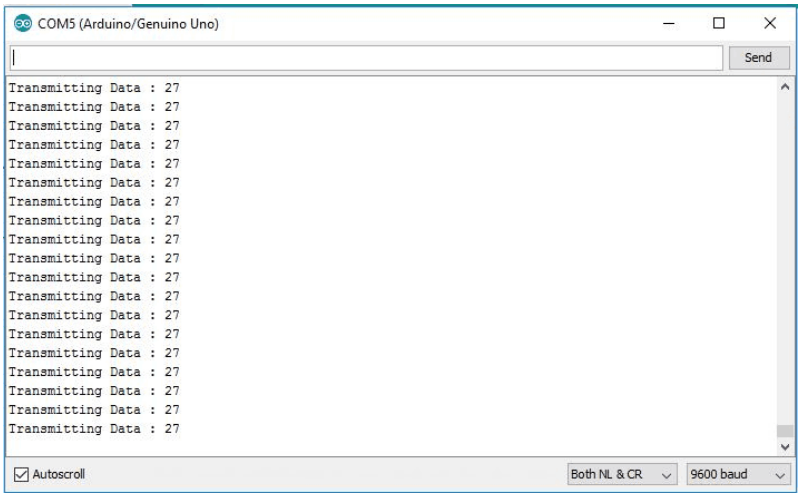
Pengujian komunikasi antar *node* dilakukan untuk mengetahui keberhasilan komunikasi antar *node*. Dengan menggunakan metode pengujian *point to point*. Pengujian *point to point* merupakan pengujian yang dimana 1 *node* menjadi pemancar dan 1 *node* menjadi penerima. Pengujian komunikasi antar *node* diatur dengan menggunakan *baudrate* diatur sebesar 9600 bps. Keberhasilan komunikasi *point to point* ditunjukkan pada Gambar 4.18dan Gambar 4.19. Dari

pengujian yang telah dilakukan, dinyatakan berhasil melakukan komunikasi *point to point* dinyatakan dengan diperoleh data “27” pada serial Monitor arduino ide. Sedangkan kegagalan melakukan komunikasi *point to point* dinyatakan dengan informasi “*failed*” diserial Monitor arduino ide.

Pengujian 1 dilakukan dengan menempatkan *Source Node* 02 pada jarak sekitar 25.4 meter dari *Master Node* atau *Node* tujuan.

Gambar 4.18 Pengujian dengan jarak 25.4 meter menggunakan protokol MAVLink

Bisa dilihat pada gambar 4.16 pengujian pertama yang dilakukan peneliti dengan jarak 25.4 meter dengan menggunakan protokol MAVLink. Pengujian dilakukan di alun-alun sidoarjo pada saat malam hari dengan menggunakan daya -18 dBm dapat dilihat bahwa node masih bisa terdeteksi oleh *gateway* dengan membuktikan posisi gps.

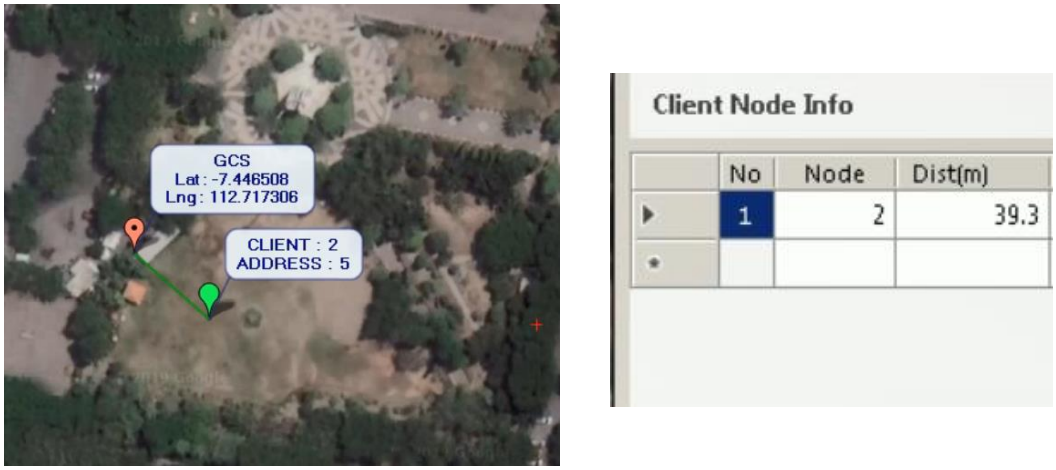


Gambar 4.19 Pengujian pengiriman data *point to point*

Seperti yang ditunjukkan dalam Gambar 4.19. Pada pengujian ini menggunakan jarak antar 1 *node* pemancar dan 1 *node* penerima adalah 25.4 meter dengan menggunakan daya pancar -18 dBm. Sedangkan daya pancar -18 dBm memiliki jarak maksimum 30 meter. Jadi daya pancar -18 dBm masih bisa melakukan komunikasi antar *node*.

2) Pengujian 2

Pengujian 2 dilakukan dengan menempatkan *Source Node* 02 pada jarak sekitar 39.3 meter dari *Master Node* atau *Node* tujuan. Seperti pada pengujian I lokasi *Master Node* bersifat statis sedangkat *Source Node* bersifat *mobile*. Karena pada penelitian ini *Source Node* dapat bergerak dan berpindah-pindah. Hasil pengujian 2 disajikan dalam bentuk *capture* gambar pada aplikasi *Ground station*, seperti yang terlihat pada gambar hasil pengujian 4.20 berikut.



Gambar 4.20 Pengujian dengan jarak 39.3 meter menggunakan protokol MAVLink

Bisa dilihat pada gambar 4.20 pengujian kedua yang dilakukan peneliti dengan jarak 39.3 meter dengan menggunakan protokol MAVLink. Pengujian dilakukan di alun-alun sidoarjo pada saat malam hari dengan menggunakan daya -18 dBm dapat dilihat bahwa node tidak bisa terdeteksi oleh *gateway* dengan membuktikan posisi gps.

Seperti yang ditunjukkan dalam Gambar 4.21. Pada pengujian ini menggunakan jarak antar 1 *node* pemancar dan 1 *node* penerima adalah 34.5 meter dengan menggunakan daya pancar -18 dBm. Sedangkan daya pancar -18 dBm memiliki jarak maksimum 30 meter. Jadi jarak 34.5 meter dengan menggunakan daya pancar -18 dBm tidak bisa melakukan komunikasi antar *node* karena melebihi jarak maksimum dari daya pancar.

Gambar 4.21 Informasi kegagalan pengiriman data *point to point*

4.5.3 Pengujian Komunikasi Antar *Node* Menggunakan *Protocol* AIS

AIS (*Automatic Identification System*) merupakan sistem pelacakan otomatis yang digunakan pada kapal dan sebagai layanan lalu lintas kapal (VTS) untuk mengidentifikasi dan menemukan kapal secara elektronik bertukar data dengan kapal terdekat lainnya, stasiun pangkalan AIS, dan satelit. Interval pelaporan AIS bervariasi tergantung pada jenis pesan AIS dan juga dapat berubah untuk jenis pesan tertentu itu sendiri tergantung pada kondisi kapal seperti ditambatkan. Data AIS yang dipertukarkan dibagi menjadi tiga jenis:

1. Data statis (mis., Nama kapal, jenis kapal, dan dimensi kapal)
2. Data dinamis (mis., Posisi kapal, jalur darat, dan heading)
3. Data terkait pelayaran (mis. Konsep saat ini, deskripsi kargo, dan tujuan).

Sehubungan dengan data statis, informasi jenis kapal termasuk dalam jenis pesan AIS lima dan disimpan dalam *bytes*, *bytes* ini memiliki nilai yang valid kisaran dalam 0-99. Kapal dapat secara langsung diidentifikasi dengan menggunakan nomor MMSI (*Maritime Mobile Service Identity*) yang bersifat unik.

Pada pengujian penulis mendeskripsikan jaringan yang terdiri dari *Node* dan Master *Node*, *protocol* AIS diterapkan pada *Client Node* yang diberi nomer 02. Untuk membangkitkan AIS *messages* pada perangkat *node* 02, format pesan disusun dengan konfigurasi berikut :

1. *Header* : "!AIVDM,", identifikasi format AIS *Messages*
2. *Messages count* : Total pesan
3. *Messages number* : Nomer pesan, jika *message_count=2 message number* bisa bernilai 1 atau 2.
4. *Sequence id* : Nomer pesan saat ini
5. *Channel* : A/B
6. *Payload* : AIS Data
7. Akhiran pesan : 0*
8. *Checksum* : 8-bit ekslusif OR dari data

Sebelum dilakukan pengujian, pesan AIS disusun pada perangkat *node* 02 sebagai berikut :

Ais messages = Header + Messages count + Messages number + Sequence id + Channel + Payload + Akhiran pesan + *Checksum*

Pada program Arduino yang dibuat penulis, dapat dijabarkan pada potongan kode program pada gambar 4.19. AIS data yang di bangkitkan sebagai *payload*, disusun dengan langkah-langkah sebagai berikut :

1. Melakukan konversi data ke 6 bit.
2. Melakukan konversi data 6 bit ke 8 bit.
3. Menyusun data yang terdiri dari

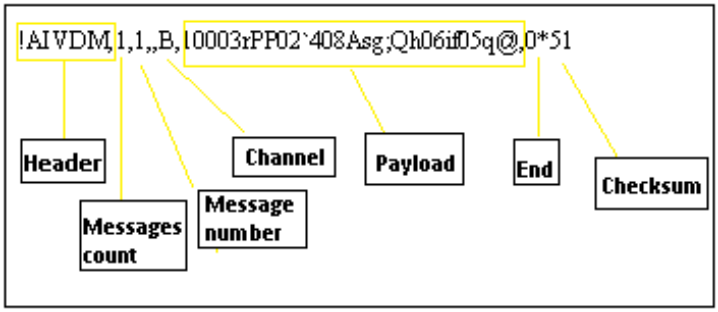
   Header + Messages count + Messages number + Sequence id + Channel + *Payload*

   Sehingga akan diperoleh format data seperti contoh data berikut

   !AIVDM,1,1,,B,10003rPP02`408Asg;Qh06if05q@,
4. Menghitung nilai *cheksum*, dengan proses XOR semua bit data pada proses nomer 3.
5. Menambahkan akhiran *0 sebelum menambahkan nilai *checksum*, sehingga format AIS secara lengkap sebagai berikut.

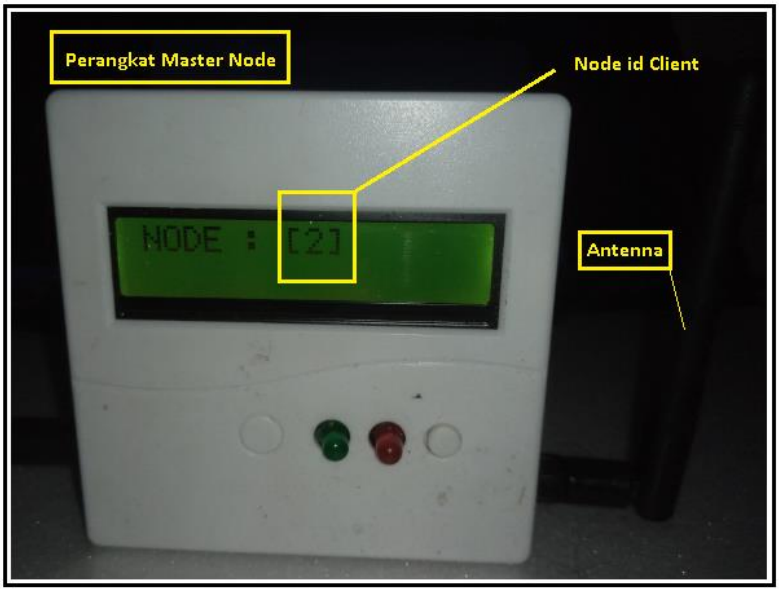
   !AIVDM,1,1,,B,10003rPP02`408Asg;Qh06if05q@,0*51

Dari hasil akhir proses encode Ais messages dapat dijelaskan pada gambar 4.22 berikut.



Gambar 4.22 Hasil akhir proses *encode* AIS *Messages*

Dari hasil pengujian yang dilakukan, tahap awal yang perlu diketahui apakah *Node* 02 telah terhubung dengan *Master Node* 00, dapat dilihat gambar 4.23 berikut.



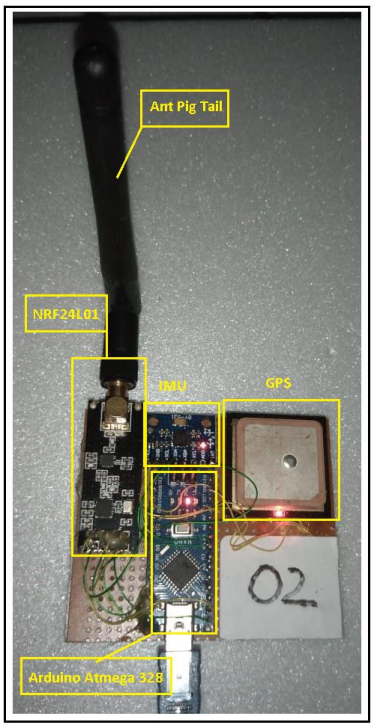
Gambar 4.23 *Source Node* 02 telah terhubung dengan Master *Node*

Gambar 4.23 menampilkan alat dari *gateway* yang bermaksud bahwa node 2 sudah terhubung ke *gateway*. Gambar 4.23 menunjukan alat dari node 2 atau terminal.

Untuk mendapatkan *capture* data yang diterima oleh *node* tujuan atau *Master Node*, dilakukan proses pengambilan data melalui serial komunikasi TTL pada perangkat Master *Node* melalui serial Monitor Arduino Ide maupun dapat dilakukan menggunakan serial Monitor program Hterm. Pada pengujian yang dilakukan penulis secara khusus membuat sebuah aplikasi berbasis Windows menggunakan Visual Studio 2012 sebagai *Ground station* Monitor. Pada program aplikasi yang dibuat memiliki fitur untuk melakukan pengambilan data yang dikirim oleh *Source Node* 02 ke *Master Node* melalui komunikasi USB, selain itu program aplikasi yang dibuat dapat melakukan decode data AIS untuk mendapatkan pesan asli yang terkandung pada data AIS. Secara lengkap dapat dijelaskan fitur yang disediakan aplikasi yang dibuat, yaitu :

1. Dapat melakukan komunikasi melalui serial port dengan perangkat *Master Node*.

2. Dapat melukan *encode* data Ais.
3. Dapat menampilkan peta elektronik lokasi *Source Node*.
4. Dapat melakukan log data dalam bentuk tabel.

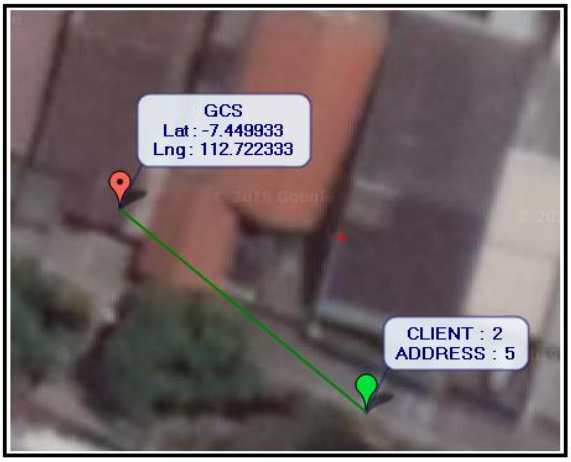


Gambar 4.24 Perangkat *Node* 02 Sebagai *Source Node*

1) Pengujian 1

Pengujian I dilakukan dengan menempatkan *Source Node* 02 pada jarak sekitar 20 meter dari *Master Node* atau *Node* tujuan. Pengukuran jarak dapat dilakukan dengan menghitung jarak dua lokasi menggunakan GPS. Pada *Source Node* 02 diatur interval waktu pengiriman data sebesar 3000 ms, yang berarti setiap 3000ms maka *Source Node* 02 akan melakukan *update* data dengan mengirimakan data ke *Node* tujuan atau *Master Node*. Data yang dikirm merupakan data *protocol* AIS dengan jumlah paket 47 *byte*. Pada pengujian yang dilakukan akan dilakukan analisa dan pengamatan untuk melihat apakah data yang diterima sesuai dengan data yang dikirim dan apakah terjadi kehilangan paket pada saat proses pengiriman data. Dengan mengatur jarak transmisi dapat dilihat apakah data yang diterima tetap utuh.

Pada gambar 4.25 merupakan hasil *capture* pengujian pertama pada jarak 20.4 meter peta elektronik yang memperlihatkan lokasi *Ground station* atau lokasi *Destination Node* dengan *Source Node*.



Gambar 4.25 Lokasi Perangkat *Source Node* Sumber 02 dan lokasi *Ground station* atau *Node* tujuan

Client Node Info

	No	Node	Dist(m)
▶	1	2	20.4
*			

Gambar 4.26 Jarak *Node* Sumber 02 dengan *Ground station* atau *Node* tujuan

Dari hasil pengujian ini diperoleh data hasil *decode* AIS Messages yang ditunjukkan pada gambar 4.27.

Decode Ais Messages

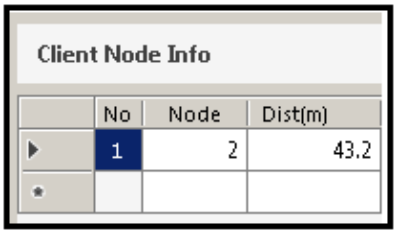
Message Type	[1] Scheduled Position Report
User ID	1002
Navigation Status	[0] Under way using engine
SOG	2.8 km/h
COG	35.2 °
True heading	48 °
Latitude	-7.450022
Longitude	112.722453

Gambar 4.27 Hasil Decode AIS Messages

2) Pengujian 2

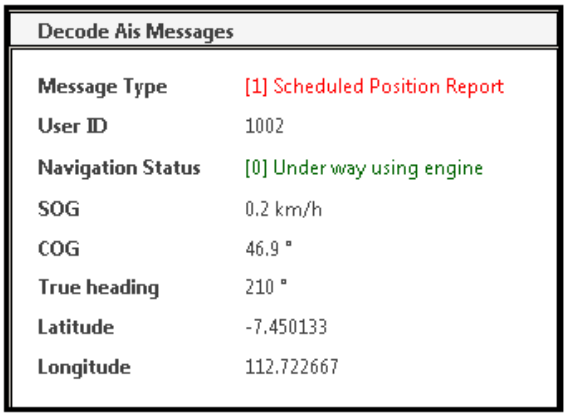
Pengujian II dilakukan dengan menempatkan *Source Node* 02 pada jarak sekitar 43.2 meter dari *Master Node* atau *Node* tujuan.

Gambar 4.28 Lokasi Perangkat *Source Node* Sumber 02 dan lokasi *Ground station* atau *Node* tujuan



Gambar 4.29 Jarak *Node* Sumber 02 dengan *Ground station* atau *Node* tujuan

Dari hasil pengujian ini diperoleh data hasil *decode* AIS Messages yang ditunjukkan pada gambar 4.30.



Gambar 4.30 Hasil Decode AIS Messages

4.5.4 Pengujian Perfoma Jaringan *Mesh*

Pengujian performa jaringan dilakukan untuk mengetahui seberapa handal jaringan dalam melakukan pertukaran data antar *node*. Parameter pengujian yang dilakukan penulis adalah :

1. *Throughput*

   Throughput dinyatakan sebagai *volume* data yang berhasil dikirim dalam satuan waktu. Merupakan ukuran seberapa cepat atau lambat jaringan yang diukur.

2. *Packet loss*

   Merupakan pengukuran seberapa banyak paket yang hilang pada proses pengiriman data.

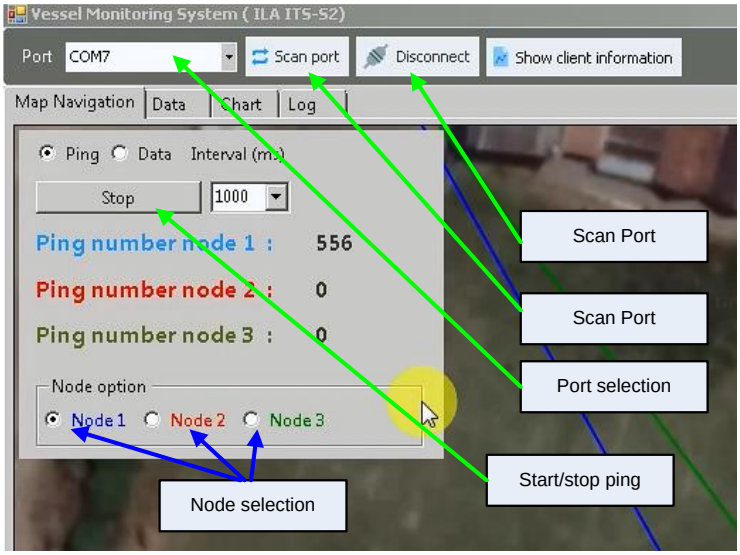
3. *End to End Delay*

   Merupakan pengukuran interval waktu yang dibutuhkan untuk mengirimkan data dari pengirim ke penerima.

Proses pengujian *Quality Of Service* (QOS) jaringan yang dibangun menggunakan perangkat nirkabel NRF24L01, dilakukan dengan mengirimkan *payload* sebesar 32 *byte* ke *node client* yang dilakukan pengujian. Jumlah *packet* yang dikirimkan tidak dibatasi, sehingga faktor lama waktu pengujian merupakan parameter pengujian *Quality Of Service* (QOS). Langkah-langkah pengujian *Quality Of Service* (QOS) pada jaringan nirkabel NRF24L01 dapat dijelaskan sebagai berikut:

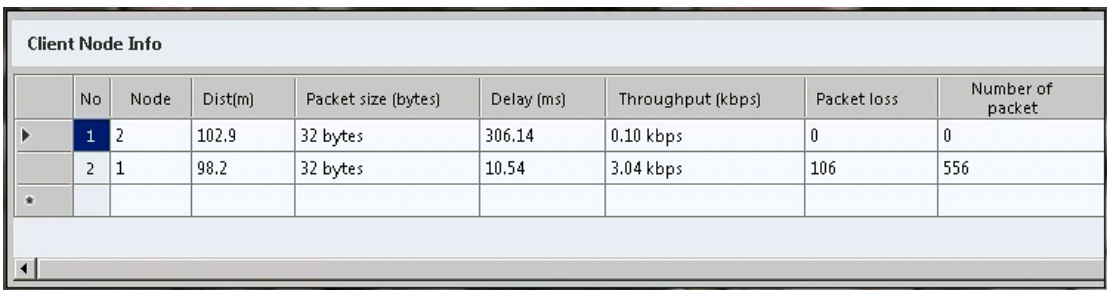
1. Menjalankan aplikasi Monitoring yang dibuat menggunakanVisual Studio 2012.
2. Menghubungkan perangkat *gateway* ke perangkat laptop.
3. Melakukan *scan* port pada aplikasi.
4. Menentukan port usb *gateway*.
5. Menghubungkan aplikasi dengan *gateway*.
6. Menyalakan setiap *node*.
7. Melakukan *scan node* yang terhubung ke jaringan melalui aplikasi.
8. Melakukan *ping* ke *node* yang diuji.
9. Menampilkan informasi *node* pada aplikasi.

Gambar 4.31 berikut merupakan gambar-gambar langkah-langkah pengujian QOS (*Quality Of Service*).



Gambar 4.31 Langkah-langkah pengujian QOS (*Quality Of Service*).

Untuk mendapatkan hasil pengukuran dapat ditampilkan pada antar muka aplikasi untuk mendapatkan informasi QOS (*Quality Of Service*). Tampilan antar muka program aplikasi dapat ditunjukkan pada Gambar 4.31. Pada pengujian yang dilakukan pada *node* 1 dengan jarak transmisi 98.2 meter. Pada tampilan antar muka aplikasi ditunjukkan pengukuran *delay* dalam ms, *throughput* dalam kbps dan *packet loss*. Interval ping diatur pada 1000 ms, sehingga ping ke *node* 1 dilakukan tiap 1 detik.



Client Node Info

No	Node	Dist(m)	Packet size (bytes)	Delay (ms)	Throughput (kbps)	Packet loss	Number of packet
1	2	102.9	32 bytes	306.14	0.10 kbps	0	0
2	1	98.2	32 bytes	10.54	3.04 kbps	106	556

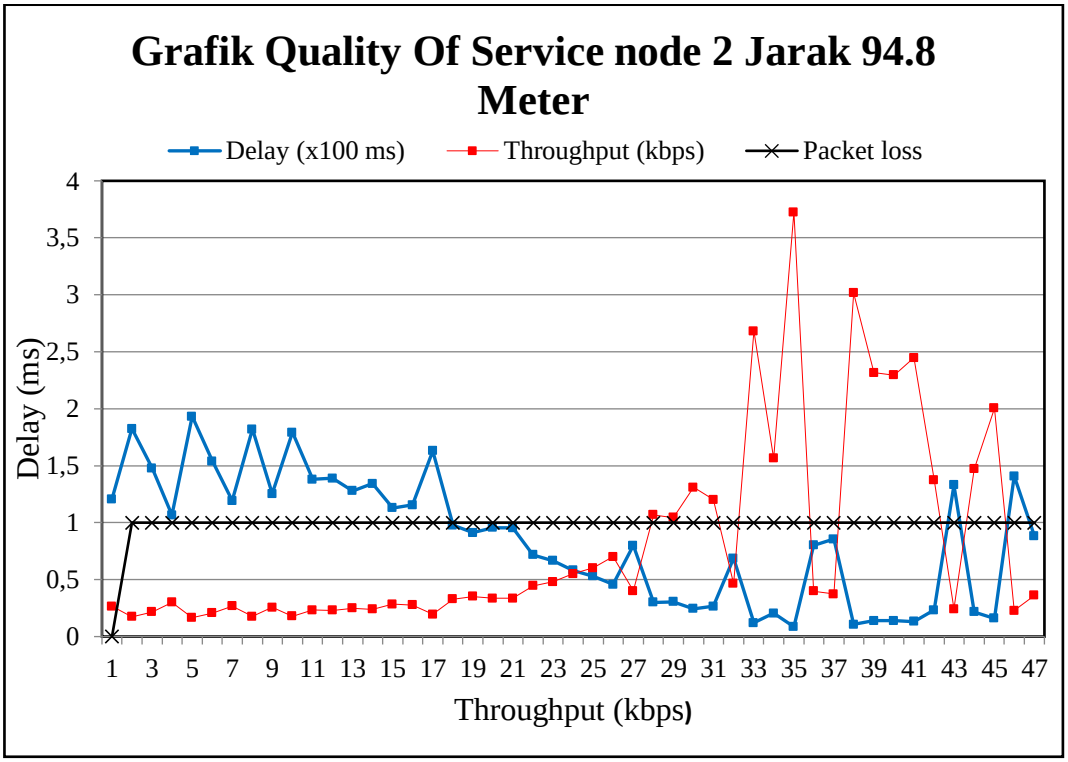
Gambar 4.32 Tampilan antar muka pengukuran QOS (*Quality Of Service*).

Pada tabel pengujian berikut merupakan hasil pengukuran yang dilakukan penulis dalam beberapa variasi jarak transmisi.

Tabel 4.3 Pengukuran Performa Jaringan node 2 Pada Jarak 94.8 meter

No	*Packet size*	*Delay* (ms)	*Throughput* (kbps)	*Packet loss*	*Packet send*	*Packet success*
1	32	120.72	0.265	0	6	6
2	32	182.5	0.175	1	9	8
3	32	147.792	0.217	1	10	9
4	32	106.514	0.3	1	11	10
5	32	193.294	0.166	1	13	12
6	32	153.902	0.208	1	15	14
7	32	119.336	0.268	1	16	15
8	32	181.876	0.176	1	17	16
9	32	125.328	0.255	1	18	17
10	32	179.102	0.179	1	19	18
11	32	137.972	0.232	1	20	19
12	32	138.986	0.23	1	21	20
13	32	127.896	0.25	1	22	21
14	32	134.166	0.239	1	24	23
15	32	112.968	0.283	1	25	24
16	32	115.608	0.277	1	26	25
17	32	163.124	0.196	1	27	26
18	32	97.638	0.328	1	28	27
19	32	91.216	0.351	1	29	28
20	32	95.51	0.335	1	30	29

Pengukuran QOS (*Quality Of Service*) dilakukan dengan mengukur sejumlah paket data sebanyak 47 paket dengan *payload 32 byte* yang dikirim oleh *gateway* ke *node*. Interval pengiriman data dari *gateway* ke *node* uji diatur sebesar 1000 ms. Pengukuran QOS dilakukan pada jarak antara *gateway* dengan *node* uji sebesar 94.8 meter. Proses pengukuran dilakukan oleh aplikasi yang dibuat penulis menggunakan Visual Studio 2012. Informasi data pengukuran *throughput*, *delay* dan *packet loss* diperoleh dari perangkat *gateway* yang dikirimkan ke perangkat pc atau laptop melalui komunikasi serial. Berdasarkan hasil pengukuran pada Tabel 4.3 dapat dibuat grafik untuk mem-*plot* data pengukuran QOS dalam bentuk grafik, yang ditunjukkan pada Gambar 4.33 berikut.



Gambar 4.33 Grafik QOS (*Quality Of Service*) node 2 jarak 94.8 meter.

Pada gambar 4.33 pengukuran QOS pada node 2 dengan jarak 94.8 meter diperoleh data dari perangkat *gateway* yang dikirimkan ke perangkat pc melalui komunikasi serial pada tabel 4.3 menunjukan hasil delay 182.5 ms, *Throughput* 0.175 kbps dan *packet loss* sebesar 1.

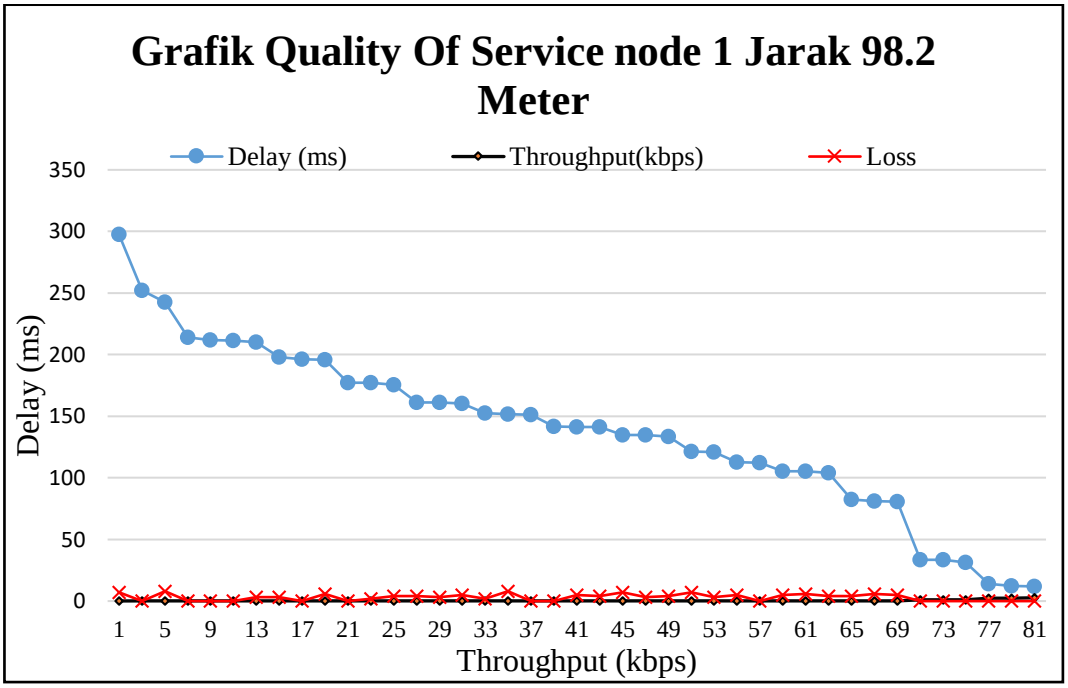
Pada tabel pengujian berikut merupakan hasil pengukuran yang dilakukan penulis dalam beberapa variasi jarak transmisi, dengan menggunakan node 1 pada jarak 98.2 meter.

Tabel 4.4 Pengukuran Performa Jaringan node 1 Pada Jarak 98.2 meter

No	*Packet size*	*Delay* (ms)	*Throughput* (kbps)	*Packet loss*	*Packet send*	*Packet success*
1.	32	297,442	0,11	7	75	68
2.	32	252,274	0,13	0	0	0
3.	32	242,65	0,14	8	84	76
4.	32	239,38	0,16	1	38	37
5.	32	238,624	0,17	4	42	38
6.	32	235,66	0,19	1	13	12
7.	32	235,342	0,23	0	5	5

Tabel lanjutan						
8.	32	232,734	0,24	7	77	70
9.	32	231,376	0,26	8	87	79
10.	32	225,492	0,28	7	76	69
11.	32	224,832	0,29	4	30	26
12.	32	221,676	0,30	0	14	14
13.	32	221,184	0,35	0	33	33
14.	32	219,338	0,39	4	61	57
15.	32	214,064	0,40	0	25	25
16.	32	211,932	0,52	0	0	0
17.	32	211,37	0,59	0	27	27
18.	32	210,246	0,62	3	29	26
19.	32	207,998	0,67	1	36	35
20.	32	205,46	0,65	7	79	72

Pengukuran QOS (*Quality Of Service*) dilakukan dengan mengukur sejumlah paket data sebanyak 47 paket dengan *payload 32 byte* yang dikirim oleh *gateway* ke *node*. Interval pengiriman data dari *gateway* ke *node* uji diatur sebesar 1000 ms. Pengukuran QOS dilakukan pada jarak antara *gateway* dengan *node* uji sebesar 98.2 meter. Proses pengukuran dilakukan oleh aplikasi yang dibuat penulis menggunakan Visual Studio 2012. Informasi data pengukuran *throughput*, *delay* dan *packet loss* diperoleh dari perangkat *gateway* yang dikirimkan ke perangkat pc atau laptop melalui komunikasi serial. Berdasarkan hasil pengukuran pada Tabel 4.4 dapat dibuat grafik untuk mem-*plot* data pengukuran QOS dalam bentuk grafik, yang ditunjukkan pada Gambar 4.34 berikut



Gambar 4.34 Grafik QOS (Quality Of Service) node 1 jarak 98.2 meter.

Pada gambar 4.34 pengukuran QOS pada node 1 dengan jarak 94.8 meter diperoleh data dari perangkat *gateway* yang dikirimkan ke perangkat pc melalui komunikasi serial pada tabel 4.4 menunjukan hasil delay 297.442 ms, *Throughput* 0.11 kbps dan *packet loss* sebesar 7.

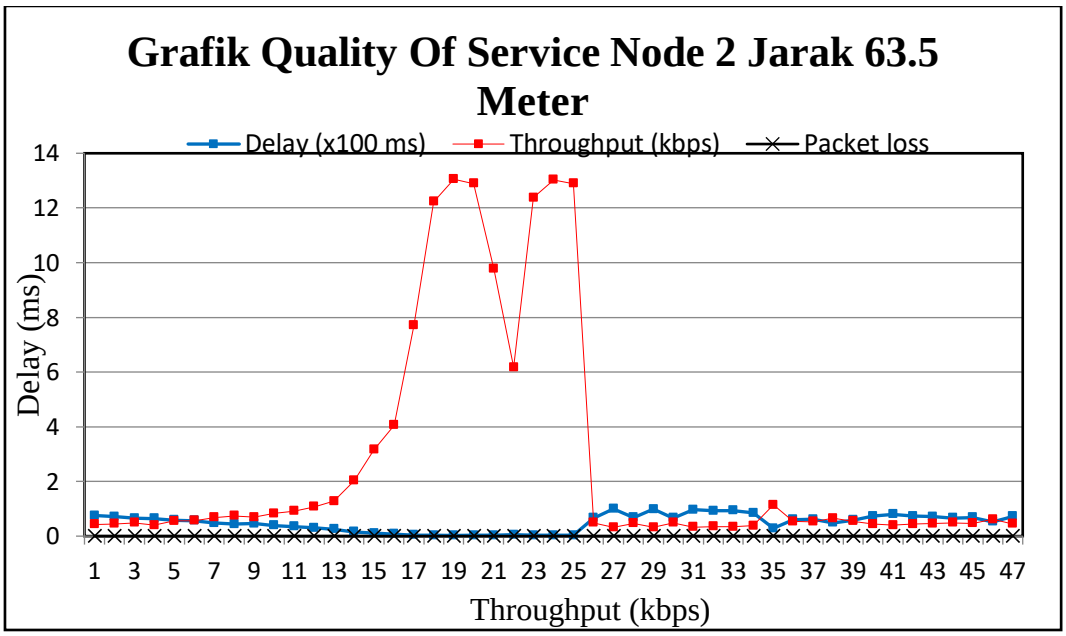
Pada tabel pengujian berikut merupakan hasil pengukuran yang dilakukan penulis dalam beberapa variasi jarak transmisi, dengan menggunakan node 2 pada jarak 63.5 meter.

Tabel 4.5 Pengukuran Performa Jaringan node 2 Pada Jarak 63.5 meter

No	*Packet* size	*Delay* (ms)	*Throughput*(kbps)	Loss	*Packet* send	*Packet* success
1	32	74.796	0.43	0	1	1
2	32	70.914	0.45	0	2	2
3	32	65.668	0.49	0	3	3
4	32	63.81	0.40	0	4	4
5	32	58.202	0.55	0	5	5
6	32	56.588	0.57	0	6	6
7	32	47.192	0.68	0	7	7
8	32	43.504	0.74	0	8	8
9	32	46.106	0.69	0	9	9
10	32	38.854	0.82	0	10	10

Tabel lanjutan						
11	32	35.112	0.91	0	11	11
12	32	29.854	1.07	0	12	12
13	32	25.088	1.28	0	13	13
14	32	15.822	2.02	0	14	14
15	32	10.134	3.16	0	15	15
16	32	7.884	4.06	0	16	16
17	32	4.152	7.71	0	17	17
18	32	2.616	12.23	0	18	18
19	32	2.452	13.05	0	19	19
20	32	2.482	12.89	0	20	20

Pada pengujian pengukuran QOS (*Quality Of Service*) jarak 63.5 meter dilakukan dengan mengukur sejumlah paket data sebanyak 47 paket dengan *payload 32 byte* yang dikirim oleh *gateway* ke *node*. Interval pengiriman data dari *gateway* ke *node* uji diatur sebesar 1000 ms. Proses pengukuran dilakukan oleh aplikasi yang dibuat penulis menggunakan Visual Studio 2012. Informasi data pengukuran *throughput*, *delay* dan *packet loss* diperoleh dari perangkat *gateway* yang dikirimkan ke perangkat pc atau laptop melalui komunikasi serial. Berdasarkan hasil pengukuran pada Tabel 4.5 dapat dibuat grafik untuk mem-*plot* data pengukuran QOS dalam bentuk grafik, yang ditunjukkan pada Gambar 4.35 berikut.



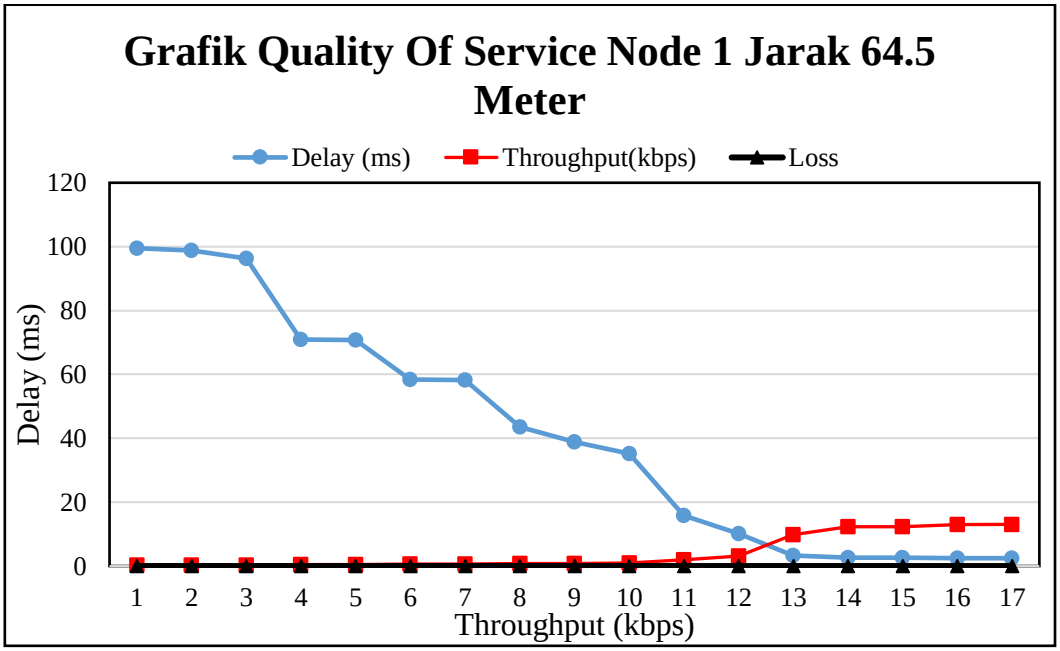
Gambar 4.35 Grafik QOS (*Quality Of Service*) jarak 63.5 meter.

Pada gambar 4.35 pengukuran QOS pada node 2 dengan jarak 94.8 meter diperoleh data dari perangkat *gateway* yang dikirimkan ke perangkat pc melalui komunikasi serial pada tabel 4.5 menunjukan hasil delay 74.796 ms, *Throughput* 0.43 kbps dan *packet loss* sebesar 0.

Pada tabel pengujian berikut merupakan hasil pengukuran yang dilakukan penulis dalam beberapa variasi jarak transmisi, dengan menggunakan node 1 pada jarak 64.5 meter.

Tabel 4.6 Pengukuran Performa Jaringan node 1 Pada Jarak 64.5 meter

No	Packet size	Delay (ms)	Throughput(kbps)	Loss	Packet send	Packet success
1.	32	31,96	1,001252	0	3	3
2.	32	27,074	1,181946	1	34	33
3.	32	21,856	1,464129	1	32	31
4.	32	20,256	1,579779	1	4	3
5.	32	19,73	1,621896	1	33	32
6.	32	19,504	1,640689	1	26	25
7.	32	19,48	1,64271	1	35	34
8.	32	17,754	1,802411	1	24	23
9.	32	17,678	1,81016	1	29	28
10.	32	17,598	1,818388	1	28	27
11.	32	17,438	1,835073	1	27	26
12.	32	17,386	1,840561	1	25	24
13.	32	17,226	1,857657	1	30	29
14.	32	17,172	1,863499	1	31	30
15.	32	16,272	1,966568	1	17	16
16.	32	12,958	2,469517	1	8	7
17.	32	12,238	2,614806	1	5	4
18.	32	11,292	2,833865	1	15	14
19.	32	10,358	3,089399	1	13	12
20.	32	10,054	3,182813	1	21	20



Gambar 4.36 Grafik QOS (Quality Of Service) jarak 64.5 meter

Pada gambar 4.36 pengukuran QOS pada node 1 dengan jarak 64.5 meter diperoleh data dari perangkat *gateway* yang dikirimkan ke perangkat pc melalui komunikasi serial pada tabel 4.6 menunjukan hasil delay 31.96 ms, *Throughput* 1.01 kbps dan *packet loss* sebesar 0.

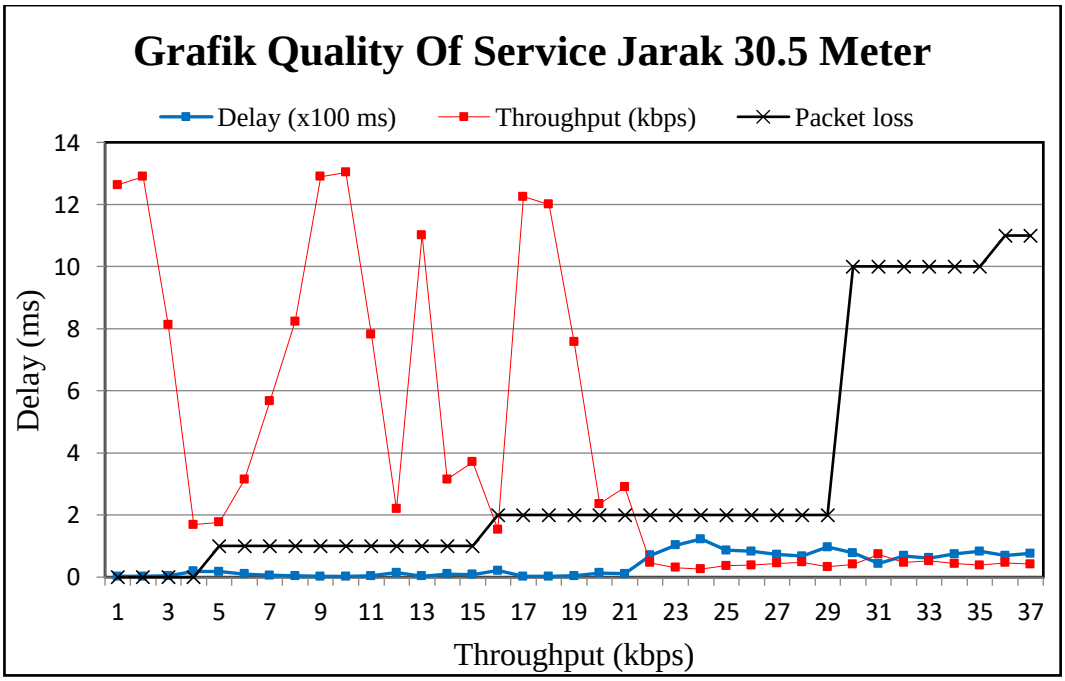
Pada tabel pengujian berikut merupakan hasil pengukuran yang dilakukan penulis dalam beberapa variasi jarak transmisi, dengan menggunakan node 2 pada jarak 30.5 meter.

Tabel 4.7 Pengukuran Performa Jaringan Pada Jarak 30.5 meter

No	*Packet* size	*Delay* (ms)	*Throughput*(kbps)	Loss	*Packet* send	*Packet* success
1	32	2.534	12.63	0	1	1
2	32	2.482	12.89	0	2	2
3	32	3.936	8.13	0	3	3
4	32	18.904	1.69	0	4	4
5	32	18.15	1.76	1	6	5
6	32	10.186	3.14	1	7	6
7	32	5.64	5.67	1	8	7
8	32	3.886	8.23	1	9	8
9	32	2.482	12.89	1	10	9
10	32	2.456	13.03	1	11	10
11	32	4.098	7.81	1	12	11

Tabel lanjutan						
12	32	14.58	2.19	1	13	12
13	32	2.904	11.02	1	14	13
14	32	10.15	3.15	1	15	14
15	32	8.632	3.71	1	16	15
16	32	20.844	1.54	2	17	15
17	32	2.612	12.25	2	18	16
18	32	2.666	12.00	2	19	17
19	32	4.228	7.57	2	20	18
20	32	13.546	2.36	2	21	19

Pada pengujian pengukuran QOS (*Quality Of Service*) jarak 30.5 meter dilakukan dengan mengukur sejumlah paket data sebanyak 47 paket dengan *payload 32 byte* yang dikirim oleh *gateway* ke *node*. Interval pengiriman data dari *gateway* ke *node* uji diatur sebesar 1000 ms. Proses pengukuran dilakukan oleh aplikasi yang dibuat penulis menggunakan Visual Studio 2012. Informasi data pengukuran *throughput*, *delay* dan *packet loss* diperoleh dari perangkat *gateway* yang dikirimkan ke perangkat pc atau laptop melalui komunikasi serial. Berdasarkan hasil pengukuran pada Tabel 4.7 dapat dibuat grafik untuk mem-*plot* data pengukuran QOS dalam bentuk grafik, yang ditunjukkan pada Gambar 4.37 berikut.



Gambar 4.37 Grafik QOS (*Quality Of Service*) jarak 30.5 meter

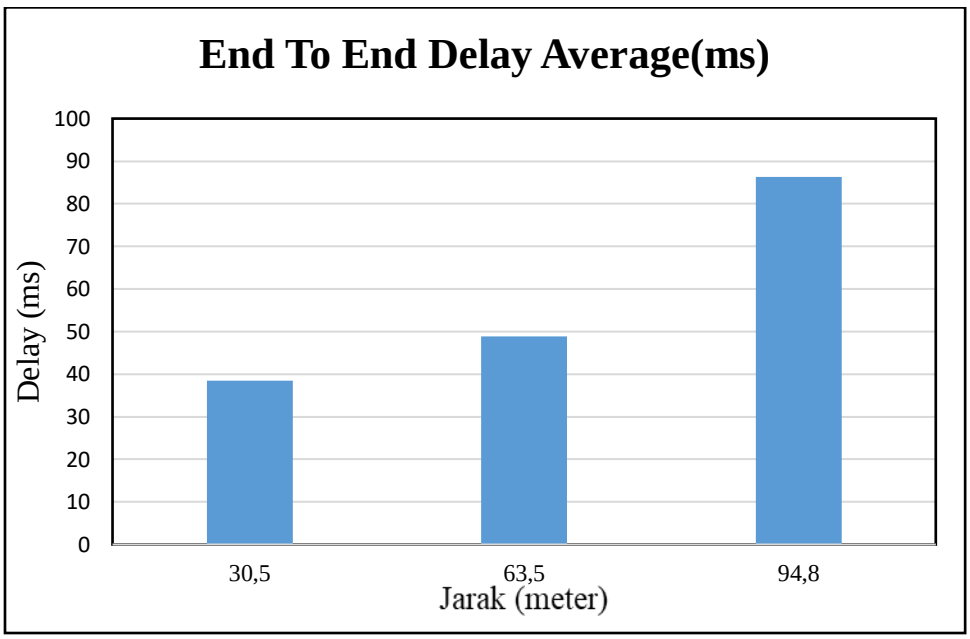
Pada gambar 4.37 pengukuran QOS pada node 2 dengan jarak 64. meter diperoleh data dari perangkat *gateway* yang dikirimkan ke perangkat pc melalui komunikasi serial pada table 4.6 menunjukan hasil delay 2.534 ms, *Throughput* 12.63 kbps dan *packet loss* sebesar 0.

Berdasarkan pengujian QOS (*Quality Of Service*) jaringan dengan 3 variasi jarak transmisi dapat diperoleh rata-rata *delay* pada masing-masing jarak yang ditunjukkan pada Tabel 4.8.

Tabel 4.8 Rata-rata delay

No	Distance(m)	*delay*(ms)
1	30.5	38.439
2	63.5	48.82
3	94.8	86.3

Hasil rata-rata *delay* dari tiap pengujian yang telah dilakukan, dapat menjelaskan bahwa dengan bertambahnya jarak mengakibatkan waktu yang yang dibutuhkan untuk mengirimkan data dari *gateway* ke masing-masing *node* akan bertambah lama. Tabel 4.8 menjelaskan pada jarak 94.8 meter rata-rata *delay* adalah 86.3 ms sementara pada jarak transmisi 30.5 m rata-rata *delay* adalah 38.43 ms. Gambar 4.38 menjelaskan perbandingan *delay* masing-masing pengujian.



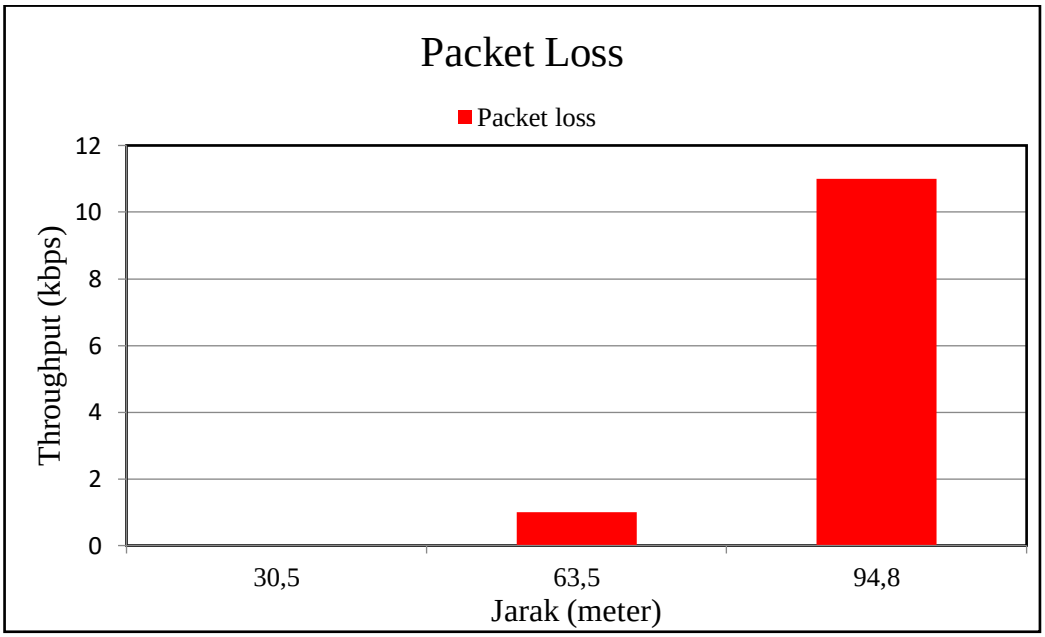
Gambar 4.38 Grafik perbandingan *end to end delay*

Berdasarkan pengujian *Quality Of Service* (QOS) jaringan dengan 3 variasi jarak transmisi dapat diperoleh nilai *packet loss* pada masing-masing jarak yang ditunjukkan pada Tabel 4.9.

Tabel 4.9 Tabel Packet Loss

No	Distance(m)	Packet loss
1	30.5	0
2	63.5	1
3	94.8	11

Tabel 4.9 menjelaskan pada jarak 94.8 meter, hasil pengukuran *packet loss* dari tiap pengujian yang telah dilakukan, dapat menjelaskan bahwa dengan bertambahnya jarak mengakibatkan *packet loss* akan semakin besar saat mengirimkan data dari *gateway* ke masing-masing node. Hasil dari tabel 4.9 dapat dilihat pada gambar 4.39.



Gambar 4.39 Perbandingan *packet loss*

Gambar 4.39 menjelaskan perbandingan *packet loss* masing-masing pengujian *packet loss* sebesar 11 paket, sementara pada jarak transmisi 60.5 meter terjadi *packet loss* sebesar 1 paket dan pada jarak 30.5 meter tidak terjadi *packet loss*. Proses pengukuran *packet loss* dilakukan pada sisi gateway, selama proses pengiriman data ke masing-masing *node gateway* akan menunggu hingga *node*

membalas dengan ack yang menunjukkan bahwa data telah diterima oleh masing-masing *node*. Selama kurun waktu tertentu *node* tidak membalas dengan ack, maka *gateway* akan melakukan *re-transmit* data ke *node*. Pengaturan *retries* diatur pada 15 kali, hingga 15 kali *re-transmit* data ke *node gateway* tidak mendapat balasan ack maka pengiriman data dianggap gagal dan disimpulkan bahwa telah terjadi *packet loss* pada proses pengiriman data.

Tabel 4.10 Tabel perbandingan antara protokol Ais dan Mavlink.

No	**Deskripsi**	**Ais messages**	**Mavlink**
1	Penggunaan	Vessel	UAV
2	Frekuensi *carrier*	Channel A 161.975Mhz (87B) Channel B 162.025Mhz (88B)	433Mhz, 915Mhz
3	Modulasi	FM	FSK
4	Payload	168 bits	255 bits
5	CRC	2 bytes	2 bytes
6	Header	6 bytes	5 bytes

Dari perbandingan *protocol* yang digunakan pada pengujian jaringan, dapat disimpulan bahwa penggunaan *protocol* mavlink dapat digunakan secara *custome* dibanding protocol AIS. Protokol mavlink pada pembentukan payload nya, dapat disusun sesuai dengan kebutuhan. Sementara protocol AIS harus mengikuti standar yang yang telah ditentukan. Pada pembentukan payload protocol mavlink data-data dapat disusun pada payload pada lokasi bytes ke 1 hingga ke 32, ukuran byte data sangat menentukan penentuan lokasi data pada payload.

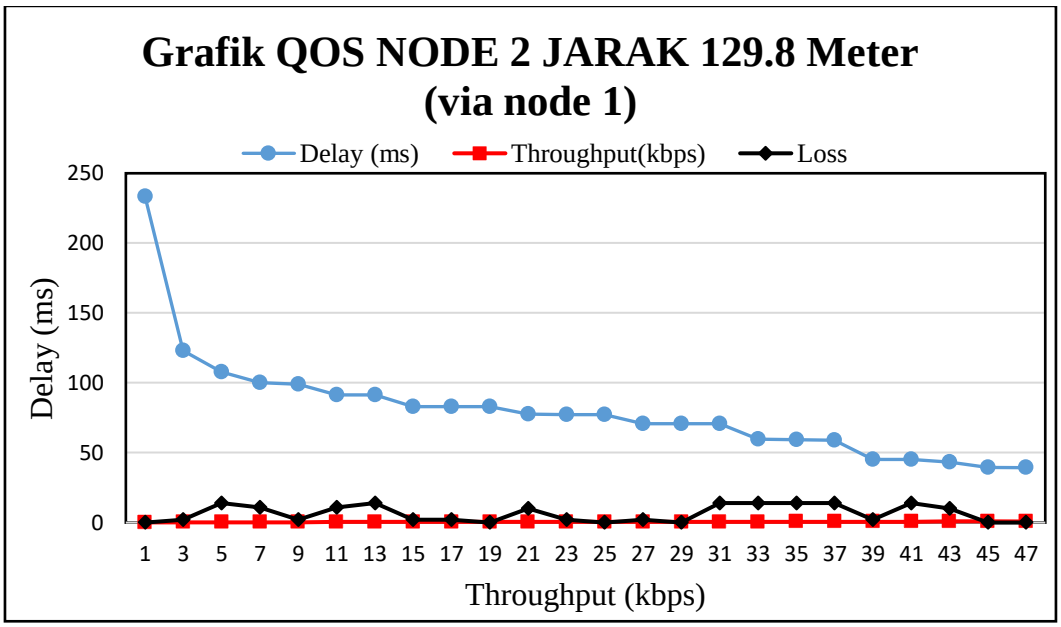
Penentuan lokasi harus disesuaikan antara pengirim dan penerima, jika sebuah data menempati 3 bytes lokasi maka pada penerima proses pembentukan ke data asli harus diambil dari 3 bytes data sesuai dengan lokasi *bytes* pada pengirim. Terdapat dua perbedaan mendasar antara dua protocol AIS dan Mavlink adalah dalam proses pembentukan payload, protocol AIS lebih hemat pada ukuran data. Protokol Ais melakukan konversi raw data ke bit-bit data dengan ukuran 6 bit, sementara pada protocol Mavlink ukuran data tidak mengalami perubahan ukuran, yaitu 8 bit.

Pada tabel 4.11 pengujian berikut merupakan hasil pengukuran yang dilakukan penulis melalui node 2 dalam jarak transmisi 129.8 meter dengan perantara *node* 1.

Tabel 4.11 Pengukuran Performa Jaringan *node* 2 Pada Jarak 129.8 meter (via *node* 1)

Node 2 Jarak 129.8 meter (via node 1)					
Packet size	**Delay (ms)**	**Throughput(kbps)**	**Loss**	**Packet send**	**Packet success**
32	238,812	0,14	0	0	0
32	233,434	0,14	0	0	0
32	122,896	0,27	2	25	23
32	103,008	0,32	2	24	22
32	96,982	0,33	2	31	29
32	86,782	0,37	2	26	24
32	83,156	0,39	2	27	25
32	77,794	0,42	10	41	31
32	73,188	0,44	0	1	1
32	72,608	0,45	2	28	26
32	69,85	0,46	2	23	21
32	67,254	0,48	2	30	28
32	43,238	0,75	10	42	32
32	20,844	1,54	2	17	15
32	18,904	1,71	0	4	4
32	18,15	1,77	1	6	5
32	14,58	2,21	1	13	12
32	13,546	2,37	2	21	19
32	11,03	2,91	2	22	20
32	10,186	3,15	1	7	6
32	10,15	3,17	1	15	14
32	8,632	3,73	1	16	15
32	5,64	5,68	1	8	7
32	4,228	7,57	2	20	18
32	4,098	7,81	1	12	11
32	3,936	8,14	0	3	3
32	3,886	8,24	1	9	8
32	2,904	11,02	1	14	13
32	2,666	12,01	2	19	17
32	2,612	12,26	2	18	16
32	2,534	12,63	0	1	1
32	2,482	12,81	0	2	2

Pengukuran QOS (*Quality Of Service*) dilakukan dengan mengukur sejumlah paket data sebanyak 47 paket dengan *payload 32 byte* yang dikirim oleh *gateway* ke *node*. Interval pengiriman data dari *gateway* ke *node* uji diatur sebesar 1000 ms. Pengukuran QOS dilakukan pada jarak antara *gateway* dengan *node* 2 uji sebesar 129.8 meter (via node 1). Proses pengukuran dilakukan oleh aplikasi yang dibuat penulis menggunakan Visual Studio 2012. Informasi data pengukuran *throughput*, *delay* dan *packet loss* diperoleh dari perangkat *gateway* yang dikirimkan ke perangkat pc atau laptop melalui komunikasi serial. Berdasarkan hasil pengukuran pada Tabel 4.9 dapat dibuat grafik untuk mem-*plot* data pengukuran QOS dalam bentuk grafik, yang ditunjukkan pada Gambar 4.38 berikut.



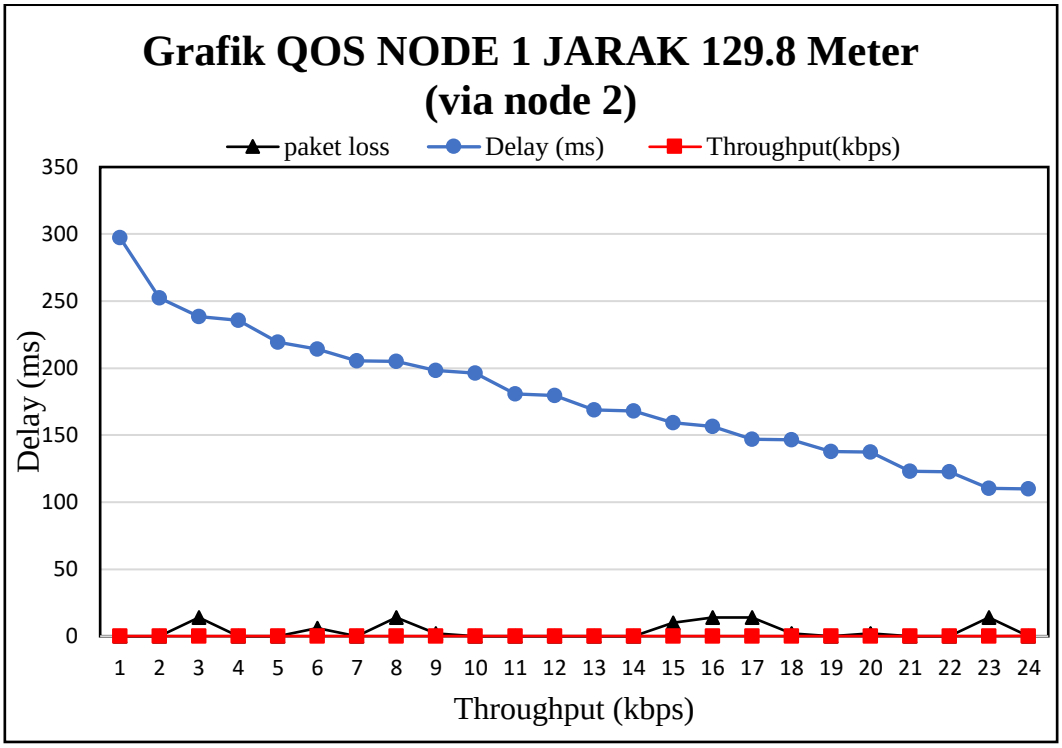
Gambar 4.40 grafik pengujian node 2 dengan jarak 129.8 meter dengan perantara node 1

Pada tabel 4.11 pengujian berikut merupakan hasil pengukuran yang dilakukan penulis melalui node 1 dalam jarak transmisi 129.8 meter dengan perantara node 2.

Tabel 4.12 Pengukuran Performa Jaringan node 1 Pada Jarak 129.8 meter (via node 2)

Node 1 Jarak 129.8 meter (via node 2)				
Delay (ms)	Throughput (kbps)	paket loss	Packet send	Packet success
297,442	0,107584	0	0	0
252,274	0,126846207	0	0	0
242,65	0,131877189	0	0	0
239,38	0,13367867	2	25	23
221,184	0,144675926	14	102	88
219,338	0,145893552	0	57	57
214,064	0,149488004	13	94	81
211,932	0,150991828	0	58	58
211,37	0,151393291	13	95	82
210,246	0,152202658	1	5	4
207,998	0,153847633	14	129	115
205,46	0,155748077	2	24	22
205,368	0,155817849	1	6	5
195,914	0,163336974	0	63	63
193,268	0,165573194	6	41	35
192,46	0,166268315	11	51	40
190,856	0,167665675	2	16	14
190,734	0,167772919	2	31	29
189,402	0,168952809	14	123	109
188,86	0,169437679	0	72	72
172,198	0,185832588	0	67	67
171,832	0,186228409	14	99	85

Berdasarkan hasil pengukuran pada Tabel 4.12 dapat dibuat grafik untuk mem-*plot* data pengukuran QOS dalam bentuk grafik, yang ditunjukkan pada Gambar 4.41 berikut.



Gambar 4.41 grafik pengujian node 1 dengan jarak 129.8 meter dengan perantara node 2

Bertambahnya besar *payload* yang dikirimkan akan mempengaruhi waktu pengiriman data, nilai waktu pengiriman sangat dipengaruhi oleh proses ack pada penerima untuk konfirmasi data yang diterima. Selama proses ack belum diterima oleh pengirim maka pengirim akan melakukan *retries* pengiriman hingga ack diterima. Konfirgurasi *retries* diatur pada pengirim dengan maksimal *retries* sebanyak 15 kali.

Pengujian variasi jarak transmisi membuktikan bahwa semakin jauh jarak transmisi mengakibatkan potensi kehilangan paket semakin besar, pada pengujian pada jarak 129.8 meter diperoleh *packet loss* sebesar 10 paket. Pada jarak transmisi 30.5 meter diperoleh *packet loss* sebesar 1 paket. *Packet loss* tidak hanya dipengaruhi oleh faktor jarak transmisi, faktor *obstacle* atau penghalang dapat mempengaruhi kehilangan paket.

BAB 5
KESIMPULAN

5.1. Kesimpulan

Kesimpulan-kesimpulan yang diperoleh dapat disebutkan, yaitu :

1. Pada konsep perancangan jaringan nirkabel yang dibuat diperoleh sebuah rancangan jaringan nirkabel yang dapat diintergrasikan dengan perangkat mikrokontroler dengan biaya rendah dan konsumsi daya yang rendah. Dari pengujian yang dilakukan penggunaan baterai 1s 3.7V 2200mAh dapat digunakan dalam kurun waktu 7 hari *non-stop*.
2. Pengujian variasi jarak transmisi membuktikan bahwa semakin jauh jarak jangkauan transmisi akan mengakibatkan waktu pengiriman data akan semakin lama. Hal ini dibuktikan pada jarak 90.8 meter waktu transmisi adalah 86.3 ms, sementara pada jarak 30.5 meter membutuhkan waktu 38.5 ms.
3. Pengujian variasi jarak transmisi membuktikan bahwa semakin lama *delay* transmisi akan mengakibatkan *throughput* semakin kecil.
4. Berdasarkan hasil pengujian yang telah dilakukan membuktikan jika posisi antena jauh dari tanah akan menghasilkan jarak jangkauan lebih jauh.
5. Pengujian variasi jarak transmisi membuktikan bahwa semakin jauh jarak transmisi mengakibatkan potensi kehilangan paket semakin besar, pada pengujian pada jarak 90.8 meter diperoleh *packet loss* sebesar 11 paket. Pada jarak transmisi 30.5 meter diperoleh *packet loss* sebesar 1 paket. *Packet loss* tidak hanya dipengaruhi oleh faktor jarak transmisi, faktor *obstacle* atau penghalang dapat mempengaruhi kehilangan paket.

5.2.Saran

Berdasarkan hasil penelitian komunikasi data VMeS (*Vessel Messaging System*) pada kapal nelayan yang telah dilakukan penulis dapat diberikan saran untuk pengembangan sistem lebih lanjut. Saran-saran tersebut adalah :

1. Mengembangkan sistem transmisi dengan daya pancar lebih besar, hal ini berkaitan dengan jarak transmisi yang lebih jauh.

2. Menggunakan antenna sebagai media transmisi dengan *gain* lebih tinggi, hal ini berkaitan dengan jangkauan transmisi dan penerimaan yang lebih baik.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

[1] IMO, 1998. Recommendation on performance standard for a universal shipborne Automatic Identification System (AIS), London: IMO Resolution MSC 74 (69) [2] NMES. "Automatic Identification Systems (AIS)". 2011.

[2] K.Hata, J.Fukuto, K.Hasegawa, K.Niwa : Evaluation of AIS communication using AIS Simulator, The journal of Japan Institute of Navigation. No.117, Japan Institute of Navigation, Japan, pp.27-33, 2007.

[3] Eric E. Johnson, Zibin Tang, Manikanden Balakrishnan, Huiyan Zhang, and Srugun Sreepuram. "*Routing* in HF Ad-Hoc WANs". MILCOM 2004. 2004.

[4] Charles E. Perkins, Elizabeth M. Royer. "Ad-hoc On-Demand Distance Vector *Routing*". In Proc WMCSA'99. Second IEEE Workshop on *Mobile* Computing Systems and Applications. 1999.

[5] IEC: Maritime navigation and radiocommunication equipment and systems - Class B shipborne equipment of the automatic identification system (AIS) - Part 1: Carrier-sense time division multiple access (CSTDMA) techniques, IEC 62287-1 Ed.1, IEC, 2006

[6] Purnama, B. E.," Pemanfaatan *Global Positioning System* Untuk Pelacakan Objek Bergerak", Journal *Speed* – Volume 2 No 2 – 2009.

[7] Technical Characteristic for a Universal Shipborne Automatic Identification System Using Time division Multiple Access in the Maritime Band, ITU-R Recommendation M.1371-4 (2002)

[8] Proakis John G. Salehi M., "Digital Communications 5th Edition", McGraw Hill 2008.

[9] *Pengembangan Sistem Pengoperasian AIS pada satelit LAPAN Ekuatorial untuk Pemantauan Kapal di Wilayah Indonesia*, Lembaga Penerbangan dan Antariksa Nasional, Bogor (2010).

[10] Afif, M. Ardita, M. Affandi A., "Implementasi *Protocol Routing* Jaringan Ad Hoc Multi User pada *Gateway* untuk Sistem Komunikasi Kapal Laut," Jurnal Teknik ITS Vol. 1, September 2012.

[11] Mahesh K. M., Samir R D., "*On-demand Multipath Distance Vector Routing in Ad hoc Network*", 2001.

[12] Johnson, D., "*Routing in Ad hoc Networks of Mobile Host*", *Proceeding IEEE Workshop on Mobile Computer System and Applications*.1994

[13] Imantaka A., "*Rancang Bangun Layanan SMS Pada Teknologi VmeS Untuk Sistem Komunikasi Kapal Laut*", Tugas Akhir JTE FTI-ITS, Juni 2010

[14] Hyo S. K., Sugoog Shon, Sung K. C., Sung, K. C.," *A Simulation Study of P-Persistent CSMA-CA for IEEE 802.15.4 LR-WPAN*". 2007

[15] Lestari, D. S. Setijadi, E. Suwadi, "Perancangan dan Implementasi Modulator FSK untuk Perangkat Transmitter Satelit ITS-SAT pada Frekuensi 436,915 MHz", Jurnal Teknik POMITS Vol. 2, No. 2, 2013.

[16] Betke, Klaus. (2001). The NMEA 0183 *Protocol*. USA: National Marine Electronics Association. NMEA. (2015). The NMEA 0183 Information Sheet. USA: National Marine Electronics Association.

[17] NMEA. (2002). NMEA 0183 Standard for Interfacing Marine Electronic Device. USA: National Marine Electronics Association.

[18] Andhika, F. Pitana, T. Affandi, A. "*Protocol* Interchangeable Data pada VMeS (Vessel Messaging System) dan AIS (Automatic Identification System)," Jurnal Teknik ITS Vol. 1, September 2012.

[19] Lestari, D. S. Setijadi, E. Suwadi, "Perancangan dan Implementasi Modulator FSK untuk Perangkat Transmitter Satelit ITS-SAT pada Frekuensi 436,915 MHz", Jurnal Teknik POMITS Vol. 2, No. 2, 2013.

[20] Komatsu, Hiroaki. *Furuno FA-30 AIS receiver Operator's Manual*, Hyogo, 2007.

[21] H.-S. W. So, K. Fall, and J. WalrandIn-Stat, "*Packet loss* behavior in a *wireless* broadcast sensor network," tech. rep., University of California, Berkeley, 2003

[22] Peraturan Menteri Komunikasi Dan Informatika Republik Indonesia, Perencanaan Penggunaan Spektrum Frekuensi Radio Pada Pita Frekuensi Radio 350 – 438 MHz, Bogor (2013).

Halaman ini sengaja dikosongkan

LAMPIRAN

1. Pengujian pada *node* 1 (*Protocol* AIS) pada jarak 64.5 meter

***NODE* 1 JARAK 64.5 Meter**					
***Packet* size**	***Delay* (ms)**	***Throughput*(kbps)**	**Loss**	***Packet* send**	***Packet* success**
32	2,43	13,16872428	0	1	1
32	31,96	1,001251564	0	3	3
32	20,256	1,579778831	1	4	3
32	12,238	2,614806341	1	5	4
32	10,052	3,18344608	1	6	5
32	12,958	2,469516901	1	8	7
32	7,24	4,419889503	1	11	10
32	2,43	13,16872428	1	12	11
32	10,358	3,089399498	1	13	12
32	6,402	4,998437988	1	14	13
32	11,292	2,833864683	1	15	14
32	16,272	1,966568338	1	17	16
32	4,412	7,25294651	1	18	17
32	5,414	5,910602143	1	19	18
32	2,43	13,16872428	1	20	19
32	10,054	3,182812811	1	21	20
32	17,754	1,802410724	1	24	23
32	17,386	1,840561371	1	25	24
32	19,504	1,640689089	1	26	25
32	17,438	1,835072829	1	27	26
32	17,598	1,818388453	1	28	27
32	17,678	1,81015952	1	29	28
32	17,226	1,85765703	1	30	29
32	17,172	1,863498719	1	31	30
32	21,856	1,464128843	1	32	31
32	19,73	1,62189559	1	33	32
32	27,074	1,181945778	1	34	33
32	19,48	1,642710472	1	35	34
32	17,676	1,810364336	1	36	35
32	21,678	1,476150936	1	38	37
32	21,358	1,498267628	1	39	38
32	19,162	1,669971819	1	40	39
32	19,37	1,652039236	1	41	40
32	18,23	1,755348327	1	42	41
32	31,324	1,021580896	1	43	42

32	21,434	1,492955118	1	44	43
32	17,578	1,82045739	1	45	44
32	21,474	1,490174164	1	46	45
32	17,222	1,858088491	1	47	46
32	18,022	1,775607591	1	48	47
32	17,384	1,840773125	1	49	48
32	17,454	1,833390627	1	50	49
32	17,508	1,827735892	1	51	50
32	17,306	1,849069687	1	52	51
32	18,288	1,749781277	1	53	52
32	29,71	1,077078425	1	54	53
32	29,534	1,083496987	1	55	54
32	19,266	1,660957127	1	56	55
32	29,106	1,099429671	1	57	56
32	18,152	1,762891141	1	58	57

2. Pengujian pada *node* 1 (*Protocol* AIS) pada jarak 98.2 meter

***NODE* 1 JARAK 98.2 Meter**					
***Packet* size**	***Delay* (ms)**	***Throughput*(kbps)**	**Loss**	***Packet* send**	***Packet* success**
32	77,044	0,415347074	0	1	1
32	107,446	0,297824023	0	2	2
32	150,256	0,212969865	0	3	3
32	155,18	0,206212141	0	4	4
32	150,026	0,213296362	0	5	5
32	102,548	0,312048992	1	7	6
32	180,778	0,17701269	1	8	7
32	161,682	0,197919373	1	9	8
32	124,5	0,257028112	1	10	9
32	139,496	0,229397259	1	11	10
32	100,402	0,318718751	1	12	11
32	152,498	0,209838818	1	13	12
32	113,576	0,281749665	1	14	13
32	59,104	0,541418517	1	15	14
32	80,32	0,398406375	1	16	15
32	116,152	0,275501068	2	18	16
32	117,344	0,272702482	2	19	17
32	69,288	0,461840434	3	21	18
32	114,68	0,279037321	3	22	19
32	159,698	0,200378214	3	23	20
32	85,878	0,372621626	3	24	21

32	144,092	0,222080338	3	25	22
32	124,696	0,25662411	4	27	23
32	118,792	0,269378409	4	28	24
32	93,2	0,343347639	4	29	25
32	104,158	0,307225561	4	30	26
32	76,188	0,42001365	4	31	27
32	105,912	0,302137624	4	32	28
32	70,52	0,45377198	4	33	29
32	64,756	0,494162703	4	34	30
32	84,664	0,37796466	4	35	31
32	82,354	0,388566433	4	36	32
32	138,8	0,23054755	4	37	33
32	68,81	0,465048685	4	38	34
32	61,408	0,521104742	4	39	35
32	63,044	0,507582006	4	40	36
32	45,734	0,699698255	4	41	37
32	28,038	1,141308225	4	42	38
32	36,636	0,873457801	4	43	39
32	24,152	1,324942034	4	44	40
32	28,242	1,133064231	4	45	41
32	8,912	3,590664273	4	46	42
32	16,242	1,970200714	4	47	43
32	8,43	3,795966785	4	48	44
32	8,866	3,609293932	4	49	45
32	8,398	3,810431055	4	50	46
32	8,512	3,759398496	4	51	47
32	8,376	3,820439351	4	52	48
32	25,364	1,261630658	4	53	49
32	38,53	0,830521671	4	54	50
32	84,85	0,377136123	4	55	51
32	8,294	3,858210755	4	56	52
32	172,198	0,185832588	4	58	54
32	133,426	0,239833316	4	59	55
32	156,13	0,204957407	4	60	56
32	219,338	0,145893552	4	61	57
32	146,636	0,218227448	4	62	58
32	88,48	0,361663653	5	64	59
32	114,184	0,280249422	5	65	60
32	80,432	0,397851601	5	66	61
32	122,228	0,261805806	5	67	62
32	111,954	0,285831681	5	69	64
32	112,89	0,283461777	5	70	65
32	161,586	0,198036959	5	71	66

32	9,15	3,49726776	0	15	15
32	26,602	1,202917074	0	16	16
32	22,132	1,445870233	0	17	17
32	10,03	3,190428714	0	18	18
32	27,35	1,170018282	0	19	19
32	8,42	3,800475059	0	20	20
32	11,904	2,688172043	0	21	21
32	83,118	0,384994827	0	22	22
32	23,428	1,365886973	0	23	23
32	145,218	0,220358358	0	24	24
32	214,064	0,149488004	0	25	25
32	163,684	0,195498644	0	26	26
32	211,37	0,151393291	0	27	27
32	115,644	0,276711286	0	28	28
32	196,23	0,163073944	0	29	29
32	192,46	0,166268315	0	30	30
32	177,174	0,180613408	0	31	31
32	141,536	0,226090889	0	32	32
32	221,184	0,144675926	0	33	33
32	150,218	0,213023739	1	35	34
32	207,998	0,153847633	1	36	35
32	186,968	0,171152283	1	37	36
32	239,38	0,13367867	1	38	37
32	205,052	0,156057976	1	39	38
32	177,044	0,180746029	2	41	39
32	117,826	0,271586916	2	42	40
32	134,73	0,237512061	3	44	41
32	200,594	0,159526207	4	47	43
32	122,86	0,260459059	5	49	44
32	160,368	0,199541056	5	50	45
32	171,832	0,186228409	5	51	46
32	195,914	0,163336974	6	54	48
32	189,402	0,168952809	6	55	49
32	105,738	0,302634814	6	56	50
32	146,798	0,217986621	6	58	52
32	146,824	0,217948019	6	59	53
32	126,388	0,253188594	6	60	54
32	148,7	0,215198386	6	61	55
32	110,242	0,290270496	6	62	56
32	84,922	0,376816373	6	63	57
32	105,464	0,303421073	6	64	58
32	126,452	0,25306045	6	65	59
32	113,872	0,281017283	6	66	60

3. Pengujian pada *node* 2 (*Protocol* MAVLink) pada jarak 63.5 meter

***NODE* 2 JARAK 63.5 Meter**					
Packet size	*Delay* (ms)	*Throughput*(kbps)	Loss	*Packet* send	*Packet* success
32	74,796	0,427830365	0	1	1
32	70,914	0,451250811	0	2	2
32	65,668	0,48729975	0	3	3
32	58,202	0,549809285	0	4	4
32	56,588	0,565490917	0	5	5
32	47,192	0,678081031	0	6	6
32	43,504	0,735564546	0	7	7
32	46,106	0,694052835	0	8	8
32	38,854	0,823596026	0	9	9
32	35,112	0,911369332	0	10	10
32	29,854	1,071883165	0	11	11
32	25,088	1,275510204	0	12	12
32	15,822	2,022500316	0	13	13
32	10,134	3,157686994	0	14	14
32	7,884	4,058853374	0	15	15
32	4,152	7,707129094	0	16	16
32	2,616	12,2324159	0	17	17
32	2,452	13,05057096	0	18	18
32	2,482	12,89282836	0	19	19
32	3,274	9,773976787	0	20	20
32	5,184	6,172839506	0	21	21
32	2,586	12,37432328	0	22	22
32	2,456	13,02931596	0	23	23
32	2,482	12,89282836	0	24	24
32	65,96	0,485142511	0	25	25
32	99,498	0,321614505	0	26	26
32	68,342	0,468233297	0	27	27
32	98,84	0,323755565	0	28	28
32	66,146	0,483778309	0	29	29
32	96,27	0,332398463	0	30	30
32	92,116	0,347388076	0	31	31
32	93,492	0,342275275	0	32	32
32	84,278	0,379695769	0	33	33
32	28,178	1,135637732	0	34	34
32	59,926	0,533991923	0	35	35
32	61,46	0,520663846	0	36	36
32	48,808	0,655630225	0	37	37
32	58,364	0,548283188	0	38	38

32	73,504	0,435350457	0	39	39
32	80,092	0,399540528	0	40	40
32	73,502	0,435362303	0	41	41
32	70,724	0,452463096	0	42	42
32	65,778	0,486484843	0	43	43
32	68,12	0,469759248	0	44	44
32	52,378	0,610943526	0	45	45
32	71,73	0,446117385	0	46	46

4. Pengujian pada *node* 2 (*Protocol* MAVLink) pada jarak 94.8 meter

***NODE* 2 JARAK 94.8 Meter**					
***Packet* size**	***Delay* (ms)**	***Throughput*(kbps)**	**Loss**	***Packet* send**	***Packet* success**
32	238,812	0,133996617	0	0	0
32	73,188	0,437230147	0	1	1
32	233,434	0,137083715	0	0	0
32	2,534	12,62825572	0	1	1
32	2,482	12,89282836	0	2	2
32	3,936	8,130081301	0	3	3
32	18,904	1,692763436	0	4	4
32	18,15	1,763085399	1	6	5
32	10,186	3,141566856	1	7	6
32	5,64	5,673758865	1	8	7
32	3,886	8,234688626	1	9	8
32	2,482	12,89282836	1	10	9
32	2,456	13,02931596	1	11	10
32	4,098	7,808687164	1	12	11
32	14,58	2,19478738	1	13	12
32	2,904	11,01928375	1	14	13
32	10,15	3,15270936	1	15	14
32	8,632	3,707136237	1	16	15
32	20,844	1,53521397	2	17	15
32	2,612	12,25114855	2	18	16
32	2,666	12,00300075	2	19	17
32	4,228	7,56859035	2	20	18
32	13,546	2,36232098	2	21	19
32	11,03	2,901178604	2	22	20
32	69,85	0,458124553	2	23	21
32	103,008	0,310655483	2	24	22

32	122,896	0,260382763	2	25	23
32	86,782	0,368740061	2	26	24
32	83,156	0,384818895	2	27	25
32	72,608	0,440722785	2	28	26
32	67,254	0,47580813	2	30	28
32	96,982	0,329958137	2	31	29
32	77,794	0,411342777	10	41	31
32	43,238	0,740089736	10	42	32
32	68,318	0,468397787	10	43	33
32	61,75	0,518218623	10	44	34
32	74,112	0,431778929	10	45	35
32	82,918	0,385923442	10	46	36
32	69,436	0,46085604	11	48	37
32	76,218	0,41984833	11	49	38
32	91,28	0,350569676	11	50	39
32	100,164	0,319476059	11	51	40
32	83,818	0,38177957	11	53	42
32	65,95	0,485216073	11	54	43
32	54,69	0,585116109	11	56	45
32	91,162	0,351023453	12	58	46
32	38,386	0,833637264	12	59	47
32	70,484	0,454003746	12	60	48
32	54,684	0,585180309	12	61	49
32	61,41	0,521087771	12	62	50
32	90,948	0,351849408	12	63	51
32	72,606	0,440734925	12	64	52
32	66,412	0,481840631	12	65	53
32	57,646	0,555112237	12	66	54
32	36,518	0,876280191	12	67	55
32	65,248	0,490436488	12	68	56
32	83,326	0,384033795	12	69	57
32	40,07	0,798602446	12	70	58
32	61,99	0,516212292	12	71	59
32	58,338	0,548527546	12	72	60
32	50,98	0,627697136	12	73	61
32	47,752	0,670129	12	74	62
32	37,48	0,853788687	12	75	63
32	34,69	0,922456039	12	76	64
32	29,232	1,09469075	12	77	65
32	17,28	1,851851852	12	78	66
32	2,478	12,91364003	13	80	67
32	2,59	12,35521236	13	81	68
32	5,548	5,767844268	13	82	69

32	14,944	2,141327623	13	83	70
32	2,456	13,02931596	13	84	71
32	6,296	5,082592122	13	85	72
32	2,614	12,24177506	13	86	73
32	2,56	12,5	13	87	74
32	2,454	13,0399348	13	88	75
32	2,666	12,00300075	13	89	76
32	3,462	9,243212016	13	90	77
32	6,402	4,998437988	13	91	78
32	2,562	12,490242	13	92	79
32	10,372	3,085229464	13	93	80
32	107,574	0,297469649	13	94	81
32	104,822	0,305279426	13	95	82
32	71,174	0,449602383	13	96	83
32	85,26	0,375322543	13	97	84
32	81,242	0,393884936	13	98	85
32	95,754	0,334189694	14	99	85
32	84,62	0,378161191	14	100	86
32	37,32	0,857449089	14	101	87
32	107,84	0,296735905	14	102	88
32	91,454	0,349902683	14	103	89
32	48,888	0,654557356	14	104	90
32	76,998	0,41559521	14	105	91
32	74,608	0,428908428	14	106	92
32	37,126	0,861929645	14	107	93
32	91,876	0,348295529	14	108	94
32	24,852	1,287622727	14	109	95
32	82,56	0,387596899	14	110	96
32	91,24	0,350723367	14	111	97
32	39,038	0,819714125	14	112	98
32	62,336	0,513347023	14	113	99
32	72,076	0,443975803	14	114	100
32	57,622	0,555343445	14	115	101
32	89,122	0,359058369	14	116	102
32	84,832	0,377216145	14	117	103
32	59,606	0,536858705	14	118	104
32	79,726	0,401374708	14	119	105
32	45,074	0,709943648	14	120	106
32	63,314	0,505417443	14	121	107
32	54,712	0,584880831	14	122	108
32	96,918	0,330176025	14	123	109
32	79,566	0,402181836	14	124	110
32	27,048	1,183081928	14	125	111

32	29,694	1,077658786	14	127	113
32	54,444	0,5877599	14	128	114
32	103,642	0,308755138	14	129	115
32	30,49	1,049524434	14	130	116
32	58,918	0,543127737	14	131	117
32	64,028	0,499781346	14	132	118
32	70,83	0,451785966	14	133	119
32	80,278	0,398614814	14	134	120
32	57,196	0,559479684	14	135	121
32	76,178	0,420068786	14	136	122
32	91,532	0,34960451	14	137	123
32	69,508	0,460378661	14	138	124
32	86,422	0,370276087	14	139	125
32	53,438	0,598824806	14	140	126
32	74,556	0,429207576	14	141	127
32	59,238	0,540193795	14	142	128
32	55,906	0,572389368	14	143	129
32	50,266	0,636613218	14	144	130
32	29,454	1,086439872	14	145	131
32	36,764	0,870416712	14	146	132
32	29,958	1,068162094	14	147	133
32	23,976	1,334668001	14	148	134
32	18,948	1,688832594	14	149	135
32	11,188	2,860207365	14	150	136
32	4,334	7,383479465	14	151	137
32	2,85	11,22807018	14	152	138

5. Pengujian Ad-Hoc pada jarak 129.8 meter

***NODE* 2 JARAK 129.8 Meter (via *node* 1)**					
***Packet* size**	***Delay* (ms)**	***Throughput*(kbps)**	**Loss**	***Packet* send**	***Packet* success**
32	238,812	0,133996617	0	0	0
32	73,188	0,437230147	0	1	1
32	233,434	0,137083715	0	0	0
32	2,534	12,62825572	0	1	1
32	2,482	12,89282836	0	2	2
32	3,936	8,130081301	0	3	3
32	18,904	1,692763436	0	4	4
32	18,15	1,763085399	1	6	5
32	10,186	3,141566856	1	7	6
32	5,64	5,673758865	1	8	7
32	3,886	8,234688626	1	9	8

32	2,482	12,89282836	1	10	9
32	2,456	13,02931596	1	11	10
32	4,098	7,808687164	1	12	11
32	14,58	2,19478738	1	13	12
32	2,904	11,01928375	1	14	13
32	10,15	3,15270936	1	15	14
32	8,632	3,707136237	1	16	15
32	20,844	1,53521397	2	17	15
32	2,612	12,25114855	2	18	16
32	2,666	12,00300075	2	19	17
32	4,228	7,56859035	2	20	18
32	13,546	2,36232098	2	21	19
32	11,03	2,901178604	2	22	20
32	69,85	0,458124553	2	23	21
32	103,008	0,310655483	2	24	22
32	122,896	0,260382763	2	25	23
32	86,782	0,368740061	2	26	24
32	83,156	0,384818895	2	27	25
32	72,608	0,440722785	2	28	26
32	67,254	0,47580813	2	30	28
32	96,982	0,329958137	2	31	29
32	77,794	0,411342777	10	41	31
32	43,238	0,740089736	10	42	32
32	68,318	0,468397787	10	43	33
32	61,75	0,518218623	10	44	34
32	74,112	0,431778929	10	45	35
32	82,918	0,385923442	10	46	36
32	69,436	0,46085604	11	48	37
32	76,218	0,41984833	11	49	38
32	91,28	0,350569676	11	50	39
32	100,164	0,319476059	11	51	40
32	83,818	0,38177957	11	53	42
32	65,95	0,485216073	11	54	43
32	54,69	0,585116109	11	56	45
32	91,162	0,351023453	12	58	46
32	38,386	0,833637264	12	59	47
32	70,484	0,454003746	12	60	48
32	54,684	0,585180309	12	61	49
32	61,41	0,521087771	12	62	50
32	90,948	0,351849408	12	63	51
32	72,606	0,440734925	12	64	52
32	66,412	0,481840631	12	65	53
32	57,646	0,555112237	12	66	54

6. Proses *Payload Protocol* AIS

a) Melakukan konversi data ke 6 bit.

Proses konversi data ke 6 bit dapat dijelaskan pada kode program berikut

Inisialisasi nilai dan menyimpannya pada LSB

sixbit_*payload*[0]=type & B00111111;

Mengambil 2 bit pengulangan yang terendah dan masukkan ke bit 5,4 kemudian ambil bit 29,28,27,26 dalam mmsi dan masukkan ke dalam 4 bit terendah

sixbit_*payload*[1]=((repeat & B00000011)<<4)+((*NODE* & 0x3c000000)>>26);

Mengambil bit 25 ~ 20 dari mmsi

sixbit_*payload*[2]= ((*NODE* & 0x03f00000)>>20);

Mengambil bit 19 ~ 14 dari mmsi

sixbit_*payload*[3]= ((*NODE* & 0x000fc000)>>14);

Mengambil bit 13 ~ 8 dari mmsi

sixbit_*payload*[4]= ((*NODE* & 0x3f00)>>8);

Mengambil bit 7 ~ 2 dari mmsi

sixbit_*payload*[5]= ((*NODE* & 0xfc)>>2);

Mengambil bit 1 ~ 0 dari mmsi ke bit 5 ~ 4, mendatkan bit ke 3 ~ 0 dari status sebagai bit 3 ~ 0

sixbit_*payload*[6]=((*NODE* & B00000011)<<4)+((status & B00001111));

Mengambil bit 7~2 dari data *turn*

sixbit_*payload*[7]= (turn & B11111100)>>2;

Mengambil bit ke 1 ~ 0 dari sekuen bit 5 ~ 4, bit ke 9 ~ 6 dari data SOG sebagai bit ke 3 ~ 0

sixbit_*payload*[8]= ((turn & B00000011)<<4)+((SOG & 0x03c0)>>6);

Mengambil bit ke 5~0 dari data SOG (*speed over ground*)

sixbit_*payload*[9]= (SOG & 0B00111111);

Mengambil bit 0 dari akurasi sebagai bit 5, dan bit 27-23 dari lon sebagai bit 4-0

sixbit_*payload*[10]= ((accuracy & B00000001)<<5)+((long10kmin & 0x0f800000)>>23);

Mengambil bit ke 22 ~ 17 dari data *longitude*

sixbit_*payload*[13]= ((long10kmin & 0x07E0)>>5);

Mengambil bit 4-0 dari data *longitude* sebagai bit ke 5-1 dan bit ke 26 dari *latitude* untuk bit 0

sixbit_*payload*[14]= ((long10kmin & B00011111)<<1)+((lat10kmin & 0x04000000)>>26);

bit ke 25-20 dari data *latitude*

sixbit_*payload*[15]= ((lat10kmin & 0x03F00000)>>20);

bit ke 19-14 dari data *latitude*

sixbit_*payload*[16]= ((lat10kmin & 0x0FC000)>>14);

bit ke 13-8 dari data *latitude*

sixbit_*payload*[17]= ((lat10kmin & 0x3F00)>>8);

bit ke 7-2 dari data *latitude*

sixbit_*payload*[18]= ((lat10kmin & 0xFC)>>2);

Mengambil bit ke 1-0 dari data *latitude* menjadi bit ke 5-4 dan COG bit ke 11-8 sebagai bit ke 3-0

sixbit_*payload*[19]= ((lat10kmin & B00000011)<<4)+((COG & 0x0F00)>>8);

bit ke 7-2 dari data COG

sixbit_*payload*[20]= ((COG & 0xFC)>>2);

Mengambil bit ke 1-0 dari COG sebagai bit ke 5-4 dan bit 8-5 sebagai bit 3-0

sixbit_*payload*[21]= ((COG & B00000011)<<4)+((compass & 0x01E0)>>5);

Mengambil bit ke 4-0 dari heading, bit ke 5-1 dan bit 5 waktu (detik) sebagai bit 0

sixbit_*payload*[22]= ((compass & B00011111)<<1)+((sec & B00100000)>>5);

Mengambil bit ke 4-0 dari waktu(detik) dan 1 dari data *manuver*

sixbit_*payload*[23]= ((sec & B00011111)<<1)+((manuver & B00000010)>>1);

Mengambil bit 0 dari data manuver sebagai bit ke 5, bit ke 2-0 sebagai bit ke 4-2, raim bit 0 sebagai bit ke 1 dan sync_state bit ke 1 sebagai bit ke 0

sixbit_*payload*[24]= (((maneuver & B00000001)<<5)+((spare & B00000111)<<2)+((raim &B00000001)<<1)+((sync_state & B00000010)>>1))
;

Mengambil data sync_state bit 0 sebagai bit 5, slot time_out bit 2 ~ 0 sebagai bit 4 ~ 2, dan bit 4 ~ 3 dari waktu(jam) sebagai bit ke 1 ~ 0

sixbit_*payload*[25]= (((sync_state & B00000001)<<5)+((slot_time_out & B00000111)<<2)+((hour & B00011000)>>3));

Mengambil bit 2 ~ 0 dari jam + 6-4 dari menit

sixbit_*payload*[26]= (((hour & B00000111)<<3)+((minute & B01110000)>>4));

bit 3 ~ 0 menit + 2 bit sisa

sixbit_*payload*[27]= (((minute & B00001111)<<2)+(second & B00000011));

b) Melakukan konversi data 6 bit ke 8 bit.

Proses konversi data 6 bit ke 8 bit dapat dijelaskan pada kode program berikut

Pada masing-masing *array* data *payload* dilakukan proses konversi

Data 6 bit dijumlahkan dengan nilai 48

Data_six_bit+= 48;

Jika nilai bit lebih besar dari 87, maka data six bit akan dijumlahkan dengan nilai 8

if (Data_six_bit > 87){ Data_six_bit += 8;}

c) Menyusun data yang terdiri dari

Header + Messages count + Messages number + Sequence id + Channel + *Payload*

Sehingga akan diperoleh format data seperti contoh data berikut

!AIVDM,1,1,,B,10003rPP02`408Asg;Qh06if05q@,

d) Menghitung nilai checksum, dengan proses XOR semua bit data pada proses nomer 3.

e) Menambahkan akhiran *0 sebelum menambahkan nilai checksum, sehingga format AIS secara lengkap sebagai berikut.

!AIVDM,1,1,,B,10003rPP02`408Asg;Qh06if05q@,0*51

7. program

a. Penyusunan mavlink payload

```
mav_put_4.val = latitude;
    buf[0] = mav_put_4.bytes[3];
    buf[1] = mav_put_4.bytes[2];
    buf[2] = mav_put_4.bytes[1];
    buf[3] = mav_put_4.bytes[0];
    mav_put_4.val = longitude;
    buf[4] = mav_put_4.bytes[3];
    buf[5] = mav_put_4.bytes[2];
    buf[6] = mav_put_4.bytes[1];
    buf[7] = mav_put_4.bytes[0];
    mav_put_4.val = SOG;
    buf[8] = mav_put_4.bytes[3];
    buf[9] = mav_put_4.bytes[2];
    buf[10] = mav_put_4.bytes[1];
    buf[11] = mav_put_4.bytes[0];
    mav_put_4.val = COG;
    buf[12] = mav_put_4.bytes[3];
    buf[13] = mav_put_4.bytes[2];
    buf[14] = mav_put_4.bytes[1];
    buf[15] = mav_put_4.bytes[0];
    mav_put_4.val = altitude;
    buf[16] = mav_put_4.bytes[3];
    buf[17] = mav_put_4.bytes[2];
    buf[18] = mav_put_4.bytes[1];
    buf[19] = mav_put_4.bytes[0];
    mav_put_4.val = hdop;
    buf[20] = mav_put_4.bytes[3];
    buf[21] = mav_put_4.bytes[2];
    buf[22] = mav_put_4.bytes[1];
    buf[23] = mav_put_4.bytes[0];
```

```
            buf[24] = numSatelite;
            mav_put_2.val = (int)temperature;
            buf[25] = mav_put_4.bytes[1];
            buf[26] = mav_put_4.bytes[0];
        mav_put_4.val = pressure;
        buf[27] = mav_put_4.bytes[3];
        buf[28] = mav_put_4.bytes[2];
        buf[29] = mav_put_2.bytes[1];
        buf[30] = mav_put_2.bytes[0];
```

b. Pengiriman mavlink payload

```
      if (!mesh.write(&buf, 'D' , 32)) {
        if ( !mesh.checkConnection() ) {
            if(!mesh.renewAddress()){
               mesh.begin();
            }
        }
      }
   }
 }
```

c. Proses penanganan request dari gateway pada node

```
if(network.available()){
  payload_ payload_t;
  RF24NetworkHeader header_;
  network.read(header_,&payload_t,sizeof(payload_t));
```

d. Request data

```
if(payload_t.f == true){
        IS_REQ_DATA = true;
}else{
```

e. Ping

```
        IS_REQ_DATA = false;
        if (!mesh.write(&payload_t, 'M' , sizeof(payload_t))) {
          Serial.println("#Sent back failed");
        }else{
          Serial.println("#Sent back success!...");
        }
    }
}
```

f. Kelas penanganan 32 bytes end to end delay

```
struct payload_
{
    unsigned long a1; //4 bytes
    unsigned long a2;
    unsigned long a3;
    unsigned long a4;
    unsigned long a5;
    unsigned long a6;
    unsigned long a7;
    boolean f;      //4bytes
};
```

g. Kode program gateway penanganan 32 bytes end to end delay

```
if((char)header.type=='M') {
        unsigned long respTime;
        Payload_t payload_t;
```

```
            if(network.read(header,&payload_t,sizeof(payload_t))){
          double enddelay = ((double)(micros()-payload_t.a1)/2);
          unsigned long has_resp_time = micros();
          uint32_t packet_delay = micros() - respTime;
          int CLIENT_NODE = 0;
          for(int i=0; i< mesh.addrListTop; i++){
              if( (int)header.from_node == mesh.addrList[i].address){
                  CLIENT_NODE = mesh.addrList[i].nodeID;
                  NODE_INDEX = i;
                  node_target = mesh.addrList[i].address;
                  break;
              }
          }
          Serial.print("#");
          Serial.print(CLIENT_NODE);
          Serial.print(",");
          Serial.print(enddelay,0);
          Serial.print(",");
          Serial.print(sizeof(respTime));
          Serial.println();
          REQ_TIMER = millis();
          PING_PONG = false;
        }
}
```

h. Kode program Ground station penanganan request data dan ping

```
private void Timerreq_Tick(object sender, EventArgs e) {
          try{
             if (serial.IsOpen){
                Invoke((MethodInvoker)delegate{
                if (dataGridView1.Rows.Count > 1)
```

```
{
    if (rbData.Checked) {
      if (PING_NODE_I < dataGridView1.Rows.Count - 1)
      {
          int node =
          Convert.ToInt16(dataGridView1.Rows[PING_NODE_I].
          Cells[1].Value.ToString());
          String str = "D" + node;
          serial.WriteLine(str);
                PING_NODE_I++;
        }else{
                PING_NODE_I = 0;
        }
    }else{
      if (rb1.Checked){
        Boolean available = false;
        for (int k = 0; k < dataGridView1.Rows.Count - 1; k++)
        {
            int m =
            Convert.ToInt16(dataGridView1.Rows[k].Cells[1].Val
            ue.ToString());
          if (m == 1){
            available = true;
            break;
          }
        }
        if (available){
          String str = "P" + 1;
          SEND_PACKET[0]++;
          serial.WriteLine(str);
        }else{
          if (serial.IsOpen)
```

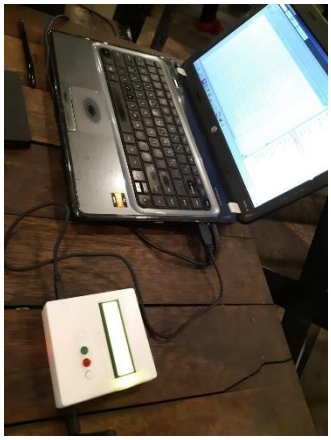
```
        {
            btnReq.Text = "Start";
        }
        timerreq.Stop();
        MessageBox.Show("Node 1 not available.....");
    }
}
else if (rb2.Checked)
{
    Boolean available = false;
    for (int k = 0; k < dataGridView1.Rows.Count - 1; k++)
    {
        int m =
        Convert.ToInt16(dataGridView1.Rows[k].Cells[1].Valu
        e.ToString());
        Console.WriteLine(m);
        if (m == 2)
        {
            available = true;
            break;
        }
    }
    if (available){
        String str = "P" + 2;
        SEND_PACKET[1]++;
        serial.WriteLine(str);
    }else{
        if (serial.IsOpen)
        {
            btnReq.Text = "Start";
        }
        timerreq.Stop();
```

```
        MessageBox.Show("Node 2 not available.....");
    }
}
else if (rb3.Checked)
{
    Boolean available = false;
    for (int k = 0; k < dataGridView1.Rows.Count - 1; k++)
    {
             int m =
            Convert.ToInt16(dataGridView1.Rows[k].Cells[1].
            Value.ToString());
            if (m == 3){
                available = true;
                break;
            }
    }
    if (available)
    {
        String str = "P" + 3;
        SEND_PACKET[2]++;
        serial.WriteLine(str);
    }
    else
    {
        if (serial.IsOpen)
        {
            //serial.WriteLine("S");
            btnReq.Text = "Start";
        }
        timerreq.Stop();
        MessageBox.Show("Node 3 not available.....");
    }
```

```
                    }
                }
                lblNumPacket1.Text = string.Format("{0:0}", SEND_PACKET[0]);
                lblNumPacket2.Text = string.Format("{0:0}", SEND_PACKET[1]);
                lblNumPacket3.Text = string.Format("{0:0}", SEND_PACKET[2]);
            }
        });
    }
    }catch(Exception ex)
    {
    }
}
```

8. *Prototype* yang telah dibuat untuk penelitian thesis






Halaman ini sengaja dikosongkan

DAFTAR INDEX