TUGAS AKHIR – EE 184801

PERENCANAAN PROTOKOL DATA-*LINK LAYER* UNTUK SISTEM KOMUNIKASI RADIO *HIGH FREQUENCY* NVIS BERBASIS USRP

Vina Amalia Fitrianingrum
NRP 07111745000022

Dosen Pembimbing
Prof. Ir. Gamantyo Hendrantoro, M.Eng., Ph.D
Dr. Ir. Achmad Mauludiyanto, MT.

DEPARTEMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro
Institut Teknologi Sepuluh Nopember
Surabaya 2019

[Halaman ini sengaja dikosongkan]

FINAL PROJECT – **EE 184801**

DATA-LINK LAYER PROTOKOL PLANNING FOR HIGH FREQUENCY NVIS RADIO COMMUNICATION SYSTEMS USRP BASED

Vina Amalia Fitrianingrum
NRP 07111745000022

Supervisor
Prof. Ir. Gamantyo Hendrantoro, M.Eng., Ph.D
Dr. Ir. Achmad Mauludiyanto, MT.

ELECTRICAL ENGINEERING DEPARTMENT
Faculty of Electrical Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2019

[Halaman ini sengaja dikosongkan]

PERNYATAAN KEASLIAN TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul **"Perencanaan Protokol Data-*Link Layer* Untuk Sistem Komunikasi Radio *High Frequency* NVIS Berbasis USRP"** merupakan hasil karya intelektual mandiri, diselesaikan menggunakan bahan – bahan yang diijinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua refrensi 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, Juli 2019

Vina Amalia Fitrianingrum
07111745000022

[Halaman ini sengaja dikosongkan]

PERENCANAAN PROTOKOL *DATA-LINK LAYER* UNTUK SISTEM KOMUNIKASI RADIO *HIGH FREQUENCY* NVIS BERBASIS USRP

TUGAS AKHIR

Diajukan untuk Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik
Pada
Bidang Studi Teknik Telekomunikasi Multimedia
Departemen Teknik Elektro
Institut Teknologi Sepuluh Nopember

Menyetujui :

Dosen Pembimbing I	Dosen Pembimbing II
Prof. Ir. Gamantyo H, M.Eng. Ph.D	Dr. Ir. Achmad Mauludiyanto, MT.
NIP. 197011111993031002	NIP. 196109031989031001

SURABAYA
JULI, 2019

[Halaman ini sengaja dikosongkan]

PERENCANAAN PROTOKOL *DATA-LINK LAYER* UNTUK SISTEM KOMUNIKASI RADIO *HIGH FREQUENCY* NVIS BERBASIS USRP

Nama : Vina AmaliaFitrianingrum
Pembimbing 1 : Prof. Ir. Gamantyo Hendrantoro, M.Eng., Ph.D
Pembimbing 2 : Dr. Ir. Achmad Mauludiyanto, MT.

ABSTRAK

Sistem komunikasi radio merupakan sistem yang sedang berkembang. Dalam perkembangannya pemanfaatan Frekuensi tinggi pada sistem komunikasi, merupakan frekuensi yang dapat menjangkau jarak yang jauh dikarenakan pemanfaatan Ionosfer. Jangkauan yang jauh ini menghasilkan potensi positif pada sistem komunikasi darurat di Negara Indonesia. Pemanfaatan lainnya adalah menyampaikan informasi baik berupa suara, ataupun teks. Namun pemanfaatan Ionosfer untuk menyampaikan informasi teks belum dimaksimalkan, dan seharusnya bisa dimanfaatkan dalam keadaan darurat / daerah yang lumpuh dikarenakan bencana. Informasi bencana merupakan informasi yang penting sehingga butuh sistem yang dapat meminimalisir terjadinya informasi yang eror. Untuk memenuhi kebutuhan tersebut, maka dilakukan perancangan dan uji coba suatu protokol Lapisan *Data-Link* untuk *Single Link*. Dengan menggunakan lapisan *Data-Link* terdapat prosedur pengiriman data sehingga karakter/teks berupa *frame* dipahami di lapisan fisik yang menggunakan bit agar dapat ditransmisikan dan informasi disampaikan pada penerima. Dilakukan penyaringan protokol yang sesuai dengan komunikasi radio dan komunikasi dengan kanal HF, maka dipilih AX.25 dikarenakan didalam AX.25 terdapat prosedur pengiriman dengan adanya *error detection* dan menggunakan *error control*. Spesifikasi frekuensi kerja yang digunakan adalah 7-8 MHz. Setelah uji coba perancangan dengan jumlah karakter/*byte* bervariasi, maka terdapat hasil berupa susunan *sub- field* dalam satu *frame* sesuai dengan protokol AX.25 serta dapat menyampaikan informasi dengan baik tanpa ada *error* dikarenakan ada *error detection*, waktu rata-rata 3,16 – 15,09 detik. Dengan modulasi BPSK keseluruhan sistem mendapatkan hasil baik apabila menggunakan *bandwidth* lebih besar dari 30kbps.

Kata kunci : Frekuensi tinggi ,*Data-Link Protocol,AX.25*

[Halaman ini sengaja dikosongkan]

DATA-LINK LAYER PROTOKOL PLANNING FOR HIGH FREQUENCY NVIS RADIO COMMUNICATION SYSTEMS USRP BASED

Name : Vina Amalia Fitrianingrum
Supervisor 1 : Prof. Ir. Gamantyo Hendrantoro, M.Eng., Ph.D
Supervisor 2 : Dr. Ir. Achmad Mauludiyanto, MT.

ABSTRACT

Radio communication system is a developing system. In the development of *High Frequency* utilization in the communication system, it is a frequency that can reach long distances due to the use of the ionosphere. This far-reaching reach positive potential for emergency communication systems in the State of Indonesia. Other uses are conveying information in the form of sound, or text. But the use of the Ionosphere to deliver text information has not been maximized, and should be able to be utilized in emergencies / areas that are paralyzed due to disasters. Disaster information is im*port*ant information so it needs a system that can minimize the occurrence of error information. To meet these needs, the design and trial of a Data-*Link Layer* protocol for Single *Link* was carried out. Using the Data-*Link layer* there is a procedure for sending data so that characters / text in the form of *frame*s are understood in the physical *layer* that uses bits to be transmitted and information is conveyed to the recipient. Protocol screening is done according to radio communication and communication with HF *channel*s, then AX.25 is chosen because in AX.25 there is a shipping procedure with error detection and using error control. The specifications of the working frequency used are 7-8 MHz. After the design test with the number of characters / *byte*s varies, then there are results in the form of sub-*field* arrangements in one *frame* according to the AX.25 protocol and can convey information well without any errors due to error detection, the average time is 3,16 – 15,09 seconds. With BPSK modulation the entire system gets good results when using bandwidth greater than 30kbps.

Keywords **:** *High Frequency, Protocol Data-Link ,AX.25*

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Segala puji dan syukur penulis panjatkan kepad Allah SWT karena berkat rahmat-Nya penulis dapat menyelesaikan tugas akhir yang berjudul **"Perencanaan Protokol Data-*Link Layer* untuk Sistem Komunikasi Radio *High Frequency* NVIS Berbasis USRP".** Penulis mengucapkan banyak terima kasih kepada seluruh pihak yang terlibat dan membantu menyelesaikan karya tulis ini. Oleh karena itu, ucapan terima kasih penulis sampaikan secara khusus kepada:

1. Kedua orang tua dan keluarga besar
2. Bapak Gamantyo Hendrantoro dan Bapak Achmad Mauludiyanto sebagai pembimbing
3. Rekan – rekan Lintas Jalur angkatan 2017, terutama mahasiswa Teknik Telekomunikasi Multimedia
4. Rekan – rekan Tugas Akhir bimbingan Bapak Gamantyo Hendrantoro dan Bapak Achmad Mauludiyanto yang selalu menyemangati dan menemani
5. Sahabat penghuni laboratorium Antena dan Propagasi beserta seluruh kakak tingkat yang selalu mendukung.
6. Pihak lain yang ikut membantu penulis tidak dapat disebutkan namanya satu-persatu

Semoga karya tulis ini dapat bermanfaat bagi penulis sendiri maupun bagi penelitian selanjutnya.

Surabaya, Juli 2019

Vina Amalia Fitrianingrum
NRP. 07111745000022

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

BAB 1
PENDAHULUAN

1.1. Latar Belakang

Negara Indonesia merupakan Negara kepulauan terbesar di Dunia dengan 13.666 pulau yang terpisah dari Sabang hingga Merauke dengan jarak sekitar 8.514 km. Dengan jarak yang tergolong jauh menuntut adanya infrastruktur telekomunikasi yang handal untuk membuat komunikasi jarak jauh maupun dekat menjadi tak ada batasan. Apalagi ketika terjadi kondisi darurat dimana membutuhkan komunikasi darurat, seperti halnya setelah gempa dan tsunami berkekuatan 7,4 SR melanda Kabupaten Donggala Kota Palu dan sekitarnya Jumat (28/9/2018), jaringan telekomunikasi terputus termasuk telepon selular yang mengalami gangguan sinyal. Untuk mendukung koordinasi pihak yang membantu, alat bantu komunikasi sangat dibutuhkan. Maka dari itu Regu Perhubungan dari Detasemen Perhubungan (Denhub) Divif 2 Kostrad turun ke lokasi dan langsung mendirikan instalasi komunikasi, yaitu dengan memanjat tiang milik BMKG setempat untuk memasang single sideband (SSB) dan Instalasi *Repeater*. Lalu penyediaan 38 HT untuk seluruh unsur Satgas digunakan untuk koordinasi antar unsur Satgas[1].

Sistem komunikasi frekuensi tinggi (*High Frequency*) dengan frekuensi 3 - 30 MHz telah dikembangkan sejak tahun 1950-1960[2]. Sistem komunikasi ini dapat dimanfaatkan untuk komunikasi jarak jauh (*long distance*) dikarenakan karakteristik gelombang HF yang dipantulkan oleh lapisan ionosfer atmosfer bumi dimana lapisan ini terionisasi dan mempunyai muatan listrik, sehingga karakteristik ini mendukung kemampuan HF yang dapat menjangkau pulau - pulau atau daerah terpencil yang medannya susah untuk dijangkau komunikasi kabel layaknya Fiber Optik. Teknologi komunikasi HF memiliki beberapa keunggulan seperti murah, mobilitas tinggi dan tangguh, sehingga menarik banyak minat dari para peneliti untuk pengoptimalanan kemampuan dan keandalan perangkat. Pengembangannya tidak hanya dari sisi perangkat yang digunakan tapi juga dari sisi sistem komunikasi untuk pertukaran informasi dan bentuk informasi yang digunakan. Bentuk informasi HF yang sekarang terbatas hanya pada pengiriman suara. Namun seiring berkembangnya jaman, data komunikasi digital menginginkan informasi berupa data teks, citra dan arsip komputer. Komunikasi HF ini penting di negara / tempat yang

kurang berkembang, infrastruktur yang mahal untuk komunikasi model lain[2].

Pada penelitian sebelumnya yaitu sistem komunikasi HF NVIS yang dilakukan oleh Sarah Manalu ITS dengan jarak Surabaya – Merauke[3]. Lalu yang berhubungan dengan pengiriman dan pengolahan data berupa teks menggunakan protokol *data-Link layer* diambil dari penelitian protokol yang menggunakan Modul PAD pada sistem komunikasi ITS-SAT[4]. Didalam tugas akhir inimemiliki hal yang membedakan dengan penelitian sebelumnya, yaitu memanfaatkan sistem komunikasi digital menggunakan USRP dan rancangan sistem diimplementasikan dengan labview. Tugas akhir ini lebih fokus membahas tentang penyusunan prosedur pengiriman data (protokol) *Data-Link layer* dalam komunikasi HF dengan data berupa teks yang dikirimkan demi menunjang kurangnya kesalahan dalam penyampaian pesan. Nantinya sistem komunikasi ini menggunakan perangkat tertanam di komputer yaitu perangkat SDR (*Software Defined Radio*) USRP dengan Labview sebagai perangkat modulator dan demodulator yang dapat di program[2].

1.2. Permasalahan

Permasalahan yang dibahas dalam tugas akhir ini adalah:

1. Bagaimana rancangan protokol *data-Link* untuk sistem komunikasi HF?
2. Bagaimana penerapan protokol yang sesuai dengan karakteristik komunikasi paket radio HF?
3. Dimana penerapan sistem komunikasi HF ini?

1.3. Tujuan

Tujuan yang diharapkan tercapai setelah selesainya tugas akhir ini adalah:

1. Memperoleh rancangan sistem komunikasi untuk pertukaran informasi dengan alur informasi yang dikirim berupa pesan teks dan demikian juga masuk dipenerima.
2. Memperoleh protokol komunikasi yang dapat diimplementasikan dalam komunikasi HF dengan pengimplementasian program protokol menggunakan *user interface* SDR.

1.4. Batasan Masalah

Hal-hal yang akan diperhatikan dalam penelitian ini adalah sebagai berikut:

1. Perancangan menggunakan USRP sebagai perangkat keras SDR, Labview sebagai *software* yang didukung dengan pemrograman dalam matlab untuk membuat suatu protokol. Perangkat keras lain yang digunakan adalah octoclock, *high power amplifier*, *Power Supply* 12V DC, dan LNA.
2. Penerapan protokol *data-Link* menggunakan AX.25 sebagai pengatur data berupa karakter yang akan di kirim disesuaikan dengan bagaimana susunan *Field* di data-*Link layer* dan parameter yang berhubungan dengan protokol layaknya jumlah karakter informasi.
3. Uji coba sistem HF ini dilakukan di gedung ITS Teknik Elektro Surabaya (Antena Gedung AJ dan B) dan Antena AJ - B Teknik Elektro ITS dengan gedung VEDC di Malang dan fokus terhadap komunikasi satu arah.

1.5. Metodologi

Metode yang digunakan dalam tugas akhir ini antara lain meliputi tujuh tahap yaitu: Studi Literatur, Penentuan Protokol yang Sesuai, Perancangan Sistem, Mengaplikasikan Program dan Implementasi Sistem, Pengujian Sistem, Analisa Sistem, serta Penyusunan Buku Laporan Tugas Akhir. Seperti yang terlihat pada rincian dibawah ini:

1. Studi Literatur
   Studi literatur dilakukan dengan mencari dan mempelajari beberapa buku, paper, dan jurnal baik skala nasional maupun internasional yang dapat menunjang tugas akhir ini. Materi - materi yang dibutuhkan mengenai komunikasi HF, protokol komunikasi HF yang bekerja di *data-Link layer*.
2. Penentuan Protokol yang Sesuai
   Berdasarkan studi literatur yang telah dipelajari, ditemukan protokol yang sesuai dengan lapisan *data-Link* dan sesuai dengan sistem komunikasi digital degan kanal HF.
3. Perancangan Sistem
   Perancangan dilakukan setelah penentuan protokol yang ingin digunakan. Dengan memanfaatkan bahasa pemrograman yang mendukung protokol tersebut dan sistem perangkat keras.
4. Mengaplikasikan Program dan Implementasi Sistem

Pengaplikasian program yang telah dirancang menggunakan *software* dengan bahasa pemrograman yang sesuai dengan *data-Link layer* di komunikasi HF.

5. Pengujian Sistem
   Setelah perancangan dan implementasi, berikutnya melakukan tahap pengujian sistem.
6. Analisa Sistem
   Di tahap analisa system ini, menganalisa hasil yang didapat dari pengujian perangkat dengan program yang telah diimplementasikan.
7. Penyusunan Buku Laporan Tugas Akhir
   Setelah mendapatkan hasil bahwa protokol yang sesuai dapat mengirimkan informasi berupa teks, dilakukan pengumpulan data dari proses yang sebelumnya dikumpulkan serta pengumpulan bukti sebagai pendukung hasil, selanjutnya akan disusun buku laporan tugas akhir yang terdiri dari beberapa bagian bab diantaranya:
   Bab 1 tentang Pendahuluan.
   Bab 2 tentang Tinjauan Pustaka.
   Bab 3 tentang Metodologi.
   Bab 4 tentang Analisa dan Penyelesaian Permasalahan.
   Bab 5 tentang Penutup.

1.6. Sistematika

Penulisan Tugas Akhir ini disusun berdasarkan lima bab yang tiap bagiannya membahas permasalahan yang berhubungan dengan Tugas Akhir.

Bab I : Pendahuluan

Bab ini meliputi latar belakang, perumusan masalah, tujuan penelitian, metodologi penelitian, sistematika laporan dan relevansi.

Bab II : Tinjauan Pustaka

Bab ini membahas tinjauan pustaka yang membantu penelitian, diantaranya adalah teori sistem komunikasi HF, teori NVIS, OSI *Layer* dan Protokol Lapisan *Data-Link.*

Bab III : Perancangan Sistem

Bab ini membahas perancangan protokol ke sebuah sistem komunikasi HF menggunakan labview dan USRP sebagai perangkat keras dan

selebihnya membahas tentang penentuan protokol dan men*design* program.

Bab IV : Pengujian dan Analisa
Bab ini meliputi pengujian hasil perancangan sistem komunikasi HF menggunakan protokol lapisan *Data-Link* dan analisa.

Bab V : Penutup
Bab ini merupakan kesimpulan dan saran dari hasil penelitian yang dilakukan.

1.7. Relevansi atau Manfaat

Hasil dari tugas akhir ini dapat memberikan kontribusi dalam penelitian sistem komunikasi radio dengan frekuensi tinggi dan menggunakan *data-Link protokol* sebagai bagian dalam proses suatu sistem komunikasi digital menggunakan *Software Defined Radio* sehingga dapat melewatkan informasi berupa pesan yang dikirim dan diterima.

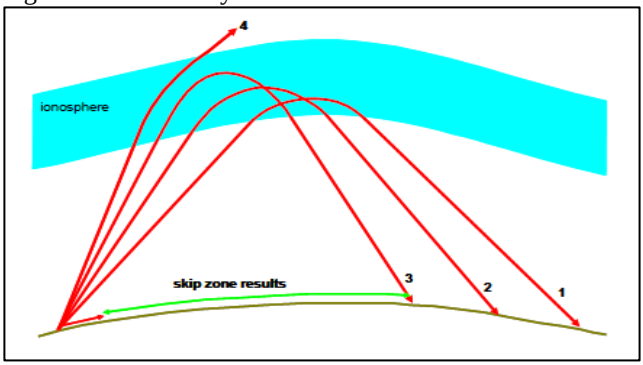
[Halaman ini sengaja dikosongkan]

BAB 2
TINJAUAN PUSTAKA

2.1 *Near Vertical Incident Skywave* (NVIS)

NVIS adalah pancaran radio pada band HF, yang memancar dengan menggunakan sudut pancaran (*Elevation Angle*) yang nyaris tegak lurus atau mendekati 90 derajat, sehingga pancaran sinyal yang dipantulkan lapisan ionosfer jatuh kembali dengan jarak 0 – 400 km. Teknik ini memungkinkan untuk menjangkau area sekitar tanpa mengalami *skip zone* yang biasanya dialami oleh gelombang HF dan daerah yang sulit untuk dijangkau seperti pegunungan dan lembah. Umumnya digunakan untuk komunikasi darurat saat bencana.

Skip Zone sering digunakan apabila yang diinginkan adalah komunikasi yang dikirimkan hanya didengar oleh penerima yang ditentukan. Frekuensi diluar penerima dan jauh dari jangkauan gelombang memungkinkan untuk tidak menerima komunikasi. Namun faktor-faktor seperti *sidescatter* dimana refleksi dari bumi diluar zona menghasilkan gelombang transmit kedalam *skip zone* dapat mempengaruhi keandalannya.



Gambar 2.1 *Frequency fixed* [5]

Frequency Fixed merupakan elevasi rendah sudut panjang *path* yang terbesar (jalur 1). Ketika sudut elevasi dinaikkan, panjang jalur menurun dan sinar dipantulkan dari yang lebih tinggidi ionosfer (jalur 2 dan 3). Jika sudut elevasi dinaikkan melampaui sudut elevasi kritis untuk frekuensi tersebut, maka gelombang menembus ionosfer dan terdapat area di sekitar pemancar yang tidak ada komunikasi gelombang

langit yang dapat diterima (jalur 4). Untuk berkomunikasi dalam "*skip zone*" ini, frekuensi harus diturunkan. Jika sinyal frekuensi tertentu tercermin ketika insiden vertikal pada ionosfer, maka tidak ada *skip zone*.

2.2 Sistem Komunikasi HF

Sistem komunikasi frekuensi tinggi (*High Frequency*) menggunakan gelombang radio frekuensi 3 - 30 MHz. Sistem komunikasi ini dapat dimanfaatkan untuk komunikasi jarak jauh dikarenakan karakter1istik gelombang HF pada saat gelombang dipancarkan dari permukaan bumi akan menghasilkan sudut elevasi tertentu yang nantinya akan dipantulkan oleh ionosfer (*ionospheric wave*) atau gelombang langit (*skywave*). Ionosfer terletak pada ketinggian 50 km sampai 500 km dari permukaan air laut. Ionosfer terbentuk ketika EUV dari matahari melepaskan elektron dari atom netral di atmosfer bumi. Gelombang yang berpropagasi melalui lapisan ionosfer inilah yang akan dipantulkan oleh partikel yang terionisasi. Kondisi yang baik memungkinkan gelombang yang dikirimkan dapat dipantulkan kembali ke bumi pada jarak tertentu dengan kondisi gelombang yang tidak tembus ke luar angkasa.

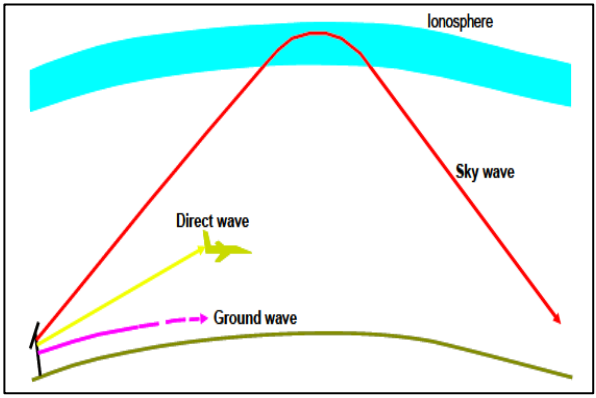
Pada *groundwave*, gelombang radio HF merambat dari sisi pengirim ke penerima melalui pemantulan objek-objek yang ada di permukaan tanah. Jarak tempuh pada propagasi *groundwave* tersebut bergantung pada konduktivas, permeabilitas, tinggi *Antenna* polarisasi *Antenna* dan topografi permukaan bumi yang dirambatinya.

Pada tipe propagasi *skywave*, gelombang radio HF mengalami proses pemantulan pada lapisan ionosfer dan dapat menjangkau jarak hingga ribuan kilometer. Pada tipe propagasi ini dapat menjelaskan mengenai berapa kali pantulan yang terjadi di ionosfer terhadap gelombang radio yang disebut dengan *hop*.

Untuk propagasi *direct wave (Line of Sight)*. Tipe propagasi ini menjangkau jarak lebih jauh daripada tipe propagasi *groundwave*. Gelombang pada tipe propagasi ini berinteraksi dengan gelombang pantul bumi tergantung pada pelepasan terminal, frekuensi, polarisasi, dan daerah yang dirambatinya [5].

Pada kenyataannya di antara *transmitter* dan *receiver*, setidaknya paling sedikit mempunyai 2 *raypath*. Keberhasilan sistem komunikasi HF sangat bergantung dari kondisi ionosfer, dikarenakan kondisi ionosfer berubah ubah dan keberhasilan sistem komunikasi dapat

dijamin dengan cara mencari penentuan frekuensi kerja yang baik, dilakukan berdasarkan penelitian frekuensi plasma yang menentukan frekuensi terendah dan tertinggi dari pemantulan gelombang radio yang dapat terjadi[2].



Gambar 2.2 Model Propagasi Geomatrikal[5]

Sistem komunikasi radio HF dengan NVIS sendiri memiliki teknik pemancaran gelombang radio HF hampir tegak lurus ke atas dengan sudut elevasi 65^0 sampai dengan 90^0. Dengan adanya sudut elevasi antena yang hampir tegak lurus, maka teknik NVIS dapat diimplementasikan dalam sistem radio HF untuk mengatasi dan menangani efek *skip zone*. Keberhasilan dalam mengimplementasikan sistem radio NVIS ini dipengaruhi oleh 3 faktor, yaitu sudut elevasi, pemilihan frekuensi, dan besar keluaran daya pada sisi pemancar. NVIS di lingkungan radio amatir biasanya menggunakan pita frekuensi 7 MHz pada siang hari dan 3.6 MHz pada malam hari. Adapun beberapa kelebihan dari sistem radio HF dengan menggunakan teknik NVIS ialah sebagai berikut:

1. Daratan rendah atau lembah tidak menjadi masalah bagi propagasi dengan teknik NVIS.
2. Tidak membutuhkan infrastruktur seperti *repeater* karena dua buah stasiun yang menggunakan teknik NVIS sudah dapat membangun sistem komunikasi yang handal tanpa dukungan dari pihak ketiga.
3. Propagasi murni NVIS relatif bebas terhadap *fading*.

Teknik NVIS dapat mengurangi *noise* dan interferensi, sehingga dapat meningkatkan besar *Signal-to-Noise Ratio* (SNR).

Mentransmisikan sinyal informasi dari satu tempat ke tempat yang lain membutuhkan media transmisi berupa gelombang radio. Dapat kita ketahui gelombang radio dapat dikirimkan dengan frekuensi yang diklasifikasi dari spektrum frekuensi 3 kHz – 300 GHz, walau memungkinkan gelombang radio bekerja pada frekuensi bawah 30 kHz.

Tabel 2.1 Pembagian *Band Frequency*

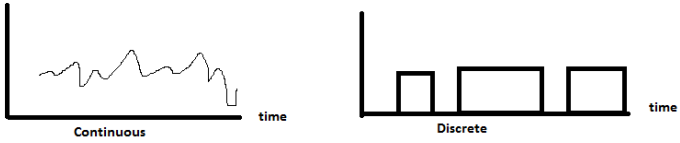
Frequency Band	Frequency Range
Extremely low frequency (ELF)	<3KHz
Very low frequency (VLF)	3-30 kHz
low frequency (LF)	30-300 kHz
Medium Frequency (MF)	300 kHz – 3MHz
High Frequency (HF)	3 -30 MHz
Very *High Frequency* (VHF)	30 – 300 MHz
Ultra *High Frequency* (UHF)	300 Mhz – 3 GHz
Super *High Frequency*(SHF)	3-30 GHz
Extra *High Frequency* (EHF)	30 – 300 GHz

2.3 Dasar Komunikasi Data

Dalam sistem komunikasi data, data ditransfer dari satu poin ke poin lainnya dalam bentuk sinyal elektronik. Terdapat 2 tipe sinyal yaitu analog dan digital :

1) Sinyal Analog

   Merupakan gelombang – gelombang elektronik yang bervariasi dan kontinyu, mempunyai nilai yang bervariasi, ditransmisikan melalui beragam media, bergantung frekuensi dan perubahan nilai seiring dengan waktu, hal tersebut dapat dilihat pada Gambar 2.3



(a) (b)

Gambar 2.3 (a) Sinyal Analog dan (b) Sinyal Digital

2) Sinyal Digital

Sinyal digital merupakan denyut-denyut tegangan yang mempunyai 2 buah nilai tetap yang direpresentasikan dengan bit "0" dan bit "1". Contoh sinyal digital ditunjukkan pada Gambar 2.3 b. Penggunaan sinyal analog ataupun digital disesuaikan dengan peralatan yang akan digunakan. Pada komunikasi yang menggunakan gelombang radio, data dimodulasi untuk kemudian dapat dikirimkan melalui gelombang radio. Sedangkan disisi penerima sinyal gelombang radio akan didemodulasi untuk memperoleh data yang dikirimkan.

Dalam suatu transmisi data, data harus diatur dalam mode komunikasi yang digunakan. Mode komunikasi dibagi ke dalam dua hal yaitu :

1) Mode Komunikasi Paralel

Data dalam mode komunikasi ini dikirimkan secara bersama-sama tanpa adanya urutan pengiriman. Dalam mode ini pengiriman data menjadi lebih cepat karena semua data dikirimkan secara bersamaan. Namun mode ini umumnya digunakan pada komunikasi yang mempunyai jarak pengiriman yang tidak terlalu jauh.

2) Mode Komunikasi Serial

Pada mode ini semua data dikirimkan dengan urutan pengiriman. Data akan dikirim secara serial atau per bit. Mode komunikasi ini secara umum digunakan pada komunikasi yang mempunyai jarak pengiriman yang jauh.

a. Mode Serial Sinkron

Mode transmisi sinkron ini merupakan mode dimana sebelum terjadi komunikasi, adanya sinkronisasi *clock* antara pengirim dan penerima. Data dikirim dalam satu blok data (*Frame*) yang berisi bit-bit pembuka (*preamble* bit), bit data / informasi itu sendiri dan bit-bit penutup (*postamble* bit). Ditambahkan pula bit-bit kontrol. Variasi ukuran *Frame* mulai 1500 *byte* sampai 4096 *byte*. Dalam komunikasi ini dapat membawa data sampai 7000 *byte* per detik. Contohnya adalah Ethernet.

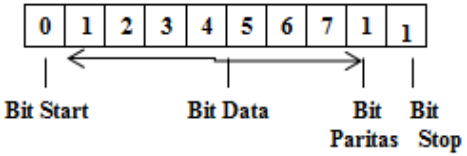
***Flag* 8 bit**	***Control***	**Data *Field***	***Control***	***Flag* 8 bit**

Gambar 2.4 Bentuk *Frame* komunikasi serial sinkron

b. Mode Serial Asinkron

Pada transmisi asinkron, sebelum terjadi komunikasi tidak diadakan sinkronisasi *clock* antara pengirim dan penerima. Data dikirim per-karakter dan masing-masing karakter memiliki bit *start* (biasanya 0) dan bit *stop* (biasanya 1).

Start bit berfungsi untuk menandakan adanya rangkaian bit karakter yang akan dicuplik. *Stop* bit berfungsi melakukan proses menunggu karakter berikutnya. Contoh dari transmisi asinkron adalah RS-232 dan USB. Setiap karakter terdiri dari 10 bit dengan rincian seperti berikut :

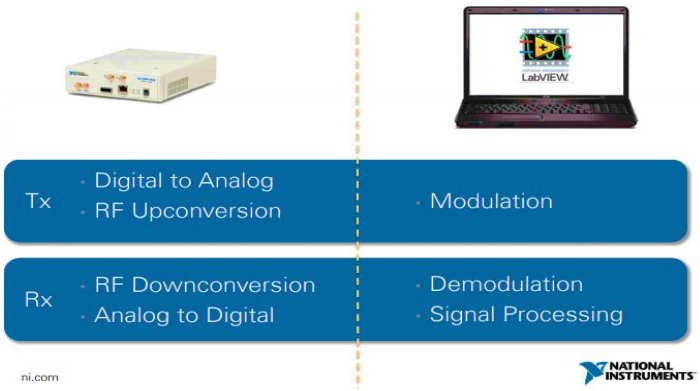


Gambar 2.5 Karakteristik bit pada komunikasi serial asinkron.

2.4 *Software-Defined Radio (SDR)*

Software-Defined Radio atau SDR adalah sistem komunikasi radio yang komponen perangkat kerasnya digantikan dengan implementasi perangkat lunak pada komputer. Komponen-komponen seperti mixer, filter, modem, detector, dan lainnya diimplementasikan sebagai *software* pada *personal computer* atau *embedded sistem*. Contoh aplikasi yang berhubungan dengan SDR adalah Labview dengan perangkat USRP (*Universal Software Radio Peripheral)* dari NI (National Instruments).

Universal Software Radio Peripheral (USRP) adalah SDR yang didesain dan dijual oleh Ettus Research dan perusahaan induknya, yaitu National Instrument. Sebagian besar USRP terhubung ke host computer melalui kabel dengan kecepatan tinggi dimana perangkat lunak pada *host computer* digunakan untuk mengatur data pemancar dan penerima pada perangkat keras USRP. Biasanya USRP menggunakan perangkat lunak Labview atau GNU Radio untuk membuat sistem SDR yang kompleks.



Gambar 2.6 *SDR Components from* NI[8]

Komponen SDR dari NI dapat dilihat bahwa (sebelah kiri) terdapat perangkat USRP, dengan Tx melakukan *digital to analog* dan *RF Upconversion* dan dari sisi Rx melakukan *RF Downconversion* dan *Analog to Digital*. Disisi *software* LabView terjadi proses modulasi dan demodulasi[8]. Pemilihan perangkat USRP adalah dari sisi *daughterboard* dengan penggunaan frekuensi yang diinginkan, salah satunya yaitu apabila menggunakan frekuensi tinggi dapat menggunakan *daughterboard* LFRX/LFTX dengan frekuensi 0 – 30MHz.

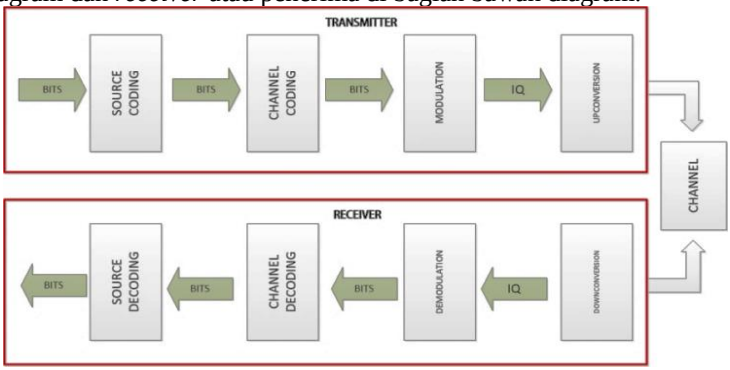
Tabel 2.2 Frekuensi Kerja *Daughterboard* USRP

Daughterboard	*Frequency Coverage*
WBX-120	50 MHz – 2.2 GHz
SBX-120	400 MHz - 4.4 GHz
CBX-120	1.2 GHz - 6 GHz
UBX-120	10 MHz - 6 GHz
WBX	50 MHz - 2.2 GHz
SBX	400 MHz - 4.4 GHz
CBX	1.2 GHz - 6 GHz
UBX-40	10 MHz - 6 GHz
TVRX2	50 MHz - 860 MHz
DSBRX2	800 MHz - 2.3 GHz
BasicRX / BasicTX	1 - 250 MHz
LFRX / LFTX	0 - 30 MHz

Gambar 2.7 *Daughter Board* LFTX dan LFRX

Pada uji coba kali ini menggunakan USRP N210 produk Ettus Research yang didalamnya terdapat sistem ADC/DAC yang berfungsi sebagai konversi sinyal analog ke digital atau digital ke analog. Selain itu terdapat *motherboard* dan *daughterboard* LFTX/LFRX sebagai pemancar atau penerima gelombang radio dan juga sebagai akuisisi data, *chip* FPGA untuk melakukan proses sebelum pengolahan sinyal *input* berbasis komputasi, dan *port* untuk koneksi ke pc /laptop menggunakan *Gigabit Ethernet*. Perangkat ini juga dilengkapi dengan enam buah lampu indikator yang menjadi parameter apakah USRP sudah beroperasi atau belum seperti led A akan menyala apabila terjadi *transmitting*, C apabila terjadi penerimaan.

Dasar sistem komunikasi digital menurut USRP sendiri terdiri dari *transmitter* (pengirim), *receiver* (penerima), dan kanal komunikasi. Gambar 2.8 mengilustrasikan tentang komponen dari sistem komunikasi digital. *Transmitter* atau pengirim terdapat pada bagian atas dari diagram dan *receiver* atau penerima di bagian bawah diagram.



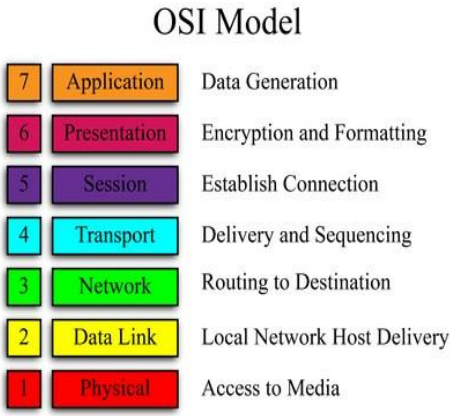
Gambar 2.8 Dasar Sistem Komunikasi Digital

Pada *transmitter* terdapat blok untuk sumber dan *channel coding*, setelah itu terdapat proses modulasi, simulasi penurunan nilai sinyal , dan *up conversion*. Pada *receiver* yang didalamnya termasuk *block* untuk *down conversion*, *matched filter*, *equalization*, demodulasi, *decoding* kanal dam sumber *decoding*.

2.5 *OSI Model*

OSI (*Open System Interconnection*) *Layer* adalah model jaringan yang terstruktur dan dikembangkan oleh badan Internasional Organization for Standarization (ISO). Terdapat 7 *layer* yaitu *Physical, data Link, network, transport, session, presentation* dan *application*.

1) *Physical Layer*
   Layer pertama dalam susunan model OSI *layer* bertanggung jawab atas proses data menjadi bit dan mentransfernya melalui media.
2) *DataLink Layer*
   Layer ini menyediakan *Link* untuk data dan memaketkannya menjadi *Frame*. Pada *layer* ini juga melakukan sinkronisasi, penanganan *eror* dan *flow Control.*
3) *Network layer*
   Layer ini bertanggug jawab menentukan alamat jaringan, menentukan rute yang harus diambil selama perjalanan dan menjaga antrian trafik di jaringan. Data pada *layer* ini berbentuk paket.
4) *Transport layer*
   Layer ini bertanggung jawab membagi data menjadi segmen, menjaga koneksi logika "*end-to-end*" antar terminal dan menyediakan penanganan eror (*eror handling*).
5) *Session layer*
   Layer ini menentukan bagaimana dua terminal menjaga, memelihara dan mengatur koneksi, bagaimana mereka saling berhubungan satu sama lain. Koneksi di *layer* ini disebut "*session*"
6) *Presentation layer*
   Layer ini bertanggung jawab bagaimana data dikonversi dan diformat untuk *transfer* data.
7) *Application layer*
   Layer ini menyediakan jasa untuk aplikasi pengguna. *Layer* ini bertanggung jawab atas pertukaran informasi antara program computer.



Gambar 2.9 OSI Model

2.6 *Data-Link Control Protokol*

Data-Link Control protokol merupakan fungsi seperti *flow Control*, *error detection*, dan juga *error Control*. Didalam data-*Link Control* protokol, *error protokol* didapatkan dari transmisi ulang *Frame* yang rusak yang belum diketahui dan meminta retransmisi lagi. HDLC (*High-Level Data-Link Control*) adalah data-*Link* protokol yang sering digunakan, mengandung hampir semua fitur yang ditemukan dalam protokol kontrol data *Link* lainnya.

2.6.1 *Flow Control*

Flow Control merupakan teknik untuk memastikan bahwa entitas yang mentransmisikan tidak melebihi entitas penerima dengan data. Terdapat dua *flow Control* yaitu *Stop-and-wait Flow Control* dengan *Sliding Window Flow Control*.

1. *Stop-and-wait Flow Control*

   Cara kerja *Stop-and-wait Flow Control* adalah entitas sumber mentransmisikan *Frame*. Setelah entitas tujuan menerima *Frame*, maka menunjukkan kesediaannya untuk menerima *Frame* lain dengan mengirimkan kembali pemberitahuan ke *Frame* yang baru saja diterima. Pengirim harus menunggu sampai menerima *acknowledgment* sebelum mengirim *Frame* berikutnya. Dengan demikian tujuan dapat menghentikan aliran data hanya dengan

menahan pengakuan. Prosedur ini berfungsi dengan baik dan, hampir tidak dapat diperbaiki ketika pesan dikirim dalam beberapa *Frame* besar. Namun, sering kali suatu sumber akan memecah blok data yang besar menjadi blok yang lebih kecil dan mengirimkan data dalam banyak *Frame*. Hal ini dilakukan karena alasan ukuran *buffer receiver* terbatas. Semakin lama waktu pengiriman, semakin besar kemungkinan akan adanya kesalahan, yang mengharuskan pengiriman ulang seluruh *Frame*[6].

2. *Sliding Window Flow Control*

Inti dari masalah yang diuraikan sejauh ini adalah bahwa hanya satu *Frame* pada satu waktu yang dapat transit. Dalam situasi di mana panjang bit *Link* lebih besar dari panjang *Frame*, hasil ketidakefisienan. Efisiensi dapat sangat ditingkatkan dengan memungkinkan beberapa *Frame* dalam perjalanan pada saat yang sama. Untuk melacak *Frame* mana yang telah diakui, masing-masing diberi label dengan nomor urut[6].

2.6.2 *Error Control*

Mengacu pada mekanisme untuk mendeteksi dan memperbaiki kesalahan yang terjadi dalam transmisi *Frame*. Mekanisme ini disebut dengan ARQ (*Automatic Repeat Request*). Efek dari ARQ adalah untuk mengubah data *Link* yang unreliable menjadi data*Link* yang reliable. Terdapat 3 versi ARQ yang sudah di standarisasi yaitu *Stop-and-Wait ARQ, Go-Back-N ARQ,* dan *Selective-reject ARQ*[6].

1. *Stop and Wait* ARQ

Stasiun pengirim mentransmisikan satu *Frame* dan harus menunggu *acknowledgment* (ACK). Tidak ada *Frame* data lain yang dapat dikirim sampai balasan stasiun tujuan tiba di stasiun sumber. Dalam hal ini terdapat 2 macam eror yang terjadi. Pertama, *Frame* yang datang di tempat tujuan bisa rusak. Penerima mendeteksi ini dengan menggunakan teknik *error detcetion* dan hanya membuang *Frame*. Untuk menjelaskan kemungkinan ini, *transmitter* dilengkapi dengan t*imer*. Setelah *Frame* ditransmisikan, *transmitter* menunggu untuk *acknowledgment*. Jika tidak ada *acknowledgment* yang diterima pada saat *timer* berakhir, maka *Frame* yang sama dikirim lagi. Jenis kesalahan kedua adalah *acknowledgment* yang rusak.

Stasiun A mengirim *Frame* dan diterima dengan benar oleh stasiun B, yang merespons dengan *acknowledgment* (ACK). ACK rusak dalam perjalanan dan tidak dapat dikenali oleh A oleh karena itu, akan habis dan mengirim ulang *Frame* yang sama[6].

2. *Go-Back-N* ARQ

Dalam metode ini, *transmitter* dapat mengirim serangkaian *Frame* berurutan modulo beberapa nilai maksimum dengan menggunakan *sliding-window flow Control technique*. Meskipun tidak ada kesalahan yang terjadi, tujuan akhir akan mengakui *Frame* masuk seperti biasa (RR = *receive ready*). Jika stasiun tujuan mendeteksi *error* dalam sebuah *Frame*, mungkin stasiun akan mengirim acknowledgment negatif (REJ = *reject*) untuk *Frame* itu, seperti dijelaskan dalam aturan berikut. Stasiun tujuan akan membuang *Frame* itu dan semua *Frame* yang masuk di masa depan sampai *Frame* dalam kesalahan diterima dengan benar. Dengan demikian, *reciever*, ketika menerima REJ, harus mentransmisikan ulang *Frame* dalam *error* ditambah semua *Frame* berikutnya yang ditransmisikan untuk sementara.

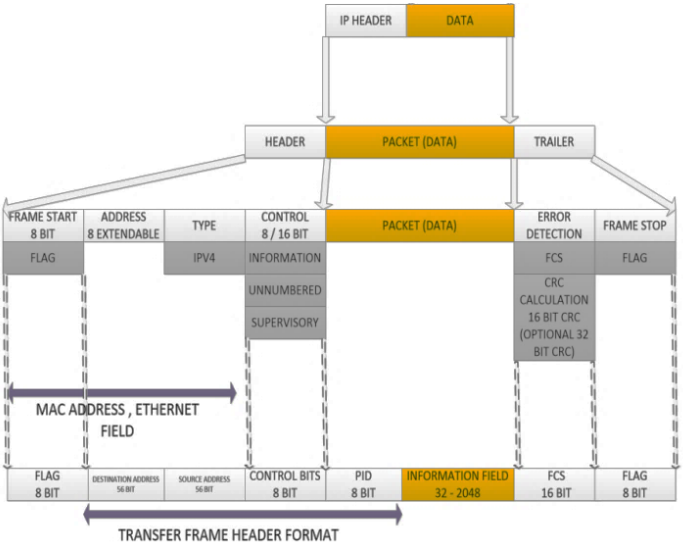
3. *Selective-Reject* ARQ

Dengan ARQ yang ditolak secara selektif, satu-satunya *Frame* yang dikirim ulang adalah mereka yang menerima *acknowledgment* negatif, dalam hal ini, disebut SREJ, atau mereka yang *time out*. Selective reject lebih efisien daripada *go-back-N* karena meminimalkan jumlah pengiriman ulang. Di sisi lain, penerima harus memelihara buffer yang cukup besar untuk menyimpan *Frame post* SREJ sampai *Frame* yang salah dikirimkan kembali dan harus mengandung logika untuk memasukkan kembali *Frame* tersebut dalam urutan yang benar. *Transmitter* juga membutuhkan logika yang lebih kompleks untuk dapat mengirim keluar dari urutan.

2.7 Protokol *AX.25*

AX.25 (*Amateur AX.25*) merupakan protokol lapisan 2 (merujuk pada OSI *Layer* Reference) yaitu Data *Link Layer* sesuai dengan Gambar 2.10. Sebagai protokol *layer* 2 AX.25 bertanggungjawab untuk menbangun *Link connection*, menyediakan prosedur logic untuk information transfer, dan *Link disconnection*. Protokol AX.25 sedikit

berbeda dengan protokol lain, karena pada protokol ini menghilangkan perbedaan kelas dari perangkat yang terrhubung oleh pada *Link*, sedangkan beberapa protokol pada *layer* data*Link* membedakan kelas dari perangkat yang terhubung[7]. Sebagai protokol yang bertanggung jawab dalam membangun dan memutuskan *Link* dan mengirim informasi, protokol ini dapat digunakan dalam sistem yang bekerja pada frekuensi tinggi yang mempunyai *noise*. Protokol ini mengikuri prinsip – prinsip yang terdapat dalam *International Standards Organization* (ISO) Information Standards (IS) 3309, 4335 dan 7809 tentang HDLC.



Gambar 2.10 Struktur paket dalam *Network* dan *DataLink Layer*

Dua lapisan bawah dari OSI *Layer* yaitu *data-Link layer* dan *physical layer*, dapat dibagi lagi menjadi fungsi *Layer* yang berbeda seperti yang ditunjukkan pada Gambar 2.10.

Tabel 2.3 Fungsi *Layer* Data*Link* dan *Physical SingleLink*[7]

Layer	**Function(s)**	
Data *Link* (2)	Segmenter	Management Data *Link*
	Data *Link*	
	Link Multiplexer	
Physical (1)	Physical	
	Silicon Radio	

Protokol AX25 merupakan protokol untuk melakukan akses jaringan sinkron antara DTE (*Data Terminal Equipment*) pada sisi pemakai dan DCE (*Data Circuit Terminating Equipment*) yang merupakan peralatan yang berada pada sisi jaringan yang lansung berhubungan dengan sisi pemakai. *Link* level, merupakan *layer* yang mempunyai aturan untuk bertukar data / data*Link Control*. Protokol yang dipakai pada lapisan ini disebut HDLC (High Level Data*Link Control*) yang melakukan beberapa hal yaitu:

1. Membangun hubungan logik melalui media yang ada seperti kabel atau atmosfer
2. Memberikan informasi mengenai perpindahan data agar data tetap pada urutannya
3. Melakukan pendeteksian kesalahan
4. Menutup hubungan logik yang telah selesai digunakan.

Pada masing-masing *layer* dibedakan menjadi beberapa *Finite State Machine* (FSM). Pada *layer physical* terdiri dari *physical* dan radio, sedangkan untuk *layer data Link* terdiri dari segmenter, *Management Data Link*, *Data Link*, dan *Link Multiplexer*. Fungsi dari masing masing FSM adalah:

1. Segmenter

Segmenter *State Machine* befungsi sebagai penerima masukan dari lapisan yang lebih tinggi.

2. *Management Data Link*

Management Data Link State Machine memiliki fungsi menyediakan parameter operasional yang digunakan antara dua stasiun. Untuk mendukung suatu fungsi khusus yang melakukan beberapa operasi secara bersamaan, sehingga dalam satu *layer data Link* bisa terdapat beberapa *Management data Link state Machine*.

3. *Data Link*

Data Link state Machine merupakan penyedia prosedur yang digunakan untuk membangun dan memutus koneksi serta tukar menukar informasi antar dua stasiun, dan *data Link state Machine* dinyatakan sebagai pusat dari protokol AX.25.

4. *Link Multiplexer*

Tugas utama dari *Link Multiplexer state Machine* adalah melewatkan satu atau lebih *data Link* dalam saluran yang sama.

5. *Physical*

Sebuah *Physical State Machine* terdapat dalam satu *layer physical. Physical State Machine* digunakan untuk menyatakan pemancar dan penerima dari sebuah stasiun radio.

2.7.1 Struktur *Frame AX.25*

Dalam pengirimannya struktur *Frame* AX.25 mengirimkan data dalam bentuk *Frame* dan tiap *Frame* tersusun dari bagian - bagian yang lebih kecil (*Field*). Setiap *Field* memiliki fungsi khusus dan terdiri dari jumlah *byte*. Dalam protokol AX.25 terdapat 3 tipe *Frame*:

1. *Frame* Informasi
   Fungsi *Frame* ini adalah untuk membawa data informasi yang dikirim oleh pengirim dan diterima oleh penerima.
2. *Frame* Pengawas atau Supervisory *Frame* (*Frame* S)
   Fungsi *Frame* ini adalah untuk melakukan pengendalian *Link*.
3. *Frame* tidak bernomor atau Unnumbered *Frame* (*Frame* U)
   Fungsi *Frame* ini adalah untuk melakukan pengendalian *Link* yang belum dilakukan oleh *Frame* pengawas seperti membangun hubungan.

Tabel 2.4 Bentuk *Frame* U dan S [7]

Flag	***Address***	***Control***	**Info**	**FCS**	***Flag***
8 bits	112/224 bits	8/16 bits	N*8 bits	16 bits	8 bits

Tabel 2.5 Bentuk *Frame* Informasi [7]

Flag	***Address***	***Control***	**PID**	**Info**	**FCS**	***Flag***
8 bits	112/224 bits	8 / 16 bits	8 bits	N*8 bits	16 bits	8 bits

Fungsi dari masing – masing *Field* adalah sebagai berikut:

1. *Field Flag*

Field Flag memiliki fungsi untuk menentukan awal dan akhir dari suatu *Frame*. Hal ini memudahkan penerima untuk mengidentifikasi suatu *Frame*. *Field* ini berisi satu *octet* yang bernilai 01111110 (7E hexa). Urutan bit *Field Flag* tidak diperbolehkan muncul pada bagian manapun dalam *Frame* yang telah lengkap. Untuk itu digunakan fasilitas bit stuffing yang setiap kali terdapat lima bit "1" muncul maka bit selanjutna diset 0.

2. ***Field Address***

Fungsi dari *Field* ini adalah sebagai pengalamatan dalam komunikasi. *Field* ini dapat mengidentifikasi stasiun pengirim, dan

pemancar. Masing-masing sub*Field* alamat pengirim dan penerima terdiri dari tujuh *octet*, enam diawal merupakan *callsign* yang merupakan susunan karakter abjad dan karakter *American Standard Code for Information Interchange* (ASCII), dan *octet* ketujuh merupakan *Secondary Station Identifier* (SSID) yang menyatakan nomor terminal dalam suatu stasiun.

Tabel 2.6 *Address* Sub*Field*

Address SubField						
A 1	A 2	A 3	A 4	A 5	A 6	A 7
Call Sign						SSID

Pada Tabel diatas A1 sampai dengan A6 merupakan *octet* yang berisi *callsign*, dan A7 merupakan *octet* yang berisi SSID. Contohnya adalah Tabel 2.7 menunjukan *octet* A1 merupakan *octet* yang dikirim terlebih dahulu.

Tabel 2.7 *Address Field*

Octet	**ASCII**	**Bin Data**	**Hex Data**
A1	N	10011100	98
A2	J	10010100	94
A3	7	01101110	6E
A4	P	10100000	A0
A5	Space	01000000	40
A6	Space	01000000	40
A7	SSID	11100000	E0
A7	SSID	CRRSSID0	

Pada *Callsign octet* terakhir diisi dengan spasi karena jumlah *callsign* kurang dari 6 karakter. Bit "R" merupakan bit cadangan, menjadi 1 apabila tidak digunakan. Sedangkan bit "C" merupakan bit perintah dari *Frame*. Dalam *Field address* terdapat 2 mode yaitu "Non-*repeater address Field encoding*" dan mode" *repeater address Field* encoding".

a. Mode non-*repeater Address Field encoding*

Mode ini digunakan saat *repeater* tidak digunakan. Pada mode ini *Field* alamat terdiri dari 14 oktet yang kemudian dibagi menjadi 2 sub*Field* alamat yaitu sub*Field* alamat tujuan dan sub*Field* alamat sumber.

Tabel 2.8 *Address Field* of *Frame*

Address Field of Frame													
Sub*Field* Alamat Tujuan							Sub*Field* Alamat Sumber						
A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14

Alamat tujuan terdiri dari *octet* A1 sampai dengan A7. *Octet* A1 sampai A6 merupakan *callsign* amatir dan *octet* A7 merupakan SSID dari stasiun radio amatir yang dikirimkan *Frame*. Sedangkan untuk alamat sumber terdiri dari A8 sampai dengan A14. Dimana A8 sampai A13 merupakan *callsign* amatir dan *octet* A14 merupakan SSID dari stasiun yang mengirim *Frame*. Pada Tabel 2.9 merupakan contoh *Frame* AX.25 dengan mode non-*repeater* menggunakan *Frame* Information (I).

Tabel 2.9 *Frame* AX.25 mode non-*repeater*

Octet	ASCII	Bin Data	Hex Data
Flag		01111110	7E
A1	N	10011100	98
A2	J	10010100	94
A3	7	01101110	6E
A4	P	10100000	A0
A5	Space	01000000	40
A6	Space	01000000	40
A7	SSID	11100000	E0
A8	N	10011100	98
A9	7	01101110	6E
A10	L	10011000	98
A11	E	10001010	8A
A12	M	10011010	9A
A13	Space	01000000	40
A14	SSID	01100001	61
Control	1	00111110	3E
PID	None	11110000	F0
FCS	Part1	XXXXXXXX	HH
FCS	Part2	XXXXXXXX	HH
Flag	Part2	01111110	7E

b. Mode *Repeater Address Field* encoding

Mode ini digunakan pada pengiriman *Frame* melalui *repeater*. Pada mode ini alamat *repeater* ditambahkan dalam *address Field* sehingga bertambah menjadi 21 oktet yang terbagi menjadi 3 sub*Field*.

Tabel 2.10 Mode *Repeater Address Field* [6]

Address Field of Frame		
Sub*Field* Alamat Tujuan	Sub*Field* Alamat Sumber	Sub*Field* Alamat *Repeater*
A1 A2 A3 A4 A5 A6 A7	A8 A9 A10 A11 A12 A13 A14	A15 A16 A17 A18 A19 A20 A21

Alamat tujuan dikodekan dengan A1 hingga A7 dimana A1 sampai A6 adalah *callsign* amatir dan A7 adalah SSID stasiun. Alamat sumber dikodekan dengan A8 hingga 14 dimana A8 hingga A13 merupakan *callsign* amatir dan A14 merupakan SSID stasiun, alamat *repeater* dikodekan dengan A15 sampai dengan A21, dimana A15 sampai A20 emerupakan *callsign* amatir dan A21 merupakan SSID stasiun. Protokol AX25 dapat menggunakan 2 *repeater* dengan panjang *Field address* 28 oktet.

Tabel 2.11 *Frame* protokol AX.25 dengan satu alamat *repeater*

Octet	ASCII	Bin Data	Hex Data
Flag		01111110	7E
A1	N	10011100	98
A2	J	10010100	94
A3	7	01101110	6E
A4	P	10100000	A0
A5	Space	01000000	40
A6	Space	01000000	40
A7	SSID	11100000	E0
A8	N	10011100	98
A9	7	01101110	6E
A10	L	10011000	98
A11	E	10001010	8A
A12	M	10011010	9A
A13	Space	01000000	40
A14	SSID	01100001	61
A15	N	10011100	98
A16	7	01101110	6E
A17	0	10011110	9E
A18	0	10011110	9E
A19	Space	01000000	40
A20	Space	01000000	40
A21	SSID	11100011	E3
Control	I	00111110	3E

Octet	ASCII	Bin Data	Hex Data
PID	None	11110000	F0
FCS	Part1	XXXXXXXX	HH
FCS	Part2	XXXXXXXX	HH
Flag	Part2	01111110	7E

3. *Field Control*

Field ini berfungsi untuk mengidentifikasi tipe *Frame* yang digunakan pada koneksi *layer* 2. *Field* terdiri dari 8 atau 16 bit. Dimana pada AX.25 terdapat tiga format *Control Field*, yaitu:

a. Information *Field* (I *Frame*)

Semua *Frame* Informasi memiliki bit 0 dari *Control Field* diatur ke "0". Pada *Frame* ini terdapat N(S) dan N(R), dimana N(S) adalah nomor urut pengiriman dari pengirim. Dan N(R) merupakan nomor penerimaan.

b. Supervisory *Field* (S *Frame*)

Frame supervisory memiliki bit 0 dari *Control Field* diatur ke "1" dan bit 1 dari *Control Field* diatur ke "0". *Frame* S menyediakan Supervisory *Link Control*.

c. Unnumbered *Field* (U *Frame*)

Unnumbered *Frame* kedua bit 0 dan 1 dari *Control Field* diatur ke "1". *Frame* U bertanggung jawab untuk menjaga *Control* tambahan atas *Link* melampaui apa yang dicapai dengan *Frame* S. *Frame* U bertanggung jawab untuk membangun dan mengakhiri koneksi *Link*.

Tabel 2.12 Bentuk *Control Field* (modulo 8)[1]

Control Field **Type**	***Control Field*** **Bits**							
	7	6	5	4	3	2	1	0
I *Frame*	N(R)			P	N(S)			0
S *Frame*	N(R)			P/F	S	S	0	1
U *Frame*	M	M	M	P/F	M	M	1	1

Tabel 2.12 merupakan dasar dari *Control Field* yang terkait dengan masing masing tiga jenis *Frame*. *Control Field* dapat menjadi satu atau dua *octet* panjang dan dapat menggunakan nomor urut untuk menjaga keutuhan *Link*. Nomor urut ini dapat berupa tiga-bit (modulo 8) seperti table 2.13 atau tujuh-bit (modulo 128). Pertama, bit yang dikirim adalah bit 0, sedangkan untuk bit yang terakhir dikirim adalah bit ke 7 untuk modulo 8 dan bit ke 15 untuk modulo 128. Bit "S" menyatakan *Frame*

supervisory dan bit "M" menyatakan *Frame* unnumbered, dan bit P/F *(Poll/Final)* digunakan pada semua jenis *Frame*.

4. Protokol Identifier Field

Field PID *(Protokol Identifier)* berfungsi untuk identifikasi model *layer* 3. *Field* PID muncul dalam *frame* informasi saja (I dan UI). Encoding dari PID yang didefinisikan adalah 0XF0 dalam hexa dan "11110000" dalam biner.

5. Information Field

Field I secara *default* memiliki panjang 256 oktet dan berisi jumlah oktet yang tidak terpisahkan. Sebagaimana ditentukan untuk memastikan bahwa urutan bit *flag* yang disebutkan tidak muncul secara tidak sengaja di tempat lain dalam sebuah *frame*, stasiun pengirim memonitor *bit sequence* atau urutan bit "1" sebanyak lima atau lebih. Kapan saja lima bit "1" yang berdekatan dikirimkan, stasiun pengirim memasukkan bit "0" setelah bit kelima "1". Selama penerimaan *frame*, setiap saat lima bit "1" yang berdekatan diterima, bit "0" setelah lima bit "1" dibuang proses ini disebut *bit stuffing*.

6. Frame Check Sequence

Frame-Check Sequence (FCS) adalah *field* berisi enam belas-bit yang dihitung oleh pengirim dan penerima bingkai. *Frame-Check Sequence* memastikan bahwa *frame* tidak rusak oleh media transmisi. *Frame-Check Sequence* dihitung sesuai dengan rekomendasi dalam dokumen referensi HDLC ISO 3309[7].

BAB 3
PERANCANGAN SISTEM

Terdapat beberapa tahapan yang dilalui dalam merancang protokol *data-Link* hingga akhirnya diimplementasikan dalam system komunikasi radio HF antara lain:

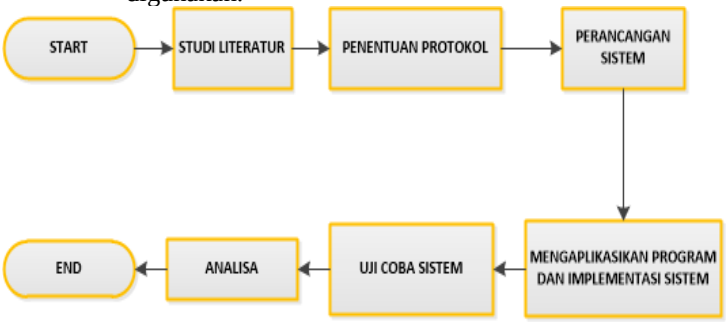
1. Studi Literatur
   Studi literatur dilakukan sebelum memulai proses perancangan untuk mengetahui hal – hal dasar yang diperlukan untuk perancangan.
2. Penentuan protokol
   Penentuan protokol didukung dengan beberapa kriteria sebagai penunjang sistem meliputi informasi yang diolah yaitu berupa teks / kumpulan karakter untuk mendukung proses pada saat transmisi dan mengurangi terjadinya *error (error detection)*. Protokol yang diperlukan untuk sistem komunikasi radio digital yaitu menggunakan protokol AX.25 dikarenakan sesuai dengan kriteria perancangan protokol.
3. Perancangan Sistem.
   Perancangan yang dilakukan disini merupakan perancangan sistem perangkat keras dan perancangan *software*. Perancangan program AX.25 menggunakan Labview dan AX.25 yang dibuat akan dirancang sesuai dengan sistem dengan informasi berupa karakter yang diolah dan sesuai dengan format suatu *Frame* AX.25. AX.25 dalam sistem bekerja pada sisi *encoder* di pengirim dan *decoder* pada penerima.
4. Mengaplikasikan program dan implementasi ke sistem komunikasi.
   Pada bagian ini dilakukan implementasi program protokol AX.25 ke sistem komunikasi yang direncanakan pada *software* Labview dan didukung oleh perangkat keras yang mendukung sistem.
5. Uji coba sistem dan Analisa
   Uji coba sistem dilakukan untuk mendapatkan analisa apakah protokol data-*link* yang di rancang dapat berjalan dengan baik atau tidak. Skenario uji coba yang dilakukan adalah :
   a. Uji coba simulasi yang didalamnya terdapat program AX.25 didukung dengan penggunaan Labview dan laptop. Uji coba ini dilakukan dengan memberikan jumlah karakter yang berbeda-beda untuk melihat kinerja program dengan parameter

waktu proses yang terjadi. Jumlah karakter yang diujikan adalah 10, 200,500, 1000 dan 2000 karakter dan masing masing karakter diulang sebanyak 5 kali pengiriman dengan dihitung durasi waktu prosesnya. Jumlah karakter mengacu pada ketentuan AX.25 *frame* untuk *field* informasi maksimal data 256 *byte*/karakter.

b. Uji coba protokol AX.25 dengan sistem komunikasi dan perangkat USRP dilakukan dengan 2 tahapan yaitu :

   i. Pengujian jarak dekat antara antenna gedung AJ dan B dilakukan dari siang hingga sore hari. Uji coba ini dilakukan dengan memberi variasi jumlah karakter seperti 10, 200,500, 1000 dan 2000 karakter dan masing masing karakter diulang sebanyak 5 kali pengiriman dengan dihitung durasi waktu prosesnya. Jumlah karakter mengacu pada ketentuan AX.25 *frame* untuk *field* informasi maksimal data 256 *byte*/karakter. Tidak hanya banyak karakter parameter konfigurasi dengan USRP juga digunakan.

   ii. Pengujian jarak jauh antara antenna di gedung Teknik Elektro dengan antenna gedung VEDC Malang. Uji coba ini dilakukan dengan memberi variasi jumlah karakter seperti 10, 200,500, 1000 dan 2000 karakter dan masing masing karakter diulang sebanyak 5 kali pengiriman dengan dihitung durasi waktu prosesnya. Jumlah karakter mengacu pada ketentuan AX.25 *frame* untuk *field* informasi maksimal data 256 *byte*/karakter. Tidak hanya banyak karakter parameter konfigurasi dengan USRP juga digunakan.



Gambar 3.1. Diagram Alir Tahapan Perancangan Sistem

3.1 Penentuan Protokol

Sebelum menentukan protokol yang sesuai dengan sistem komunikasi radio digital, kriteria sistem yang diperlukan adalah:

1. Komunikasi dengan frekuensi tinggi (*High Frequency*).
2. *Tranceiver* berupa informasi / pesan teks.
3. Komunikasi *Link to Link (single Link).*
4. Menggunakan *Software Defined Radio* sebagai perangkat utama, terutama USRP dan didukung *software* labview.
5. Penggunaan NVIS dikarenakan *urgensi* apabila suatu tempat terjadi bencana dan menyebabkan kelumpuhan telekomunikasi, daerah terpencil.

Kriteria Protokol yang sesuai dengan sistem komunikasi ini adalah:

1. Mendukung HF *Gateways*
2. Mendukung dalam komunikasi radio amatir
3. Transfer data antar *node*, dan dapat mendeteksi eror *(error detection)*
4. Terdapat mekanisme *error control* ARQ
5. *Overhead* yang rendah untuk mematuhi *bitrate* dan *bandwidth* rendah.
6. Dapat diimplementasikan di sistem komunikasi menggunakan *software* Labview[9].

Maka sesuai dengan kriteria yang diperlukan oleh sistem komunikasi ini, penulis menggunakan AX.25 protokol sebagai protokol pengiriman data untuk komunikasi radio amatir. Alasan lain adalah terdapat 3 *error control* ARQ yaitu *Stop-and-Wait*, *Go-Back-N*, dan *Selective Reject* ARQ dilihat dalam proses AX.25 dan perbandingan 3 ARQ maka AX.25 ini mengacu pada mekanisme *Selective Reject* ARQ dikarenakan adanya buffer pada penerima dan *Selective Reject* ARQ lebih efisien dari pada *Go-Back-N* karena dapat meminimalkan jumlah pengiriman ulang dengan hanya mengirim *frame* yang terjadi kesalahan saat diterima pada penerima, hal ini sesuai dengan studi yang dilakukan sebelum dipilih protokol yang diinginkan.

3.2 Perancangan Sistem

Setelah melakukan identifikasi awal dan pemilihan protokol maka selanjutnya adalah melakukan perancangan sistem untuk memodelkan sistem. Salah satu kelebihan melakukan pemodelan ini adalah dapat

mendeskripsikan rancangan proses sehingga dapat mempermudah pembuatan program yang salah satunya adalah program AX.25 dalam sistem komunikasi digital.

3.2.1 Konsep Sistem Komunikasi Data

Konsep sistem komunikasi data pada tugas akhir ini adalah pemanfaatan dari 2 lapisan paling bawah dalam OSI *Layer* yaitu *layer* 1 (pertama) yaitu lapisan fisik / physical *layer* dan *layer* 2 (kedua) yaitu lapisan *datalink*.

1) Physical Layer

Lapisan ini bertanggung jawab atas proses data menjadi bit dan mentransfernya melalui media.

2) DataLink Layer

Layer ini menyediakan *Link* untuk data dan memaketkannya menjadi *Frame*. Pada *layer* ini juga melakukan sinkronisasi, *error detection* dan *flow Control*.

Gambar 3.2 Konsep Sistem Komunikasi Data Antar Lapisan OSI

Transmitter adalah sisi dimana terjadi input data berupa informasi dalam karakter / *byte* dengan *keyboard* PC. Lalu proses terjadi dalam lapisan 2 yaitu data-*link* :

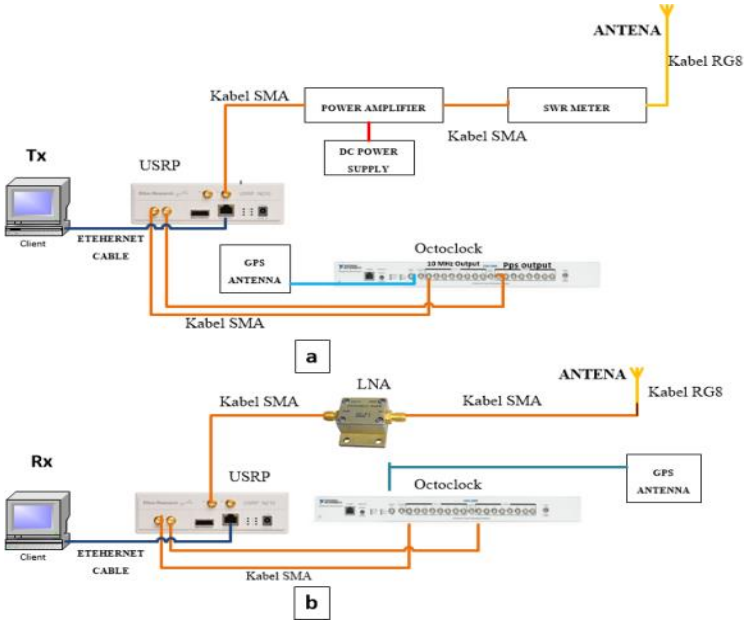
a. Karakter (byte) diubah ke dalam bit.
b. Bit informasi di susun dalam suatu *frame* AX.25 dimana bit informasi tidak berdiri sendiri melainkan terdapat flag sebagai sinkronisasi *frame* ,header *frame* dan error detection.

Setelah terjadi proses didalam data-*link layer*, selanjutnya *frame* yang berisikan barisan bit di proses oleh *physical layer* untuk di transmisikan.

Proses dalam *physical layer* yaitu terjadi modulasi serta pemancar dengan mempertimbangkan *link budget* pengirim. Spesifikasi tentang bentuk sinyal yang akan ditransmisikan, media pengantar (kabel), media penghubung (*connector*) dan lainnya juga diatur oleh *physical layer*.

Sinyal ini akan diantar ke tujuan dan diterima oleh penerima / *receiver*. Setelah itu, masuk ke *physical layer* dimana terjadi sinkronisasi gelombang pembawa dengan menggunakan *phase lock loop* dan demodulasi untuk mengubah sinyal menjadi bit, kemudian pada data-*link layer* terjadi proses bit menjadi *byte* kembali.

3.2.2 Perancangan Sistem Perangkat Keras



Gambar 3.3 Perancangan Sistem Perangkat Keras pada (a) Pengirim dan (b) Penerima

Sistem yang dibangun membutuhkan beberapa unsur komponen yang harus dilengkapi, meliputi perangkat keras dan perangkat lunak. Perangkat keras yang dibutuhkan adalah sebagai berikut:

Tabel 3.1. Daftar Perangkat Keras

No	Perangkat	Jumlah
1	*Antenna* Horizontal Dipole	2 buah
2	USRP dengan daughter board LFRX dan LFTX dengan spesifikasi frekuensi 0 – 30MHz.	2 buah
3	Octoclock Ettus Research	2 buah
4	*Antenna* GPS	2 buah
5	Laptop	2 buah
6	Low *Noise* Amplifier	1 buah
7	High *Power amplifier*	1 buah
8	DC *Power Supply*	1 buah

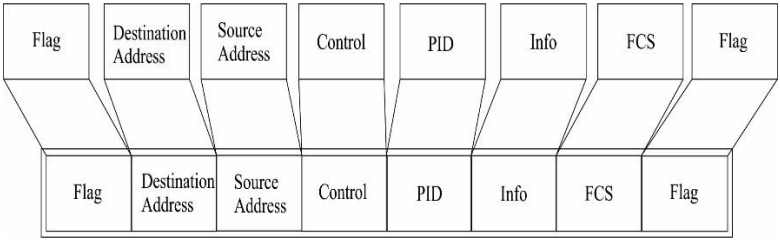
Selain perangkat keras, penggunaan perangkat lunak yang digunakan adalah Labview tahun 2014. Untuk rancangan sistem perangkat keras terdapat 2 sisi, yaitu sisi pengirim atau *transmitter* dan sisi penerima atau *receiver*. Pada sisi pengirim PC / Laptop terhubung dengan perangkat USRP menggunakan media kabel gigabit Ethernet. Setelah itu dengan menggunakan kabel SMA sebagai transmisi menghubungkan *port* REF IN USRP terhubung ke *port* satu 10 MHz OUT *channel* clock *distribution module* yang ada di octoclock dan *Port* PPS IN USRP terhubung ke *port* satu PPS OUT *Channel* clock *distribution module*. Octoclock disini berfungsi sebagai perangkat yang menyesuaikan waktu area sekitar dengan memanfaatkan GPS *Antenna*. Lalu sisi antena di hubungkan pada *port* RF1 sebagai Tx1 menggunakan kabel SMA yang terhubung ke *power amplifier*. *Power amplifier* disini berfungsi sebagai penguat daya yang keluar dari USRP agar informasi dapat ditransmisikan dan diterima di penerima. *Power amplifier* disini terhubung pada DC *Power Supply* sebagai pemberi daya dengan tegangan masukan 12 volt. Selanjutnya *power amplifier* dihubungkan ke SWR Meter menggunakan kabel RG8 gunanya agar dapat memantau terus daya yang dipancarkan. Lalu dari SWR Meter disambungkan dengan antena Dipol sebagai pemancar.

Pada sisi penerima / *receiver* PC / Laptop terhubung dengan perangkat USRP menggunakan media kabel gigabit Ethernet. Setelah itu dengan menggunakan kabel SMA sebagai transmisi menghubungkan *port* REF IN USRP terhubung ke *port* satu 10 MHz OUT *channel* clock *distribution module* yang ada di octoclock dan *Port* PPS IN USRP terhubung ke *port* satu PPS OUT *channel* clock *distribution module*. Octoclock disini berfungsi sebagai perangkat yang menyesuaikan waktu

area sekitar dengan memanfaatkan GPS *Antenna*. Lalu sisi antena di hubungkan pada *port* RF1 sebagai Rx1 menggunakan kabel SMA yang terhubung ke LNA (*Low Noise Amplifier*) pada sisi *output* dan pada sisi *input* LNA dihubungkan dengan antena Dipol sebagai penerima.

3.2.3 Perancangan Simulasi Program AX.25 pada Labview

Perancangan simulasi AX.25 sendiri terdiri dari program yang mampu menyusun informasi ke dalam bentuk *Frame* AX.25 seperti pada gambar 3.4. tentang gambaran dari masing masing *field* akan di jadikan satu dalam satu *frame* AX.25. Lalu terdapat generator *noise* yang disimulasikan.



Gambar 3.4 Rancangan Metode *Frame* AX.25

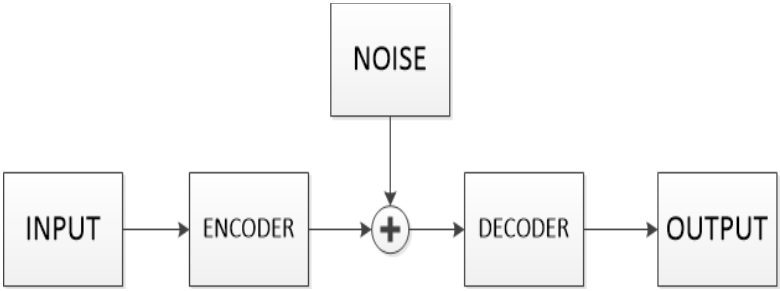
Dalam masing-masing *field* diisi dengan data berupa bilangan biner dengan susunan sesuai dengan rancangan metode *frame* AX.25. Isi per-*field* menyesuaikan dengan tabel 3.2.

Tabel 3.2 Perancangan Isi sub*field*

Octet	ASCII	Bin Data	Hex Data
Flag		01111110	7E
A1	Y	01011001	59
A2	G	01000111	47
A3	3	00110011	03
A4	E	01000101	45
A5	G	01000111	47
A6	Y	01011001	59
A7	SSID	11100000	E0
A8	A	01000001	41
A9	0	00110000	30
A10	0	00110000	30
A11	0	00110000	30

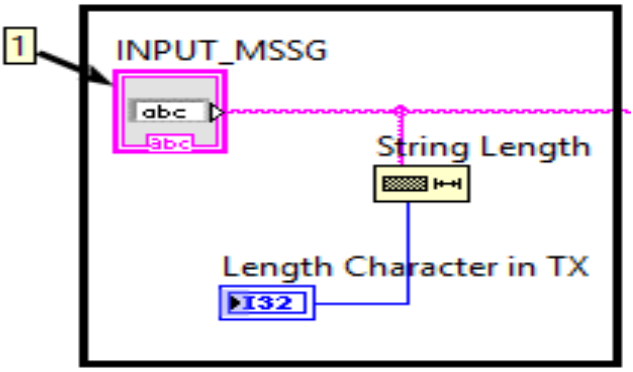
Octet	ASCII	Bin Data	Hex Data
A12	0	00110000	30
A13	2	00110010	32
A14	SSID	01100001	61
Control	1	00111110	3E
PID	None	11110000	F0
FCS	Part1	10011000100010101	13115
FCS	Part2	10011000100010101	13115
Flag	Part2	01111110	7E

Gambar 3.5 adalah diagram blok dari desain simulasi pada Labview dengan *input* berupa teks lalu diolah oleh *encoder*. Hasilnya akan ditambah dengan *noise* dari pembangkit *noise* setelah itu diteruskan ke *decoder* dan diubah menjadi teks kembali.



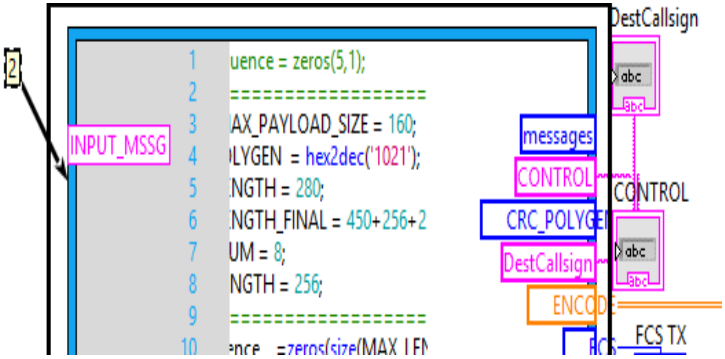
Gambar 3.5 Diagram Blok Simulasi

Proses pertama yang dilakukan adalah memberikan *input* berupa teks / karakter dimana format pada labview menggunakan indikator *String* seperti yang terlihat pada Gambar 3.6 Tidak hanya demikian, ditambahkan fungsi "*String Length*" untuk menghitung panjang jumlah karakter (*byte*) dalam *string*.



Gambar 3.6 Blok Diagram *Input* Teks pada Labview

Lalu langkah kedua adalah proses *encoder* dimana masukan teks tadi di olah menjadi barisan bit data, bukan berupa *byte* / karakter lagi. Barisan bit ini di representasikan dalam *array*.



Gambar 3.7 Tampilan *Encoder*

Proses *encoder* seperti Gambar 3.7 menggunakan LabVIEW *Mathscript*s dan skrip berbasis teks menggunakan mesin *Mathscript* RT Module. *Mathscript Node* digunakan untuk mengevaluasi skrip yang di buat di Jendela LabVIEW *Mathscript*. Didalam *Mathscript node* yang dibuat terdapat *script* main program dari proses *encoder* / bit processing. Memasukkan program disini menggunakan fungsi im*port* agar terintegrasi dengan fungsi lain diluar *main program*e, hal ini

berhubungan dengan kinerja dari LabView agar tidak terlalu berat dan lama operasinya. Program keseluruhan dari penyusunan *field* untuk menjadi suatu *frame* antara lain:

1. Menambahkan *Header*
   Menggeser 1bit ke kiri untuk memungkinkan bit ekstensi HDLC.
   a. Membuat *FLAG*
      FLAG disini berfungsi sebagai sinkronisasi antar *Frame* dimana *FLAG* dengan menyisipkan angka 1 sebanyak 6 kali diikuti dengan angka 0 diakhir atau dalam hexa direpresentasikan dengan 0×7E dalam biner “01111110”.

      Perintah dalam program :

```
%create 0b01111110 (FLAG) in byte size
for i=1:1:64
    Buffer(size_)= 0;%hex2dec('00');
    size_ = size_ + 1;
    for j = 1:1:6
        Buffer(size_)=1;%hex2dec('01');
        size_ = size_ + 1;
    end
    Buffer(size_)=0;%hex2dec('00');
    size_ = size_ + 1;
end
max= s-1;
%for (int i=0; i < s ; i++) Buffer[_size++] =
BitSequenceStuffed[i];
 for i = 1:1:max;
     Buffer(size_) = BitSequenceStuffed(i);
     size_ = size_+1;
 end
```

   b. Menambah destination *callsign* (*callsign* tujuan) dimana didalamnya terdapat ssid tujuan. Berikut program matlab sebagai perintah dalam menetapkan *callsign* tujuan.

      Perintah dalam program :

```
STR_TO_CALLSIGN ='A00002';
c =1;
Index =1;
for i=1:1:length(DestCallsign)
```

```
    %Append SSID Destination
    bitsequence(Index)                          =
bitshift(unicode2native(Dest Callsign(i)),1);
    c = c+1;
    Index = Index+1;
end
bitsequence(Index)                              =
hex2dec('60');%ssid_destination;
Index = Index+1;
```

Dapat dilihat bahwa *callsign* tujuan yang ditetapkan adalah “A000002”.

c. Menambah source *callsign* di indeks setelah destination *callsign* dimana didalamnya juga terdapat ssid sumber.

Perintah dalam program :

```
STR_FROM_CALLSIGN ='YG3EGY';
%Append Source Callsign
c =1;
for i=1:1:length(Src Callsign)
    bitsequence(Index)                          =
bitshift(uint16(Src Callsign(i)),1);
    %fprintf(num2str(bitsequence(Index)));
    %fprintf(' ');
    Index = Index+1;
    c =c+1;
end
bitsequence(Index)                              =
hex2dec('61');%ssid_source;
Index = Index+1;
```

Dapat dilihat bahwa *callsign* sumber yang ditetapkan adalah “YG3EGY”.

d. Menambahkan *Control* bits setelah *source callsign* dengan nilai dari *control* bits adalah 0×03 yang artinya menggunakan tipe *Unnumbered Information Frames* yaitu,

Perintah dalam program :

```
CONTROL = '03';%'0x03';
%Append Control bits
```

```
bitsequence(Index)  = hex2dec(CONTROL);
Index = Index+1;
```

e. Menambahkan Protokol Identifier (PID) dengan nilai 0×F0 ynag berarti tidak menggunakan protokol *layer* 3 dan ditambahkan setelah *Control* bits dengan perintah.

Perintah dalam program :

```
PID ='F0';%'0xF0';
%Append Protokol Identifier
bitsequence(Index)  = hex2dec(PID);
Index = Index+1;
```

2. Menambahkan Pesan / Informasi
Menentukan informasi yang ingin disisipkan ke dalam *Field* informasi. Didalamnya terdapat *convert bitsequence* dimana merupakan proses dari MSB menjadi LSB, dengan CRC .

Perintah dalam program :

```
%Add Message
c =1;
for i=1:1: length(txtToSending)
      bitsequence(Index) = txtToSending(1,c);
      Index = Index+1;
      c = c+1;
end
for i=1:1:Index-1
  v = (bitsequence(i));
  bitsequence(i) = MSB_LSB_swap_8bit(v);
end
```

3. Menambahkan FCS (*Frame Check Sequence*). FCS ditambahkan pada MSB (*Most Significant Bit*)

Perintah dalam program :

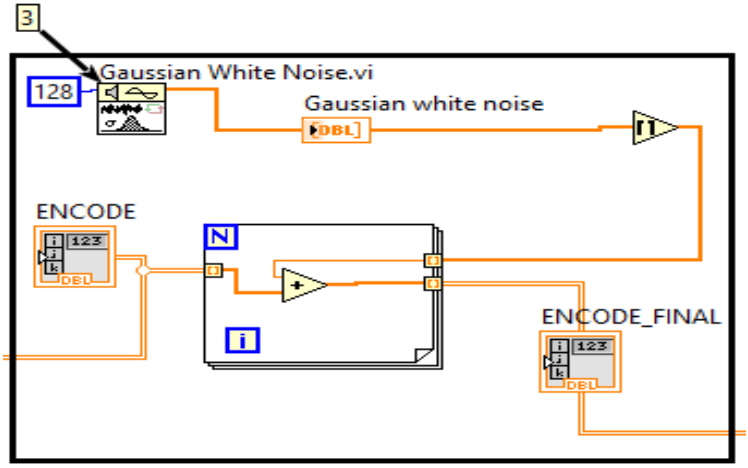
```
FCS=typecast(int16(13115),'uint16');%bin2dec('1100110
0111011');%13115;%CRC_CCITT(bitsequence,Index-1);
fprintf('FCS:');
fprintf(num2str(FCS));
fprintf('\n');
```

4. Hasil Akhir
Hasil dari proses *encoder* adalah bit sequence / barisan bit akhir dan pada saat ini terdapat pembuatan *Flag* dengan biner

01111110 lalu di *recreate* / buat ulang dalam ukuran *byte* . Setelah dibuat maka *flag* di sisipkan.

Setelah proses penyusunan menjadi satu *frame* , jadilah barisan bit akhir atau "*final sequence*".

Langkah ketiga terlihat pada Gambar 3.8 adalah penambahan *noise* generator agar seolah olah terjadi adanya *noise* pada saat proses transmisi dari *encoder* ke *decoder*, hal ini dirancang sesuai dengan sebagaimana sistem komunikasi secara nyata terjadi yaitu adanya *noise* apabila melalui media berupa udara. Hasil dari proses *encoder* adalah berupa bit *array* yang ditambah dengan nilai dari *noise* generator yang didistribusikan. Generator *noise* disini menggunakan Gaussian *white noise* dengan 128 *sample* menyesuaikan *default* dari mode Labview sendiri. Setelah ditambahkan maka dapat menghasilkan *array encode final.*



Gambar 3.8 Blok Diagram *Noise*

Langkah keempat adalah proses *decoder* dimana *input* yang masuk merupakan barisan bit (*bitsequence*) hasil dari *encoder* yang telah ditambahkan dengan *noise* seperti Gambar 3.8. Lalu proses berjalan pada *Mathscript node* yang dibuat dan didalamnya terdapat *script* main program dari proses *decoder*. Terdapat beberapa tahapan dalam proses

decoder sehingga barisan bit menjadi *string*, secara umum yaitu proses ini kebalikan dari proses *encoder*:

1. Menemukan dan Menghilangkan *Flag*.
   Proses ini merupakan awal pada saat *decoder* menerima barisan bit akhir dari pengirim lalu terjadi sinkronisasi antara *frame* satu dengan *frame* yang lain.

   Perintah dalam program :

```
for i =1:1:k-1
   if  ByteSequence(i) ~= bin2dec('01111110')
       pastFlag = true;
       ByteSequence_temp(cnt) = ByteSequence(i);
       cnt = cnt +1;
   elseif pastFlag == true
       break;
   end
end
```

2. FCS (*Frame Check Sequence*)
   Melakukan kalkulasi CRC untuk menentukan FCS apakah sesuai dengan FCS dipengirim atau tidak.
   Perintah dalam program :

```
FCS =
13115;%typecast(int16(13115),'uint16');%CRC_CCITT(B
yteSequence, k-2);
Checksum(1) = ByteSequence(k-2);
Checksum(2) = ByteSequence(k-1);

if Checksum(1) ~= bitand(bitshift(FCS,-
8),hex2dec('FF'))
   fprintf('Error in Checksum 1 : ');
   fprintf('\n');
end

ERROR = 'tidak';
if Checksum(2) ~= (bitand(FCS,hex2dec('FF')))
   fprintf('Error in Checksum 2 : ');

 ERROR='ya';
   fprintf('\n');
else
   fprintf('Correct checksum 2\n ');
end
```

3. LSB to MSB

Perintah dalam program :

```
for i=1: i : byte*length
     Byte*Sequence_temp(i) = 0;
 end
 %for (int i=0; i < k-2 ; i++) Byte*Sequence_temp[i]
= MSB_LSB_swap_8bit(Byte*Sequence[i]);
 for i=1: 1 : k-3
     Byte*Sequence_temp(i)                          =
MSB_LSB_swap_8bit(Byte*Sequence(i));
     fprintf(num2str(Byte*Sequence_temp(i)));
     fprintf('   ');
 end
```

4. *Recover Header*

Setelah merubah LSB ke MSB maka dilanjutkan ke pemulihan komponen dari header.

Perintah dalam program :

```
for i =1:1:6
    DestCS(i)=  bitshift(Byte*Sequence_temp(cnt),-1);
cnt =cnt+1;
 end
%Append Source Callsign
 %for  (int  i=0;  i  <  6;  i++)  SourceCS[i]  =
char(Byte*Sequence_temp[cnt++]>>1);
  for i=1:1:6
    SourceCS(i)=    bitshift(Byte*Sequence_temp(cnt),-
1);
    cnt =cnt+1;
  end
  %Append SSID Source
  cnt =cnt +1;
  %Append Control bits
  cnt =cnt +1;
  %Append Protocol Identifier
  cnt =cnt +1;
  %Recover message
  s = k-2-cnt;
  fprintf('Final decoded Message\n');
```

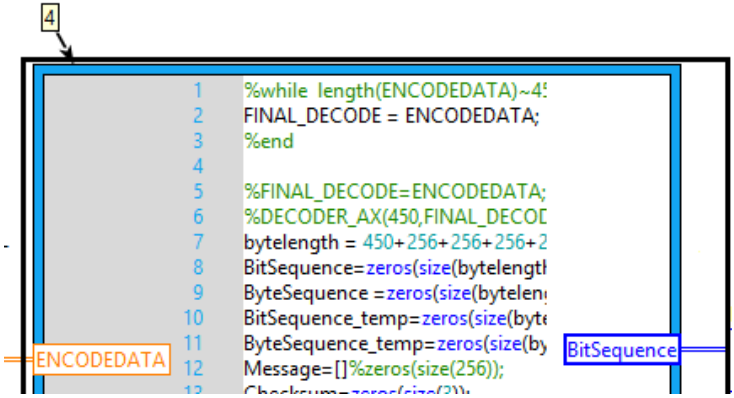
5. Menambahkan *SSID* Sumber

Menambahkan SSID sumber ini diambil dari *callsign* sumber yang diberikan pada saat proses pada transmitter.

6. Menambahkan *Control Bits*
   Menambahkan bit control disini disesuaikan nilainya seperti apa yang sudah di sesuaikan di pemancar / transmitter.

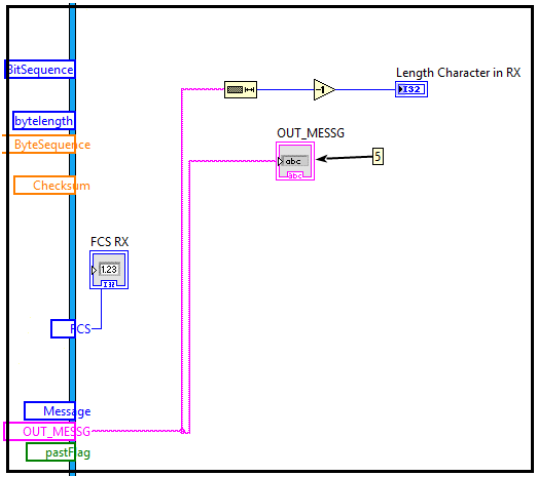
7. Menambahkan *Protokol Identifier*
   Menambahkan PID disini disesuaikan nilainya seperti apa yang sudah di sesuaikan di pemancar / transmitter.

8. *Recover Message*



Gambar 3.9 *Mathscript Node* pada Labview

Langkah kelima merupakan *output* hasil akhir dari semua proses, dimana *Indicator* OUT_MSSG bernilai *string* dan muncul pada panel / *user interface* yang berisi karakter yang asli atau yang dikirim.



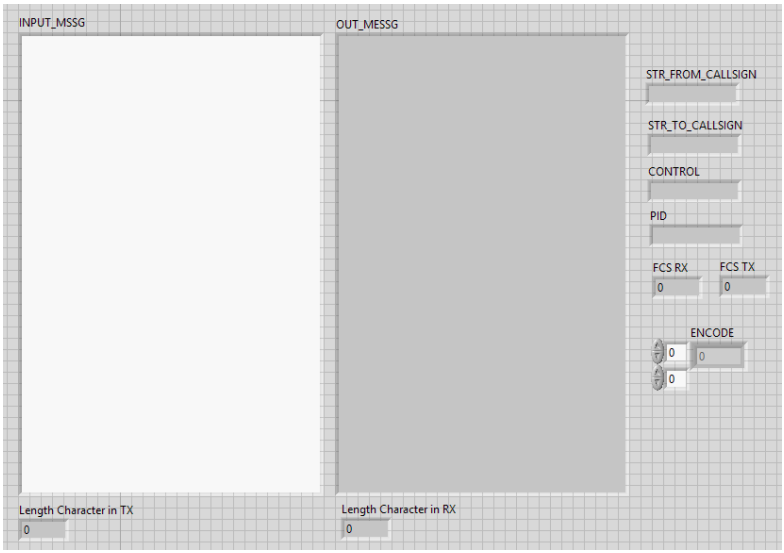
Gambar 3.10 Blok Diagram *Output*

3.2.4 Perancangan Pengujian Sistem AX.25 pada Labview

Pengujian sistem yang dibuat melalui pengukuran bertujuan untuk evaluasi sistem sederhana AX.25 yang telah dibuat. Desain sistem yang digunakan untuk pengujian adalah program yang berisi sub sistem yang menjadi *encoder* pada pengirim dan *decoder* disisi penerima. Program sistem akan disertakan pada halaman lampiran.

Hal yang perlu diperhatikan dalam pengukuran adalah data apa saja yang ingin didapatkan. Data tersebut ditampilkan pada bagian panel karena LabVIEW hanya mengambil (*capture data*) data-data yang ditampilkan pada panel. Untuk menyesuaikan kemampuan perangkat dan laptop dalam menjalankan program agar berjalan maka data yang diambil adalah urutan *Field Frame* AX.25 yang diinginkan seperti:

1. *Indicator String From Callsign* (*String*)
2. *Indicator String To Callsign* (*String*)
3. *Indicator Control* (*string*)
4. *Indicator* PID
5. *Indicator* FCS
6. Indikator informasi yang diproses

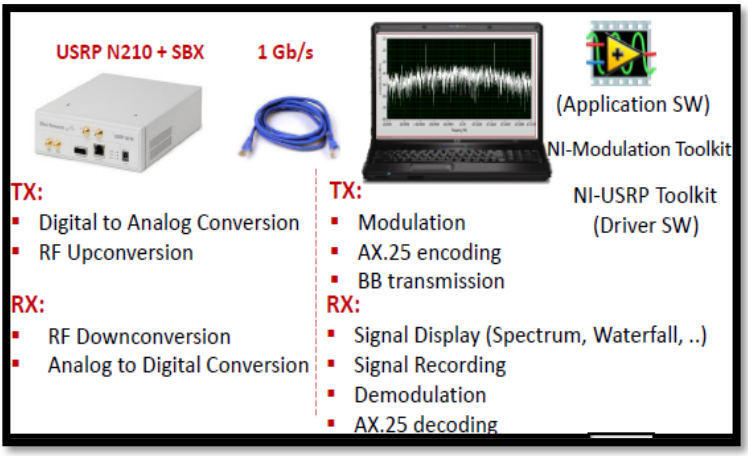


Gambar 3.11 Tampilan Panel Keseluruhan

Perancangan AX 25 akan diuji dengan beberapa macam jumlah karakter yang akan dikirimkan yaitu 10, 200, 500, 1000 dan 2000 karakter dengan mencatat waktu kecepatan proses dikirim hingga diterima sebanyak 5 kali pengambilan data.

3.3 Pengaplikasian Program AX.25 ke dalam Sistem Komunikasi Digital dengan Labview

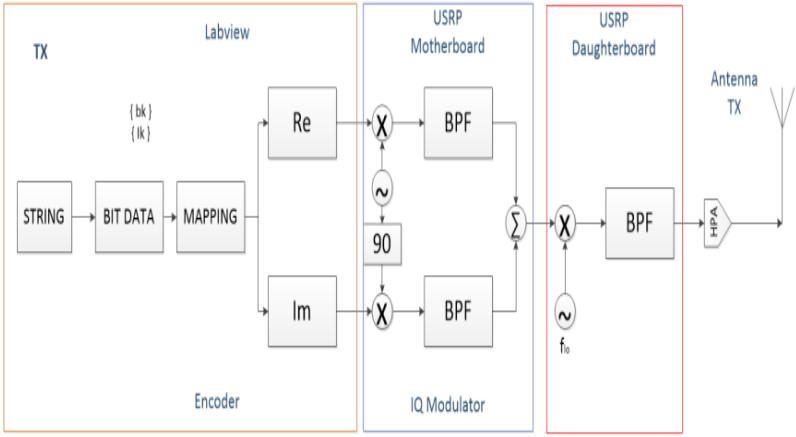
Perancangan untuk pengaplikasian AX.25 ke sistem komunikasi dapat dilihat dari Gambar 3.12, dimana antara perangkat USRP N210 *daughterboard* LFTX/LFRX dengan Laptop yang terinstall dengan software LabView dikoneksikan dengan media Ethernet 1Gb/s. Setelah mengetahui hubungan antara Labview dengan perangkat USRP, maka dirancanglah *encoder*, modulasi dan sebagai pelengkap disisi *hardware* yaitu HPA (*High Power Amplifier*) dan antenna dipol pada sisi pemancar seperti gambar 3.13 dan *decoder*, demodulasi serta menggunakan pelengkap dalam sisi *hardware* yaitu LNA (*Low Noise Amplifier*) seperti yang ditunjukan pada gambar 3.14.



Gambar 3.12 Perancangan hubungan SDR dan *platform* [9]

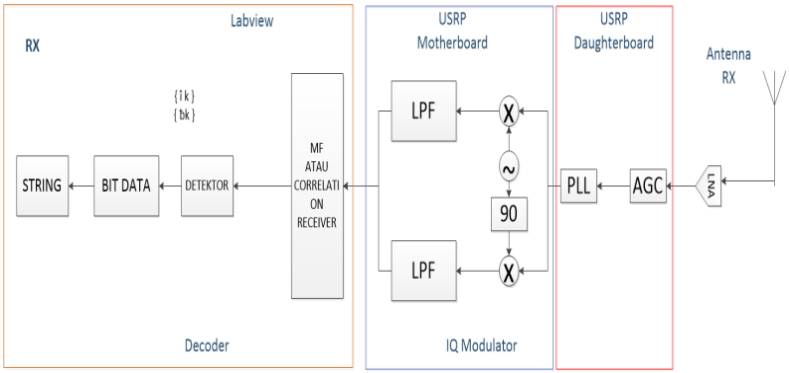
Proses yang terjadi pada sistem komunikasi sisi pemancar adalah dimana deretan bit data yang dikirimkan untuk proses transmisi data pada sistem komunikasi sebelumnya akan dilakukan mapping terlebih dahulu. Arti dari mapping disini adalah untuk mengubah deretan bit menjadi sinyal LPE (*Low Past Equivalent*) yang nantinya dapat di kelompokan ke dalam dua sinyal yaitu Re (*real*) dan Im(*Imaginner*). Proses tersebut terjadi pada *software* labview yang sudah terinstall di laptop. Setelah dari proses pemisahan sinyal tersebut, maka data akan di jadikan satu ke dalam Iq Modulator dimana untuk perangkat terdapat di perangkat *Motherboard* USRP. Dengan memiliki sistem kerja berupa dua sinyal yang masuk di kuatkan dan disesuaikan dengan kondisi kanal menggunakan oscillator dengan mengikuti pergerakan fasa hingga 90 derajat. Setelah dua sinyal itu masuk maka akan di rubah kembali menjadi sinyal BPF (*Band Pass Filter*). Kemudian kedua sinyal yang *Real* dan *Imaginer* di jumlahkan terlebih dahulu untuk dapat melihat kualitas sinyalnya. Setelah dapat melihat kualitas sinyalnya dan dirasa kurang, maka dapat di kuatkan kembali dengan oscilator sebagai penguat berdasarkan frekuensinya masing-masing. Sebelum dilakukan transmisi data pada sistem komunikasi maka sinyal tersebut kembali di rubah ke dalam Sinyal BPF (*Band Pass Filter*) dan dikuatkan kembali oleh HPA (*High Power Amplifier*) untuk dapat menembus kondisi di

ionesfer. Setelah itu maka transmisi pengiriman data dapat dilakukan melalui antena pada sisi pemancar.



Gambar 3.13 Perancangan Sistem Komunikasi pada Pemancar

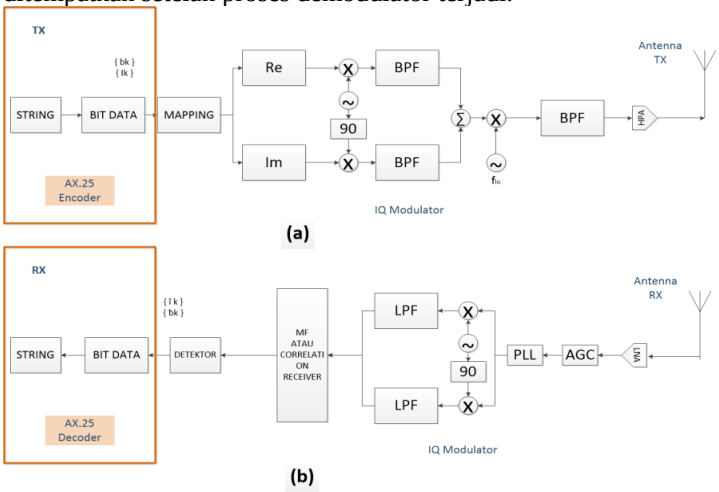
Proses yang terjadi pada sistem komunikasi sisi penerima / *receive* adalah antenna menerima sinyal dari kanal. Sinyal yang diterima tersebut adalah sinyal dalam kondisi yang kurang baik yang artinya, kondisi dimana sinyal telah melalui transmisi berupa media udara yang banyak terjadi gangguan / *noise*, maka perlu di kuatkan kembali dengan perangkat keras bernama LNA (*Low Noise Amplifier*) yang berguna untuk menguatkan kualitas sinyal dan dapat membuat data yang dikirimkan pada proses pengiriman dari pemancar dapat diterima dipenerima. Untuk meningkatkan sinyal maka diperlukan AGC (*Automatic Gain Control*) yang memiliki fungsi untuk menguatkan sinyal yang sangat lemah agar *gain* antenna yang diterima dapat berfungsi maksimal dengan kualitas sinyal yang konstan. Selain itu AGC dapat mengkompensasi suatu redaman yang biasanya terjadi pada perangkat USRP *daughterboard*. Kemudian pada IQ modulator, sinyal yang diterima pada penerima di kelompokan menjadi dua yaitu *Real* dan *Imaginer* dengan di kuatkan kembali dengan gelombang pembawa. Setelah dari proses tersebut, sinyal akan dirubah menjadi LPF (Low Pass Filter), dimana prosesnya terjadi pada *motherboard*.



Gambar 3.14 Perancangan Sistem Komunikasi pada Penerima

3.3.1 Program AX.25 dengan Keseluruhan Sistem

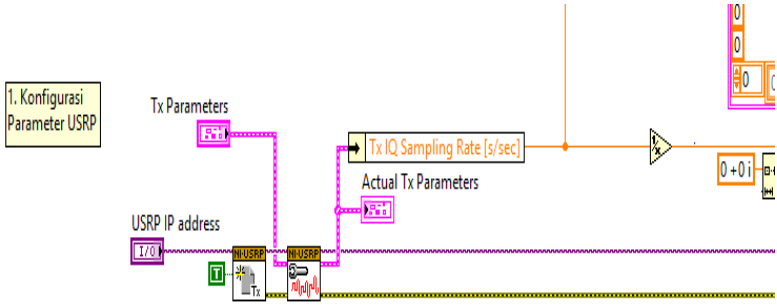
Pada langkah ini program AX.25 di integrasikan pada keseluruhan sistem yaitu sistem pemancar dan penerima. Dalam gambar 3.15 terlihat bahwa AX.25 *Encoder* berada di pengirim dan ditempatkan sebelum proses modulator dimana bit data yang dihasilkan oleh *encoder* menjadi masukan pada proses modulasi. AX.25 *Decoder* berada di penerima dan ditempatkan setelah proses demodulator terjadi.



Gambar 3.15 Perancangan (a) AX.25 *Encoder* dan (b) *Decoder* pada Sistem Komunikasi

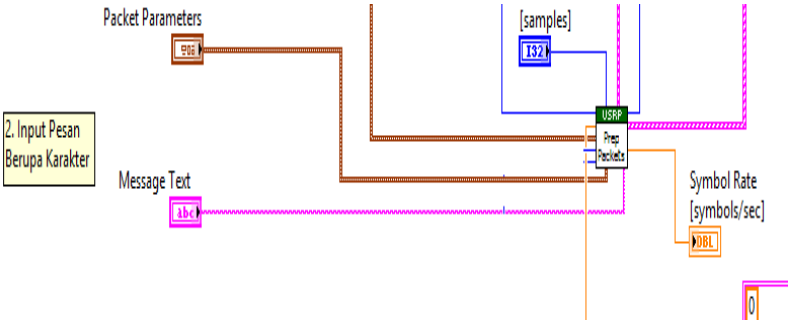
3.3.1.1 Sub Sistem Pemancar Pada LabView 2014

Subsistem pemancar ini berisi program-program sistem mulai dari konfigurasi parameter dalam Labview hingga terkonfigurasi dengan USRP, Subsistem pengolahan karakter / pesan masukan dan subsistem modulasi.



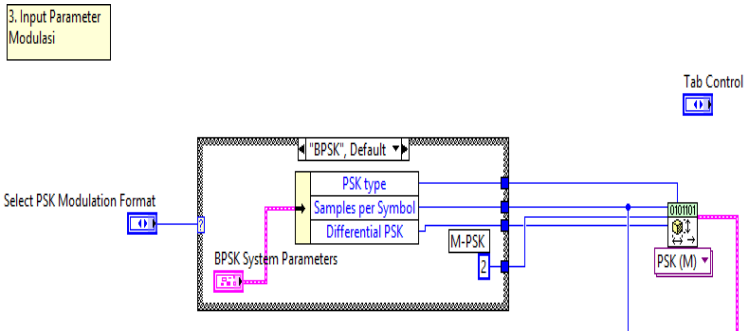
Gambar 3.16 Perancangan sub sistem langkah 1 dalam pemancar

Perancangan sub-sistem yang pertama ini adalah berupa sub-sistem dimana terjadi input parameter awal seperti USRP IP *Address*, TX IQ *Rate* [s/sec], *Tx Antenna (Active Antenna)* dan TX Frequency dalam Hz. USRP Ip address menjadi input pertama sebagai pembuka sesi pengiriman (Tx) ke perangkat yang ditentukan dalam nama perangkat dan menangani kembali sesi instrument untuk VI NI-USRP berikutnya.



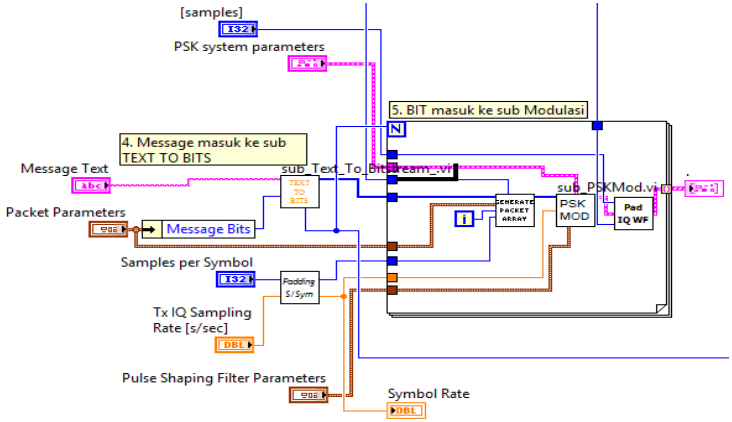
Gambar 3.16 Perancangan sub sistem langkah 2 dalam pemancar

Pada langkah ini adalah langkah dimana pemberian teks / karakter pada panel Labview sebagai data yang akan diolah dan dimasukkan sebagai *Information field* . setelah diketikkan karakter yang diinginkan, lalu masuk ke sub vi *Generate Packet* untuk diolah.



Gambar 3.17 Perancangan sub sistem langkah 3 dalam pemancar

Pada langkah 3 memasukkan parameter modulasi yang digunakan seperti modulasi BPSK yang digunakan pada tugas akhir ini dengan alasan membuat komunikasi data yang baik dan BPSK memetakan satu bit ke satu symbol lalu penentuan nilai *samples per symbol* yang ditentukan. Penggunaan modulasi M-PSK bertujuan untuk memetakan bit data / *bit streaming* ke dalam simbol-simbol M-PSK.

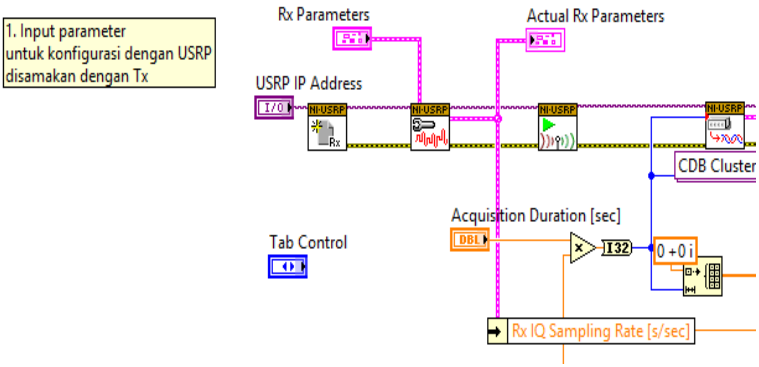


Gambar 3.18 Perancangan sub sistem langkah 4 dalam pemancar

Pada gambar 3.18 diatas adalah lanjutan dari *input text* yang diberikan dalam bentuk *string*. Setelah diberikan masukan maka data *string* terhubung dengan sub vi Text to Bits dimana pada sub ini terjadi pengolahan dari yang mulanya berupa data *string* berubah menjadi *bitstream* agar dapat diolah oleh modulasi. Didalam sub Text to Bits ini terdapat program AX.25 *encoder*. Setelah dimodulasi, *bitstream* dikemas dalam suatu bentuk *Array* lalu diproses untuk mempersiapkan sinyal untuk dipresentasikan pada grafik yang menunjukkan lokasi simbol yang terdeteksi dan transisi antara simbol-simbol.

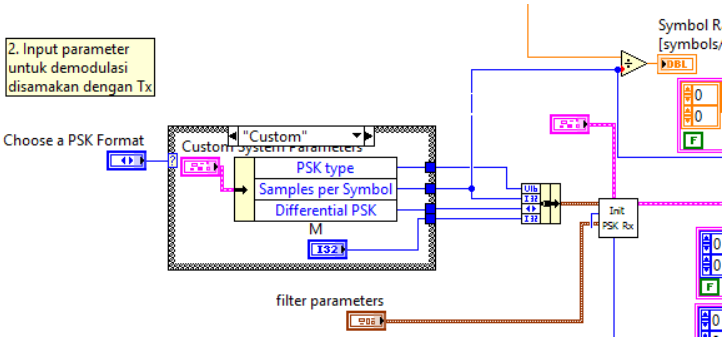
3.3.2.1 Sub Sistem PenerimaPada LabView 2014

Subsistem penerima ini berisi program-program sistem mulai dari konfigurasi parameter dalam Labview hingga terkonfigurasi dengan USRP, Subsistem modulasi, dan subsistem *decoder*.



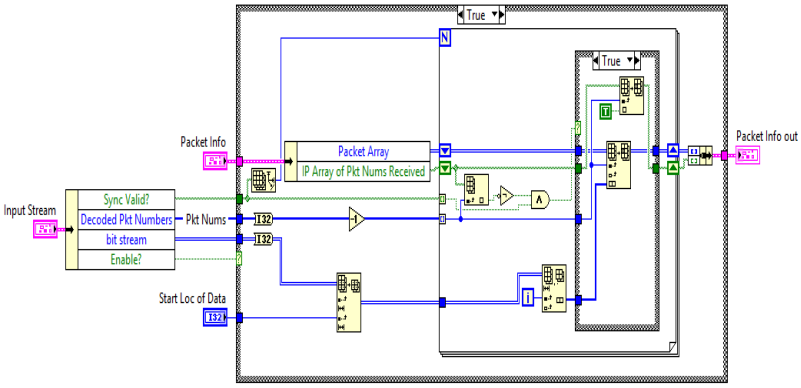
Gambar 3.19 Perancangan sub sistem langkah 1 dalam penerima

Pada langkah pertama adalah sama dengan transmitter, dimana memasukkan parameter untuk konfigurasi dengan perangkat USRP yang dimana nilainya disamakan dengan transmitter kecuali bagian USRP IP Address, USRP Address pada *receiver* sebagai pembuka sesi penerima (Rx) ke perangkat yang ditentukan dalam nama perangkat dan menangani kembali sesi instrument untuk VI NI-USRP berikutnya.



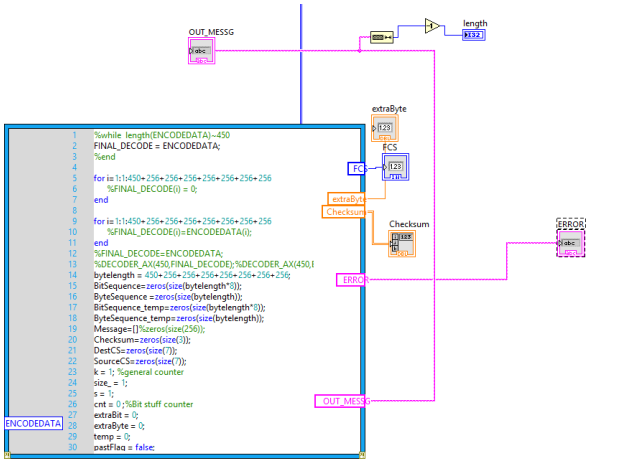
Gambar 3.20 Perancangan sub sistem langkah 2 dalam penerima

Pada langkah kedua didalam Gambar 3.20 adalah memasukkan parameter modulasi yang digunakan dan disamakan dengan transmitter, seperti tipe modulasi dan *samples per symbol*. *Samples per symbol* akan diolah dengan *IQ Rate* yaitu dengan operasi pembagian maka menghasilkan *symbol rate*.



Gambar 3.21 Perancangan sub sistem langkah 3 dalam penerima

Pada langkah ketiga didalam gambar 3.21 adalah sub sistem *reconstruct data message*, *input stream* yang didapat dari transmitter, seperti *bit stream* yang diolah di proses *decoder*.



Gambar 3.22 Perancangan sub sistem langkah 4 dalam penerima

Pada langkah keempat didalam gambar 3.22 adalah proses *decoder*, *bit stream* yang di rekonstruksi di receiver akan menjadi masukkan bagi program *decoder* AX.25. program *decoder* ini memproses bitstream menjadi bentuk *string* dengan karakter yang sama dengan transmitter kirim dan dapat dibaca oleh manusia.

3.3.2 Perancangan Pengujian AX.25 dengan keseluruhan Sistem

Perancangan pengujian dilakukan dengan menentukan konfigurasi sistem pada Labview, Menentukan Waktu dan Lokasi, dan Menentukan jumlah parameter teks yang akan dikirim.

Pada tugas akhir ini, perangkat yang digunakan sebagai pemancar dan penerima adalah dua USRP N210 dengan *daughterboard* LFTX dan LFRX. USRP N210 merupakan perangkat single *device* dengan single *channel*. Untuk menerjemahkan data dari host (laptop) ke USRP dan sebaliknya, perlu program tambahan untuk konfigurasi antara LabVIEW dan perangkat keras USRP. Beberapa parameter-parameter utama saat konfigurasi USRP pada LabVIEW:

1. ID dari perangkat USRP (*Device* ID's). Parameter ini berfungsi untuk menentukan USRP yang akan tersambung. Biasanya ID USRP berupa IP *address* USRP tersebut dengan default 192.168.10.4, 192.168.10.5.

2. Menetukan kanal RF yang digunakan dalam perangkat USRP. Selanjutnya, setiap kanal tersebut akan diatur parameter seperti *active Antenna* dan *gain*.
3. Menentukan *port* antena yang digunakan (*active Antenna*), misalnya TX1 merupakan *port* yang bisa digunakan untuk pemancar dan RX1 dan RX2 merupakan *port* yang bisa digunakan untuk penerima.
4. *IQ Rate* yang merupakan sampel per detik pada LabVIEW dan USRP yang diproduksi perusahaan National Instrument, *IQ Rate* yang tepat akan mengatur bandwidth maksimum, dimana *IQ Rate* transmitter sama dengan *receiver*.
5. *Symbol rate* merupakan *symbols per second* (simbol per detik) akan mengatur perkiraan "*actual bandwidth*". *Symbol rate* menunjukkan jumlah simbol yang ditransmisikan per detik (simbol / s). Untuk mengonversi *symbol rate* ke *bit rate*, yang menyatakan jumlah bit yang ditransfer per detik, gandakan laju simbol dengan jumlah bit per simbol yang digunakan dalam skema modulasi digital yang menarik. *Symbol rate* juga dikenal sebagai baud *rate*
6. Frekuensi carrier yang akan digunakan. Untuk frekuensi yang digunakan transmitter sama dengan *receiver*.
7. Besarnya *gain* dari pemancar ataupun penerima dimana *gain* di sini bukanlah *gain Antenna* tetapi *gain* dari USRP sesuai dengan spesifikasi perangkat keras USRP tersebut.
8. Spesifikasi modulasi yang digunakan di tentukan BPSK dengan sample per symbol yang diinginkan.

Perancangan AX 25 pada kesuluruhan sistem dilakukan pada tanggal 4-6 Mei 2019 di Gedung AJ dan Gedung B dengan jarak yaitu ± 30 meter.

Gambar 3.23 *Link* Antena Gedung AJ dan B

Uji coba kedua pada tanggal 24-26 Mei 2019 untuk jarak jauh di Gedung Depertemen Teknik Elektro ITS yang terletak di Surabaya ke Gedung TI VEDC yang berada di Malang yaitu ± 74 kilometer.

Gambar 3.24 *Link* Surabaya - Malang

Perancangan AX 25 dengan keseluruhan sistem akan diuji dengan beberapa macam jumlah karakter yang akan dikirimkan yaitu 10, 200, 500, 1000 dan 2000 karakter dengan mencatat waktu kecepatan proses dikirim hingga diterima 5 kali pengujian data. Untuk parameter pada USRP yang akan diuji dapat dilihat pada Tabel 3.3.

Tabel 3.3. Daftar Konfigurasi Labview dengan Perangkat USRP

No	Konfigurasi	Nilai
1	USRP IP *Address* TX	192.168.10.4
2	USRP IP *Address* RX	192.168.10.7
3	Frekuensi Carrier	7 – 8 MHz
3	IQ *Rate*	200k
4	Symbol *Rate*	10k – 100k
5	Modulasi	BPSK
6	*Port Antenna* TX	RF1
7	*Port Antenna* RX	RF1

BAB 4
HASIL PENGUJIAN DAN ANALISA

Setelah melakukan tahap perancangan dan implementasi AX.25 ke sistem, berikutnya dilakukan pengujian dan analisa dari hasil implementasi protokol AX.25 apakah dapat berjalan sesuai fungsinya sebagai *encoder* dan *decoder*. Tahap pengujian dan analisa ini dimaksudkan untuk mengetahui keandalan dari pengiriman data dari protokol AX.25.

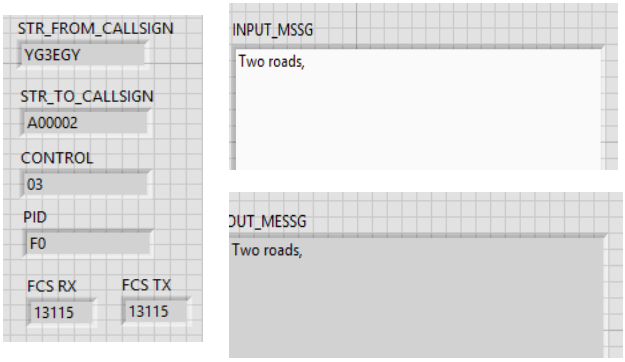
4.1 Hasil Proses Pengujian

Pada proses pengujian ini diuji dengan urutan langkah pengujian adalah meguji algoritma dan *design* protokol AX.25 di labview dengan alur sederhana (tanpa sistem keseluruhan) dan menguji *design* protokol AX.25 dengan alur dalam sistem komunikasi digital (keseluruhan). Pengujian kecepatan proses secara keseluruhan dari pengirim ke penerima juga menjadi pertimbangan kedepannya.

4.4.1 Hasil Pengujian Simulasi dengan Protokol AX.25

Pengujian Simulasi dengan Protokol AX.25 dilakukan dengan memberikan banyak karakter dengan jumlah yang diberikan adalah 10,200,500,1000,2000 karakter sebagai *input* yang ditempatkan pada *Field* informasi.

1. Pengujian pertama dengan 10 karakter yaitu sesuai dengan Gambar 4.1. Struktur *frame* AX.25 sesuai dengan yang dirancang pada bab 3.



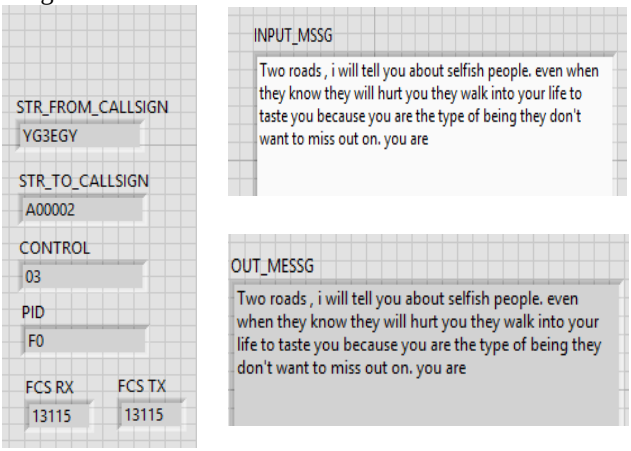
Gambar 4.1 Tampilan Struktur *Frame* dan *Input Output* 10 Karakter

Tabel 4.1. Hasil Pengujian Kecepatan Proses 10 karakter

Data Ke-	Waktu	Keterangan
1	3.75 Detik	BERHASIL
2	2.78 Detik	BERHASIL
3	3.26 Detik	BERHASIL
4	3.60 Detik	BERHASIL
5	3.16 Detik	BERHASIL

Setelah dilakukan uji coba sebanyak 5 kali, dari hasil Tabel 4.1 dihasilkan waktu terlama yaitu 3.75 detik dan untuk yang tercepat adalah 2.78 detik. Jadi rata rata waktu yang dilakukan adalah 3,16 detik.'

2. Hasil pengujian kedua dengan 200 karakter karakter yaitu sesuai dengan Gambar 4.2.



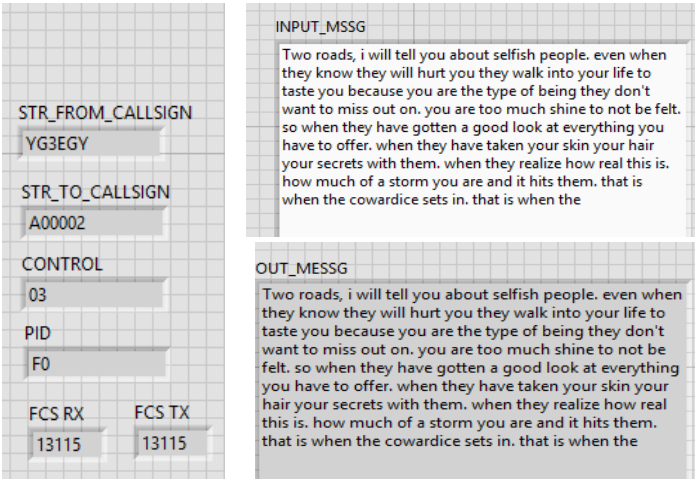
Gambar 4.2 Tampilan *Field* dan *Input Output* 200 Karakter

Tabel 4.2. Hasil Pengujian Kecepatan Proses 200 karakter

Data Ke-	Waktu	Keterangan
1	5.12 Detik	BERHASIL
2	5.3 Detik	BERHASIL
3	5.25 Detik	BERHASIL
4	5.41 Detik	BERHASIL
5	5.17 Detik	BERHASIL

Setelah dilakukan uji coba sebanyak 5 kali dari hasil Tabel 4.2 dihasilkan waktu terlama yaitu 5.41 detik dan untuk yang tercepat adalah 5.17 detik. Jadi rata rata waktu yang dilakukan adalah 5,2 detik.

3. Hasil pengujian ketiga dengan 500 karakter yaitu sesuai dengan Gambar 4.3:



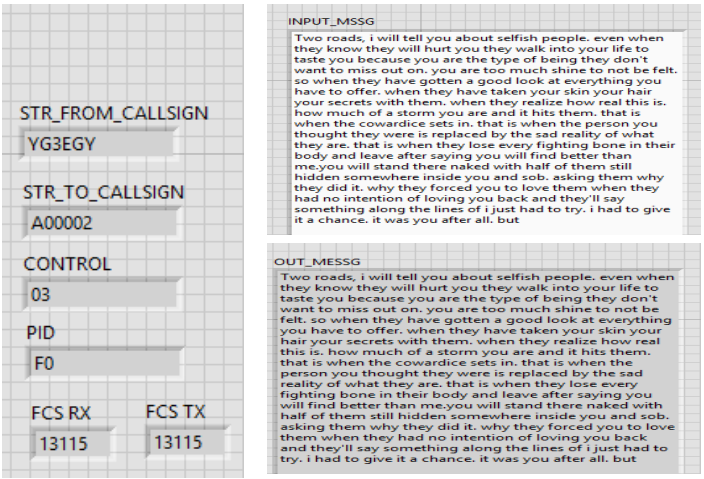
Gambar 4.3 Tampilan *Field* dan *Input Output* 500 Karakter

Tabel 4.3. Hasil Pengujian Kecepatan Proses 500 karakter

Data Ke-	Waktu	Keterangan
1	7.52 Detik	BERHASIL
2	7.67 Detik	BERHASIL
3	7.44 Detik	BERHASIL
4	7.37 Detik	BERHASIL
5	7.22 Detik	BERHASIL

Setelah dilakukan uji coba sebanyak 5 kali dari hasil tabel 4.3 dihasilkan waktu terbesar yaitu 7.67 detik dan untuk yang tercepat adalah 7.44 detik. Jadi rata rata waktu yang dilakukan adalah 7.22 detik.

4. Hasil pengujian keempat dengan memberikan karakter sebanyak 1000 karakter yaitu sesuai dengan Gambar 4.4:



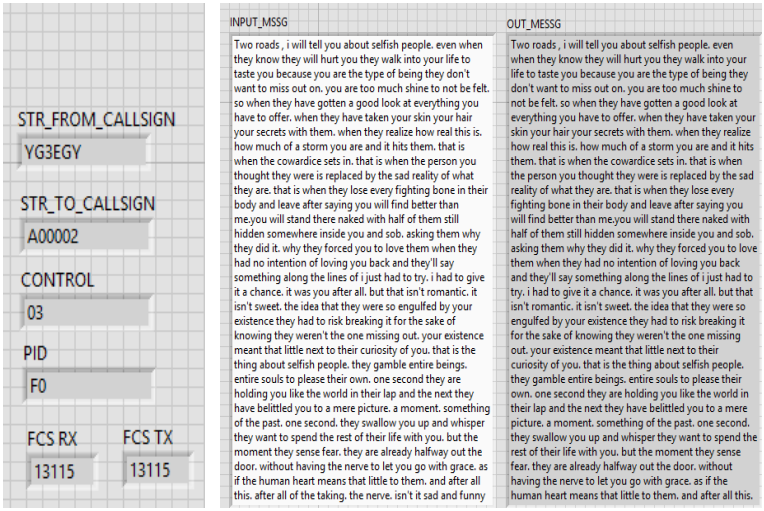
Gambar 4.4 Tampilan *Field* dan *Input Output* 1000 Karakter

Tabel 4.4. Hasil Pengujian Kecepatan Proses 1000 karakter

Data Ke-	Waktu	Keterangan
1	9.38 Detik	BERHASIL
2	9.52 Detik	BERHASIL
3	9.74 Detik	BERHASIL
4	9.27 Detik	BERHASIL
5	9.43 Detik	BERHASIL

Setelah dilakukan uji coba sebanyak 5 kali dari hasil tabel 4.4 dihasilkan waktu terlama yaitu 9,74 detik dan untuk yang tercepat adalah 9,27 detik. Jadi rata rata waktu yang dilakukan adalah 9,47 detik.

5. Pengujian kelima dengan 2000 karakter yaitu sesuai dengan Gambar 4.5:



Gambar 4.5 Tampilan *Field* dan *Input Output* 2000 Karakter

Tabel 4.5. Hasil Pengujian Kecepatan Proses 2000 karakter

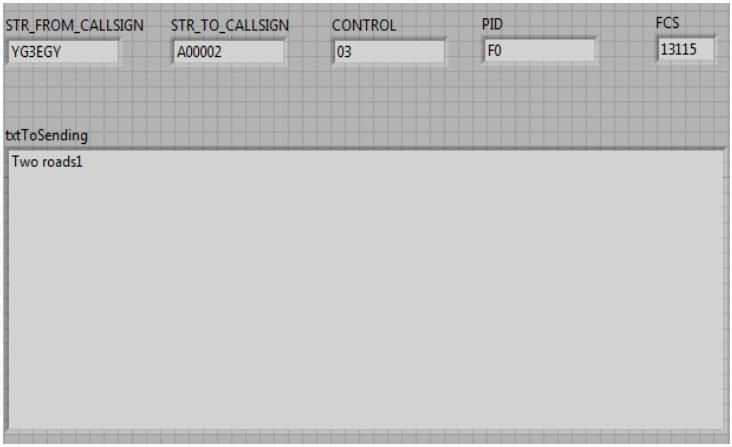
Data Ke-	Waktu	Keterangan
1	15.13 Detik	BERHASIL
2	14.49 Detik	BERHASIL
3	15.20 Detik	BERHASIL
4	15.33 Detik	BERHASIL
5	15.28 Detik	BERHASIL

Setelah dilakukan uji coba sebanyak 5 kali dari hasil tabel 4.5 dihasilkan waktu terlama yaitu 15.33 detik dan untuk yang tercepat adalah 14.49 detik. Jadi rata rata waktu yang dilakukan adalah 15.09 detik.

4.4.2 Hasil Pengujian Protokol AX.25 dengan Sistem Komunikasi

Pengujian Algoritma Protokol AX.25 bersama dengan sistem komunikasi digital secara keseluruhan, dilakukan di lokasi gedung AJ dan B. Memberikan banyak karakter dengan jumlah yang diberikan adalah 10,200,500,1000,2000 karakter sebagai *input* yang ditempatkan pada *Field* informasi :

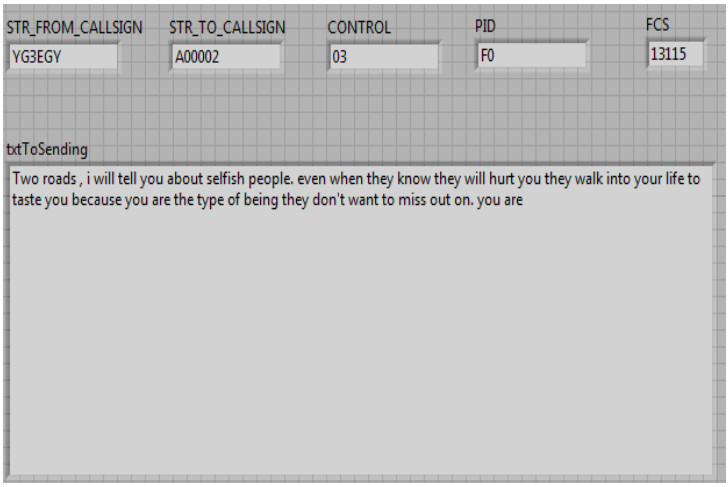
1. Pengujian pertama dengan 10 karakter yang di ketikkan pada tampilan panel utama dengan nama "*Message Text*". Karakter akan muncul dalam *field Information setelah header* yang diberikan dan sebelum *field PID.* Sebelumnya jangan lupa untuk menyesuaikan parameter-parameter yang menghubungkan dengan USRP seperti *IP Address.*



Gambar 4.6 Tampilan *Field* dengan 10 karakter

Dengan menggunakan 10 karakter untuk 5 kali uji coba dan parameter tambahan berupa IQ *rate* 200k/s dan *symbol rate* 100k sym/sec, 10 karakter dapat diterima dengan baik tanpa ada kesalahan data dengan durasi waktu pengiriman rata-rata 20 detik. Pada saat pengujian dengan *symbol rate* kurang dari 30k sym/sec pesan tidak dapat diterima.

2. Pengujian kedua dengan 200 karakter yang di ketikkan pada tampilan panel utama dengan nama "*Message Text*". Karakter akan muncul dalam *field Information setelah header* yang diberikan dan sebelum *field PID.* Sebelumnya jangan lupa untuk menyesuaikan parameter-parameter yang menghubungkan dengan USRP seperti *IP Address.*



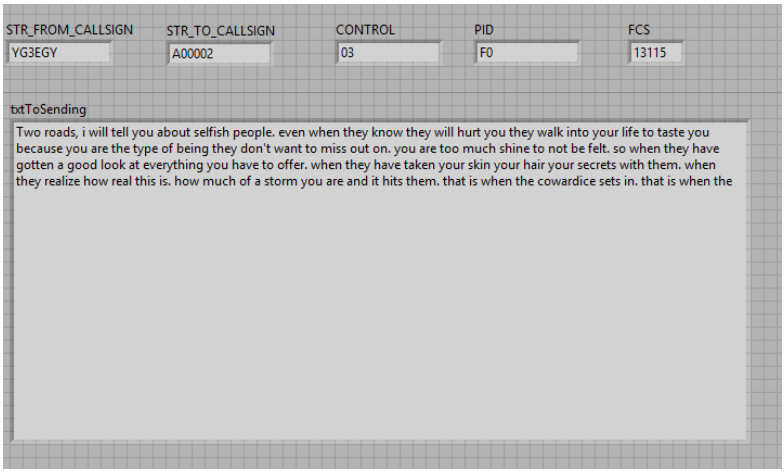
Gambar 4.7 Tampilan *Field* dengan 200 karakter

Dengan menggunakan 200 karakter untuk 5 kali uji coba dan parameter tambahan berupa IQ *rate* 200k/s dan *symbol rate* 100k sym/sec, 200 karakter dapat diterima dengan baik tanpa ada kesalahan data dengan durasi waktu pengiriman rata-rata 20 detik. Pada saat pengujian dengan *symbol rate* kurang dari 30k sym/sec pesan tidak dapat diterima.

3. Pengujian ketiga dengan 500 karakter.

Karakter dengan jumlah 500 karakter di ketikkan pada tampilan panel utama dengan nama "*Message Text*". Karakter akan muncul dalam *field Information setelah header* yang diberikan dan sebelum *field PID.* Sebelumnya jangan lupa untuk menyesuaikan parameter-parameter yang menghubungkan dengan USRP seperti *IP Address.*

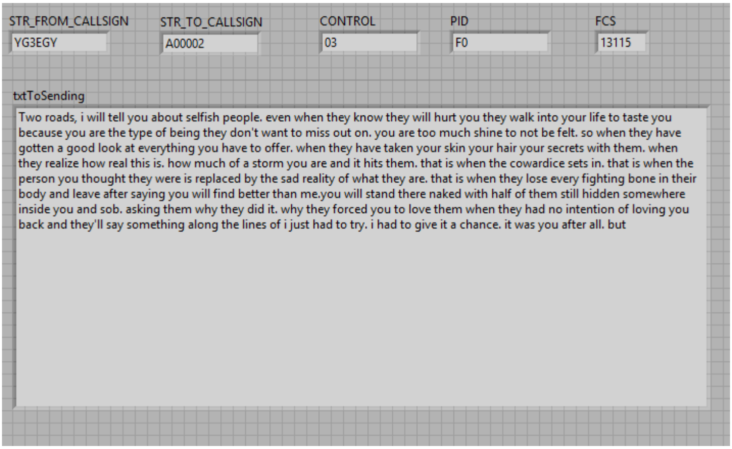
Dengan menggunakan 500 karakter untuk 5 kali uji coba dan parameter tambahan berupa IQ *rate* 200k/s dan *symbol rate* 100k sym/sec, 500 karakter dapat diterima dengan baik tanpa adanya kesalahan data dengan durasi waktu pengiriman rata-rata 30 de`tik. Pada saat pengujian dengan *symbol rate* kurang dari 30k sym/sec pesan tidak dapat diterima.



Gambar 4.8 Tampilan *Field* dengan 500 karakter

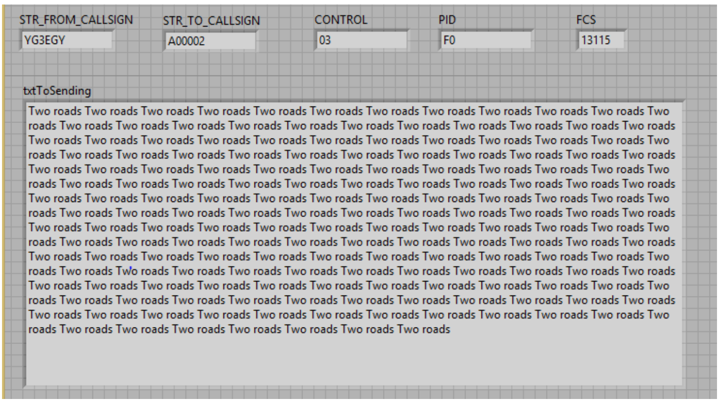
4. Pengujian keempat dengan 1000 karakter.
   Karakter dengan jumlah 1000 karakter di ketikkan pada tampilan panel utama dengan nama "*Message Text*". Karakter akan muncul dalam *field Information setelah header* yang diberikan dan sebelum *field PID.* Sebelumnya jangan lupa untuk menyesuaikan parameter-parameter yang menghubungkan dengan USRP seperti *IP Address.*

   Dengan menggunakan 1000 karakter untuk 5 kali uji coba dan parameter tambahan berupa IQ *rate* (bandwidth maksimal) 200k/s dan *symbol rate* (*actual bandwidth*) 100k sym/sec, 1000 karakter dapat diterima dengan baik tanpa adanya kesalahan data dengan durasi waktu pengiriman rata-rata 30 detik. Pada saat pengujian dengan *symbol rate* kurang dari 30k sym/sec pesan tidak dapat diterima yang berarti pesan terjadi kesalahan pada saat diterima atau tidak diterima sama sekali.



Gambar 4.9 Tampilan *Field* dengan 1000 karakter

5. Pengujian kelima dengan 2000 karakter yang diketikkan pada tampilan panel utama dengan nama “*Message Text*”. Karakter akan muncul dalam *field Information setelah header* yang diberikan dan sebelum *field PID.* Sebelumnya jangan lupa untuk menyesuaikan parameter-parameter yang menghubungkan dengan USRP seperti *IP Address.*



Gambar 4.10 Tampilan *Field* dengan 2000 karakter

Dengan menggunakan 2000 karakter untuk 5 kali uji coba dan parameter tambahan berupa IQ *rate* 200k dan *symbol rate* 100k

sym/sec, 2000 karakter dapat diterima dengan baik tanpa adanya kesalahan dalam data dengan durasi waktu pengiriman rata-rata 60 detik. Pada saat pengujian dengan *symbol rate* kurang dari 30k sym/sec pesan tidak dapat diterima.

4.2 Analisa

Pada perancangan dan pembuatan dengan menggunakan *software* Labview dibantu dengan *Mathscript Node* yang disediakan oleh Labview. Langkah awal yang dilakukan adalah merancang alur untuk mengambil data dari masukan pesan/teks dan menampilkan di bagian panel / tampilan depan Labview dengan tampilan struktur *frame* AX.25.

Pada dasarnya algoritma AX.25 yang telah dirancang bekerja dengan prosedur yaitu *encoder* mengubah masukan berupa karakter menjadi barisan bit yang diterima oleh *decoder* untuk diolah menjadi karakter kembali.

Pembuatan satu *frame* AX.25 dengan *sub-field* yang ditentukan, yaitu penambahan *Header* yang terdiri dari *Flag, Address, Control* dan PID. *Flag* 01111110 berupa 1 *flag* karakter yang digunakan untuk memisahkan *frame*s. *Flag* ini tidak ditampilkan sebagai informasi pada tampilan di Labview dikarenakan pada *decoder flag* ini akan dihapus dan hanya informasi yang disampaikan. Setelah menentukan *flag* menambah destination *callsign* (*callsign* tujuan) sebanyak 8 bit dimana didalamnya terdapat ssid tujuan dan *callsign* sumber sebanyak 8 bit dengan ssid sumber didalamnya. Setelah menambahkan *field* address, maka langkah selanjutnya menambahkan *control bits* dengan nilai dari control bits adalah 0×03 yang artinya menggunakan tipe *Unnumbered Information Frames.* Menambahkan Protokol Identifier (PID) dengan nilai 0×F0 ynag berarti tidak menggunakan protokol *layer* . *Header* diletakkan sebelum *field* informasi.

Setelah itu penambahan pesan/informasi, *field* FCS dan *flag*. *Field* informasi ini dapat menampung sebanyak 256 *byte* dalam 1 *frame.* FCS disini menggunakan kalkulasi CRC dengan generator polynomial. Dalam proses *encoder* ini semua proses bertujuan untuk menjadikan karakter menjadi barisan bit teteapi menyesuaikan dengan format *frame* AX.25. Dalam proses *decoder* mengolah barisan bit dari *encoder* untuk diterjemahkan ke dalam karakter.

Pengujian simulasi dengan program AX.25 tanpa perangkat USRP, dapat menampilkan data pada tampilan depan Labview dengan struktur *frame* yang ditentukan dan informasi karakter 10, 200 karakter dalam 1

frame. 500 karakter dalam 2 *frame*, 1000 karakter dalam 4 *frame* dan 2000 karakter delam 8 *frame*. Pengujian kecepatan proses penerimaan informasi mendapatkan hasil waktu rata-rata 3,16 – 15,09 detik, semakin banyak jumlah karakter yang diberikan maka semakin lama waktu yang dibutuhkan untuk dapat menampilkan informasi.

Pengujian program AX.25 dengan sistem komunikasi keseluruhan untuk jarak dekat antara antena gedung AJ dan B, sistemnya berhasil mengubah bentuk *string* ke dalam *bitstream* lalu diterima di *receiver* diubah ke *string* kembali dengan modulasi BPSK yang memiliki arti symbol = bit, maka *symbol rate* 100kb/sec durasi rata-rata dari 20 hingga 60 detik, tetapi dengan *symbol rate* kurang dari 30kb/sec pesan tidak dapat diterima sama sekali.

[Halaman ini sengaja dikosongkan]

BAB 5
PENUTUP

5.1 Kesimpulan

Berdasarkan dari perancangan yang telah diimplementasikan dan hasil analisa dari pengujian data yang telah dilakukan, beberapa hal yang dapat disimpulkan adalah :

1. Pengujian simulasi dengan protokol AX.25 tanpa perangkat keras USRP, menghasilkan kinerja yang baik. Hal ini dapat diketahui dari hasil pengujian dimana dari hasil pengujian total sebanyak 25 kali pengiriman tidak ditemui adanya kesalahan pada informasi yang diterima dengan rata-rata waktu pengiriman 3.16 – 15.09 detik dengan jumlah karakter 10 - 2000.
2. Dalam sistem komunikasi menggunakan frekuensi tinggi dengan pengujian jarak dekat yaitu antara Antena Gedung AJ dan B teknik Elektro ITS, protokol AX.25 dan sistem dapat beroperasi dengan baik, dikarenakan dari hasil pengujian 25 kali pengiriman tidak ditemukan adanya kesalahan dengan waktu pengiriman sekitar 20 – 60 detik untuk jumlah karakter dari 10, 200,500, 1000 dan 2000 karakter.
3. Pengaturan *symbol rate* / bandwidth yang digunakan hingga semua pesan dapat tersampaikan adalah 100kbps berbanding terbalik apabila *symbol rate* kurang dari 30kbps keseluruhan pesan tidak dapat diterima dengan modulasi BPSK.

5.2 Saran

Berikut merupakan beberapa saran yang untuk dilakukan penelitan lanjutan mengenai topik yang serupa:

a) Akan jauh lebih baik jika dalam penelitian selanjutnya disarankan menerjemahkan program AX.25 yang menggunakan *mathscript node* di jadikan dalam bentuk diagram dengan *tool* yang sudah disediakan oleh Labview. Untuk mendukung kinerja dari Labview agar lebih cepat.
b) Apabila memungkinkan, program sistem perlu dibuat sesederhana mungkin dan sesuai dengan sistem intinya saja. Atau membuat sistem yang baru.

[Halaman ini sengaja dikosongkan]

DAFTAR PUSTAKA

[1] AlFaruq, Daviq, Umar. "TNI Pasang SSB dan *Repeater* untuk Komunikasi di Palu", MetroTV News, Jawa Timur, 4 Oktober 2018

[2] L.F. McNamara, "*The Ionosphere: Communications, Surveillance, and Direction Finding*". Krieger Publishing, 1991

[3] Manalu, "*Design of measurement system for HF MIMO NVIS channel*", 4th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE), Semarang, hal. 300-305,2017

[4] Arung, Pasang, "Rancang Bangun Modul PAD (Packet Data Assembler Disassembler) Menggunakan AX-25 pada Sistem Komunikasi ITS-SAT", Jurnal Teknik Pomits, Vol. 1 No.1, Surabaya,2013

[5] ASAPS V4 tutorial, "*IPS Radio and Space Services, Introduction to HF Radio Propagation*", Australian Government, 2007

[6] Stallings, William, "*Data and Computer Communications*". Pearson. 2007

[7] Beech, William A., Nielsen, Douglas E., Taylor, Jack., "*AX.25 Link Acces Protokol v2.2*",TAPR. 1996

[8] Hands-on Course, "*An Introduction to Software Defined Radio with Labview on NI USRP*", Natinal Instrument.

[9] M. Bosco, V.P. Castelli, P. Tortora: '*Design and Implementation of Software Solutions for Satellite Ground Segment, with Application to the ESEO Mission*', 6 May 2016.

[10] Fred Halsall. *Data Communications,Computer Networks and Open Systems.* Addision-Wesley Publishing Company Inc.. 1995. ISBN: 0-20-42293-X

[11] C. Bennett, "The overheads of transnetwork fragmentation," *Comput. Networks*, vol. 6, pp. 21–36, Feb. 1982.

[12] C. Stuart, B. Maker. "*Consistent Overhead Byte Stuffing*," Trans. On Networking, vol. 7. 2 April 1999.

[13] S. Davidoviel, R Simpson. "*Detectability of Packet Communications Using Variants of the X.25 Protocol,*" 1990.

[Halaman ini sengaja dikosongkan]

LAMPIRAN A
PENGESAHAN PROPOSAL TA

Departemen Teknik Elektro
Fakultas Teknologi Elektro – ITS

EE 184801 TUGAS AKHIR – 6 SKS

Nama Mahasiswa	: Vina Amalia Fitrianingrum
Nomor Pokok	: 07111745000022
Bidang Studi	: Telekomunikasi Multimedia
Tugas Diberikan	: Semester Genap Tahun 2018 / 2019
Dosen Pembimbing	: 1. Prof. Ir. Gamantyo Hendrantoro, M.Eng., Ph.D 2. Dr. Ir. Achmad Mauludiyanto, MT.
Judul Tugas Akhir	:**Perencanaan Protokol Data-Link Layer untuk Sistem Komunikasi Radio High Frequency NVIS Berbasis USRP** **(*Data-Link Layer Protocol Planning for High Frequency NVIS Radio Communication Systems USRP based*)**

12 FEB 2019

Uraian Tugas Akhir :
Salah satu sistem komunikasi radio yang saat ini sedang berkembang adalah menggunakan gelombang *High Frequency*. Sistem komunikasi HF yang sekarang terbatas hanya pada pengiriman suara (orari). Namun seiring berkembangnya jaman, data komunikasi digital era sekarang menginginkan pengiriman sudah tidak hanya berupa suara, melainkan bervariasi seperti teks, gambar dll. Untuk memenuhi kebutuhan tersebut, maka dilakukan pengukuran dan pembuatan suatu protokol komunikasi . Link Surabaya – Malang diambil karena termasuk dalam jarak *Near-Vertical Incidence Skywave (NVIS)* merujuk kepada pancaran (sinyal) radio di band HF, yang sinyalnya dipantulkan lapisan ionosfer jatuh kembali ke area yang berjarak sekitar 0~400 Km dari asal pancaran.

Tahapan pertama di layer 2 yaitu layer datalink. Perancangan dari sisi perangkat keras menggunakan USRP, GPS Antena dan menggunakan antena dipole sedangkan pada sisi perangkat lunak, yang dipakai adalah program untuk menyusun algoritma pemrograman bisa bahasa C, C++ serta Labview Communication 1.1. Pada tugas akhir ini memiliki konsentrasi di layer datalink dengan mengimplementasikan suatu protocol komunikasi yang compatible dengan sistem komunikasi HF, sehingga Perencanaan Protokol untuk Sistem Komunikasi Radio *High Frequency* NVIS Berbasis USRP dengan link Surabaya- Malang berhasil.
Kata Kunci : *High Frequency* , SISO, NVIS, Protokol.

Dosen Pembimbing I,

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

Dosen Pembimbing II,

Dr. Ir. Achmad Mauludiyanto, MT.
NIP : 196109031989031001

Mengetahui,
Ketua Program Studi S1

Dedet C. Riawan, ST., M.Eng., Ph. D.
NIP : 197311192000031001

Menyetujui,
Kepala Laboratorium Antena dan Propagasi

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

[Halaman ini sengaja dikosongkan]

LAMPIRAN B

 # Listing program *encoder* dan *decoder*

ENCODER

```
%============================
%AX25_MAX_PAYLOAD_SIZE = 160_maximum payload size in bytes (by RF22 library)
CRC_POLYGEN  = hex2dec('1021');
MAX_LENGTH = 280;
MAX_LENGTH_FINAL = 450+256+256+256+256+256+256+254;
BYTE_NUM = 8;
MSG_LENGTH = 256;
%====================================
bitsequence    =zeros(size(MAX_LENGTH*BYTE_NUM));
finalSequence =zeros(size(MAX_LENGTH_FINAL));
RcvSequence  = zeros(size(MAX_LENGTH_FINAL));
messages  = [];%zeros(size(MSG_LENGTH));
%====================================
STR_FROM_CALLSIGN ='YG3EGY';
STR_TO_CALLSIGN ='A00002';
CONTROL = '03';%'0x03';
PID ='F0';%'0xF0';
%====================================
SrcCallsign=['Y','G','3','E','G','0','Y'];%zeros(size(7));
DestCallsign=['A','0','0','0','0','0','2'];%zeros(size(7));
ssid_source = 0;
ssid_destination = 0;
%====================================
%FCS = uint64(0);

c =1;
for i=1:1:(MAX_LENGTH*BYTE_NUM)
    bitsequence(c)=0;
    c=c+1;
end
c=1;
for  i=1:1:(MAX_LENGTH_FINAL)
     finalSequence(c) = 0;
```

```
      RcvSequence(c) = 0;
      c=c+1;
end;
c=1;

for  i=1:1:(MSG_LENGTH)
      messages(c) = hex2dec('00');%white space
      c=c+1;
end;

%====================
% TEST MESSAGES
txtToSending= reshape(INPUT_MSSG, 1, []);
%str copy to messages
c =1;
for i=1:1: length(txtToSending)
     messages(c) = txtToSending(1,c);
     c = c+1;
end

%=============================================
%ADD HEADER
%Shift bits 1 place to the left in order to allow for HDLC extension bit
%=============================================

c =1;
Index =1;
for i=1:1:length(DestCallsign)
   %Append SSID Destination
   bitsequence(Index) = bitshift((uint16(DestCallsign(i))),1);
   c = c+1;
   Index = Index+1;
end

bitsequence(Index) = hex2dec('60');%ssid_destination;
Index = Index+1;

%==================================
%Append Source Callsign
```

```
%===================================
c =1;
for i=1:1:length(Src_Callsign)
   bitsequence(Index) = bitshift(uint16(Src_Callsign(i)),1);
   %fprintf(num2str(bitsequence(Index)));
   %fprintf(' ');
   Index = Index+1;
   c =c+1;
end

bitsequence(Index) = hex2dec('61');%ssid_source;
Index = Index+1;
%===================================
%Append Control bits
bitsequence(Index)  = hex2dec(CONTROL);
Index = Index+1;

%Append Protokol Identifier
bitsequence(Index)  = hex2dec(PID);
Index = Index+1;
%===================================
%Add Message
c =1;
for i=1:1: length(txtToSending)
     bitsequence(Index) = txtToSending(1,c);
     Index = Index+1;
     c = c+1;
end
%============================
%Convert bit sequence from MSB to LSB
v  =0;
c  =1;
p  =0;

for i=1:1:Index-1
  v = (bitsequence(i));
  bitsequence(i) = MSB_LSB_swap_8bit(v);
end
```

```
%AFTER MSB LSB SWAP
%==========================================

FCS =
13115;%typecast(int16(13115),'uint16');%bin2dec('11001100111011');
%13115;%CRC_CCITT(bitsequence,Index-1);
fprintf('FCS:');
fprintf(num2str(FCS));
fprintf('\n');

%Add FCS in MSB form
%Add MS byte
%bitSequence[Index++] = (FCS >> 8) & 0xff;
%fprintf('FCS:');
%fprintf(num2str(FCS));
%fprintf('\n');
p = bitand(bitshift(FCS,-8),hex2dec('FF'));

fprintf(':');
fprintf(num2str(p));
fprintf('\n');
bitsequence(Index) =p;% bin2dec(num2str(p));
%fprintf(num2str(p));
%fprintf('\n');
Index = Index + 1;
%Add LS byte
%bitSequence[Index++] = FCS & 0xff;
p = bitand(FCS,hex2dec('FF'));
%fprintf(num2str(p));
%fprintf('\n');
bitsequence(Index) = p;%bin2dec(num2str(p));
Index = Index + 1;

%==========================================

fprintf('Length : ');
for i=1:1:Index-1
    fprintf(num2str(bitsequence(i)));
    fprintf(' ');
```

```
end
fprintf('\n');

%FINAL RESULT ENCODER

%FINAL_RESULT = (1:450);

[ENCODE,indexs] = BIT_PROCESSING(bitsequence,Index-
1,finalSequence);

for i=1:1:450+256+256+256+256+256+256+256
    FINAL_RESULT(i)=char(ENCODE(i));
end

%DECODER

DECODER
%while
length(ENCODEDATA)~450+256+256+256+256+256+256+256
FINAL_DECODE = ENCODEDATA;
%end

%FINAL_DECODE=ENCODEDATA;
%DECODER_AX(450,FINAL_DECODE);%DECODER_AX(450,ENC
ODE);
byte length = 450+256+256+256+256+256+256+256;
BitSequence=zeros(size(byte length*8));
Byte Sequence =zeros(size(byte length));
BitSequence_temp=zeros(size(byte length*8));
Byte Sequence_temp=zeros(size(byte length));
Message=[]%zeros(size(256));
Checksum=zeros(size(3));
DestCS=zeros(size(7));
SourceCS=zeros(size(7));
k = 1; %general counter
size_ = 1;
s = 1;
cnt = 0 ;%Bit stuff counter
extraBit = 0;
```

```
extraByte = 0;
temp = 0;
pastFlag = false;
BitFound = false;


for i=1:1:(byte*length*8) BitSequence(i)=0; end
for i=1:1:(byte*length) Byte*Sequence(i)=0; end
for i=1:1:(byte*length*8) BitSequence_temp(i)=0; end
for i=1:1:(byte*length) BitSequence(i)=0; end
for i=1:1:450+256+256+256+256+256+256+256
   Message(i) = 0;
end

%Convert bits to byte size
%for (int i = 0; i< byte*length ; i++)
%{
%for (register uint8_t t=128; t>0 ; t = t/2) {
%if (Buffer[i] & t) BitSequence[_size++] = 0x01;
%else BitSequence[_size++] = 0x00;
%}
%}
%
size_ = 1;
arr_nums = [128,64,32,16,8,4,2,1];
for i=1:1:byte*length
  for x=1:1:8
   if bitand(FINAL_DECODE(i),arr_nums(x))
     BitSequence(size_) = 1;
   else
     BitSequence(size_) = 0;
   end
   size_=size_+1;
  end
end

for i=2:1:size_-1
   if BitSequence(i) == BitSequence(i-1)
     BitSequence_temp(i-1) = 1;
```

```
    else
      BitSequence_temp(i-1) = 0;
    end
end
k = 1;
%Convert bit to Byte
for i = 1:8:(size_-1)
    %fprintf(num2str(i));
    %fprintf(' ');
    temp = 0;
    if BitSequence_temp(i) == hex2dec('01')
       temp = temp + bin2dec('10000000');
    end
    if BitSequence_temp(i+1) == hex2dec('01')
       temp = temp + bin2dec('01000000');
    end
    if BitSequence_temp(i+2) == hex2dec('01')
       temp = temp + bin2dec('00100000');
    end
    if BitSequence_temp(i+3) == hex2dec('01')
       temp = temp + bin2dec('00010000');
    end
    if BitSequence_temp(i+4) == hex2dec('01')
       temp = temp + bin2dec('00001000');
    end
    if BitSequence_temp(i+5) == hex2dec('01')
       temp = temp + bin2dec('00000100');
    end
    if BitSequence_temp(i+6) == hex2dec('01')
       temp = temp + bin2dec('00000010');
    end
    if BitSequence_temp(i+7) == hex2dec('01')
       temp = temp + bin2dec('00000001');
    end
    ByteSequence(k) = temp;
    %fprintf(num2str(k));
    %fprintf('. ');
    %fprintf(num2str(ByteSequence(k)));
    %fprintf('\n');
```

```
    k = k+1;
end
fprintf(num2str(ByteSequence));
fprintf('\n');
pastFlag = false;
cnt = 1;
%Find and Remove Flags
%for (int i = 0; i < k; i++)
% {
% if (ByteSequence[i] != FLAG)
% {
% pastFlag = true;
% ByteSequence_temp[cnt++] = ByteSequence[i];
% } else if (pastFlag) break;
% }

for i =1:1:k-1
  if  ByteSequence(i) ~= bin2dec('01111110')
      pastFlag = true;
      ByteSequence_temp(cnt) = ByteSequence(i);
      cnt = cnt +1;
  elseif pastFlag == true
      break;
  end
end

%Re-init
%for (int i=0; i < bytelength*8 ; i++) BitSequence[i] = 0x00;
for i=1:1:bytelength*8
   BitSequence(i) = 0;
end
k = 1;
%Convert bits to byte size
% for (int i = 0; i< cnt ; i++)
% {
% for (register uint8_t t=128; t>0 ; t = t/2) {
%       if (ByteSequence_temp[i] & t) BitSequence[k++] = 0x01;
%       else BitSequence[k++] = 0x00;
%      }
```

```
%    }

for i =1:1:cnt-1
   for x=1:1:8
   if bitand(byteSequence_temp(i),arr_nums(x))
     BitSequence(k) = 1;
   else
     BitSequence(k) = 0;
   end
   k=k+1;
   end
end
%re-init
for i=1:1:bytelength*8
   byteSequence_temp(i) = 0;
end

%Bit unstuff : Remove 0 after five consecutive 1s.
cnt = 0;
s = 1;
BitFound = false;
extraBit = 0;

%for (int i = 0; i < k ; i++)
%    {
for i =1:1:k-1
   if BitFound
      BitFound = false;
      extraBit=extraBit+1;
      continue;
   end
   if BitSequence(i) == 1
      cnt = cnt+1;
   else
      cnt = 0; %restart count at 1
   end
   if cnt == 5 % there are five consecutive bits of the same value
     BitFound = true;
     cnt = 0; % and reset cnt to zero
```

```
    end
     BitSequence_temp(s) = BitSequence(i); % add the bit to the final
sequence
      s = s+1;
end

 extraByte = (extraBit / 8);
 if mod((extraBit),8) > 0
    extraByte = extraByte+1 ;
 end
%Re-init ByteSequence
%for (int i=0; i < bytelength ; i++) ByteSequence[i] = 0x00;
for i=1:1:bytelength
    ByteSequence(i) = 0;
end


%Convert bit to Byte
k =1;
fprintf('Receive stream::');
fprintf('\n');
fprintf('==========================================\n');
for i = 1:8:(s - 1 - extraByte*8)
    %fprintf(num2str(i));
    %fprintf(' ');
    temp = 0;
    if BitSequence_temp(i) == hex2dec('01')
       temp = temp + bin2dec('10000000');
    end
    if BitSequence_temp(i+1) == hex2dec('01')
       temp = temp + bin2dec('01000000');
    end
    if BitSequence_temp(i+2) == hex2dec('01')
       temp = temp + bin2dec('00100000');
    end
    if BitSequence_temp(i+3) == hex2dec('01')
       temp = temp + bin2dec('00010000');
    end
    if BitSequence_temp(i+4) == hex2dec('01')
```

```
        temp = temp + bin2dec('00001000');
    end
    if BitSequence_temp(i+5) == hex2dec('01')
        temp = temp + bin2dec('00000100');
    end
    if BitSequence_temp(i+6) == hex2dec('01')
        temp = temp + bin2dec('00000010');
    end
    if BitSequence_temp(i+7) == hex2dec('01')
        temp = temp + bin2dec('00000001');
    end
    ByteSequence(k) = temp;
    fprintf('%c', dec2hex(ByteSequence(k)));
    fprintf(' ');
    k = k+1;
end
fprintf('\n');
fprintf('========================================\n');
%k = k - 1;
FCS =
13115;%typecast(int16(13115),'uint16');%CRC_CCITT(ByteSequence,
k-2);
Checksum(1) = ByteSequence(k-2);
Checksum(2) = ByteSequence(k-1);

if Checksum(1) ~= bitand(bitshift(FCS,-8),hex2dec('FF'))
  fprintf('Error in Checksum 1 : ');
  fprintf('\n');
end

ERROR = 'tidak';
if Checksum(2) ~= (bitand(FCS,hex2dec('FF')))
  fprintf('Error in Checksum 2 : ');

 ERROR='ya';
  fprintf('\n');
else
  fprintf('Correct checksum 2\n ');
end
```

```
 %Convert form LSB to MSB
 %for (int i=0; i < byte*length ; i++) Byte*Sequence_temp[i] = 0x00;
 for i=1: i : byte*length
    Byte*Sequence_temp(i) = 0;
 end
 %for (int i=0; i < k-2 ; i++) Byte*Sequence_temp[i] =
MSB_LSB_swap_8bit(Byte*Sequence[i]);
 for i=1: 1 : k-3
    Byte*Sequence_temp(i) = MSB_LSB_swap_8bit(Byte*Sequence(i));
    fprintf(num2str(Byte*Sequence_temp(i)));
    fprintf('   ');
 end
 fprintf('\n');
 %fprintf('Byte*Sequence_temp : ');
 %fprintf(num2str(Byte*Sequence_temp));
 fprintf('\n');


 cnt = 1;
 %Recover header
 %for (int i=0; i < 6; i++) DestCS[i] =
char(Byte*Sequence_temp[cnt++]>>1);
 for i =1:1:6
   DestCS(i)=  bitshift(Byte*Sequence_temp(cnt),-1);
    %fprintf('CNT : ');
    %fprintf(num2str(cnt));
    %fprintf(' ');
    cnt =cnt+1;
 end
 %fprintf('\n');
 %fprintf('CNT after : ');
 %fprintf(num2str(cnt));
 %fprintf('\n');
 %SSID Destination
 cnt = cnt +1;
 %Append Source Callsign
```

```
 %for (int i=0; i < 6; i++) SourceCS[i] =
char(ByteSequence_temp[cnt++]>>1);
 for i=1:1:6
   SourceCS(i)=  bitshift(ByteSequence_temp(cnt),-1);
   cnt =cnt+1;
 end
 %Append SSID Source
 cnt =cnt +1;
 %Append Control bits
 cnt =cnt +1;
 %Append Protokol Identifier
 cnt =cnt +1;
 %Recover message
 s = k-2-cnt;
 fprintf('Final decoded Message\n');

 %for (int i=0; i < s; i++)
 %  {
 %    Message[i] = char(ByteSequence_temp[cnt++]);
 %    Serial.print(Message[i]);
 %  }
 %  Serial.println("");
 cnt = cnt +1;
%MESSG=' ';
%OUT_MESSG = (1:255);
 for i=1:1:450+256+256+256+256+256+256+256
      %OUT_MESSG(i) = ' ';
 end
 for i = 1:1:s-1
    %fprintf(char(ByteSequence_temp(cnt)));
      %MESSG = strconcat( MESSG, char(ByteSequence_temp(cnt)));
    OUT_MESSG(i) = char(ByteSequence_temp(cnt+1));
    cnt=cnt+1;
    %fprintf(' ');
 end

  fprintf('\n');
```

BIT PROCESSING

```
function [RESULT,INDEX] = BIT_PROCESSING( Buffer, byte_length, finalSequence )


BitSequence  = zeros(size(byte_length*8+1));
BitSequenceStuffed = zeros(size(byte_length*8+byte_length*8/5+1));
for i=1:1:(byte_length*8+1)
   BitSequence(i) = 0;
end
for i=1:1:(byte_length*8+byte_length*8/5+1)
   BitSequenceStuffed(i) = 0;
end

k = 1; %general counter
size_ = 1;
%s = 0; %stuffed sequence counter
cnt = 0 ;%Bit stuff counter
remBits = 0;
%temp = 0;
byte_temp=zeros(size(255*8));%max message lenght 255 bytes
for i=1:1:(255*8)
   byte_temp(i) = 0;
end
%Convert bits to byte size
%for (int i = 0; i< byte_length ; i++)
%{
%for (register uint8_t t=128; t>0 ; t = t/2) {
%if (Buffer[i] & t) BitSequence[k++] = 0x01;
%else BitSequence[k++] = 0x00;
%}
%}

%k =1;
arr_nums = [128,64,32,16,8,4,2,1];
for i=1:1:byte_length
  for x=1:1:8
   if bitand(Buffer(i),arr_nums(x))
     BitSequence(k) = 1;%hex2dec('01');
```

```
    else
      BitSequence(k) = 0;%hex2dec('00');
    end
    k=k+1;
   end
end

fprintf('Size:');
fprintf(num2str(k-1));
fprintf('\n');
%stuff a 0 after five consecutive 1s.
%for (int i = 0; i < k ; i++)
%{
%if (BitSequence[i] == 0x01) cnt++;
%else cnt = 0; // restart count at 1
%BitSequenceStuffed[s++] = BitSequence[i]; // add the bit to the final
sequence
%if (cnt == 5) // there are five consecutive bits of the same value
%{
%BitSequenceStuffed[s++] = 0x00; // stuff with a zero bit
%cnt = 0; // and reset cnt to zero
%}
%}
s = 1;
c = 1;
for i =1:1:k-1
    if BitSequence(i) == 1   %hex2dec('01')
        cnt = cnt+1;
    else
        cnt = 0;
    end
    BitSequenceStuffed(s) = BitSequence(i);
    %fprintf(num2str(BitSequenceStuffed(s)));
    %fprintf('  ');
    s = s+1;
    if cnt == 5
      BitSequenceStuffed(s) = 0;%hex2dec('00');
      %fprintf(num2str(BitSequenceStuffed(s)));
      %fprintf('  ');
```

```
      s=s+1;
      cnt = 0;
    end

end
for i=1:1:64
    Buffer(size_)= 0;%hex2dec('00');
    size_ = size_ + 1;
    for j = 1:1:6
        Buffer(size_)=1;%hex2dec('01');
        size_ = size_ + 1;
    end
    Buffer(size_)=0;%hex2dec('00');
    size_ = size_ + 1;
end
max= s-1;
%for (int i=0; i < s ; i++) Buffer[_size++] = BitSequenceStuffed[i];
 for i = 1:1:max;
     Buffer(size_) = BitSequenceStuffed(i);
     size_ = size_+1;
 end

 %Insert 0b01111110 (FLAG)
 %Buffer[_size++] = 0x00;
 Buffer(size_) = 0;%hex2dec('00');
 size_ = size_+1;
 %for (int j=0; j < 6 ; j++)
 %      {
 %        Buffer[_size++] = 0x01;
 %      }
 %      Buffer[_size++] = 0x00;
 %
 for i=1:1:6
   Buffer(size_) = 1;%hex2dec('01');
   size_ = size_+1;
 end
 Buffer(size_) = 0;%hex2dec('00');
 size_ = size_+1;
```

```
for i =1:1:((255*8))
   byte_temp(i) = 0;%hex2dec('00');
end

%for (int i = 0; i< 255*8 ; i++) byte_temp[i] = 0x00;
%NRZI encoding
%for (int i=0; i < _size ; i++)
%{
%if (Buffer[i] == 0x00)
%{
%byte_temp[i+1] = ! byte_temp[i];
%}
%else
%{
%byte_temp[i+1] = byte_temp[i];
%}
%}
for i=1:1:size_-1
    if Buffer(i)== 0
       byte_temp(i+1) = ~(byte_temp(i));
    else
       byte_temp(i+1) = (byte_temp(i));
    end
    fprintf(num2str(byte_temp(i)));
    fprintf('  ');
end
fprintf('\n');

%extrabits = (_size+1) % 8;
%if (((_size+1) % 8) > 0) remBits = 8 - ((_size+1) % 8);

if mod(size_,8) > 0
   remBits = 8 - (mod(size_,8));
end
fprintf('Rembits:');
fprintf(num2str(remBits));
fprintf('\n');
%for (int i = (_size + 1) ; i < (_size + 1 + remBits ) ; i++)
%{
```

```
%byte_temp[i] = 0x01;
%}

for i=size_+1:1:size_+ 1 + remBits
    byte_temp(i) = 1;% hex2dec('01');

end

%Convert to bit after NRZI and added remaining bits to form byte
array
Index = 1;
%fprintf('BYTE TEMP:');
%fprintf(num2str(byte_temp));
fprintf('\n');
fprintf('Final sequence:\n');
for i = 1:8:(size_+ remBits)
    %fprintf(num2str(i));
    %fprintf(' ');
    temp = 0;
    if byte_temp(i) == 1 %hex2dec('01')
        temp = temp + bin2dec('10000000');
    end
    if byte_temp(i+1) == 1 %hex2dec('01')
        temp = temp + bin2dec('01000000');
    end
    if byte_temp(i+2) == hex2dec('01')
        temp = temp + bin2dec('00100000');
    end
    if byte_temp(i+3) == hex2dec('01')
        temp = temp + bin2dec('00010000');
    end
    if byte_temp(i+4) == hex2dec('01')
        temp = temp + bin2dec('00001000');
    end
    if byte_temp(i+5) == hex2dec('01')
        temp = temp + bin2dec('00000100');
    end
    if byte_temp(i+6) == hex2dec('01')
        temp = temp + bin2dec('00000010');
```

```
    end
    if byte_temp(i+7) == hex2dec('01')
        temp = temp + bin2dec('00000001');
    end
    finalSequence(Index) = temp;
    %fprintf(num2str(Index));
    %fprintf('. ');
    fprintf(num2str(finalSequence(Index)));
    fprintf('  ');
    Index = Index+1;
end
fprintf('\n');
%fprintf(num2str(finalSequence));
%fprintf('\n');
fprintf('==================================\n');
%fprintf('byte_temp : ');
%fprintf(num2str(byte_temp));
%fprintf('\n');
RESULT = finalSequence;
INDEX = Index-1;
```

MSB_TO_LSB_8BIT

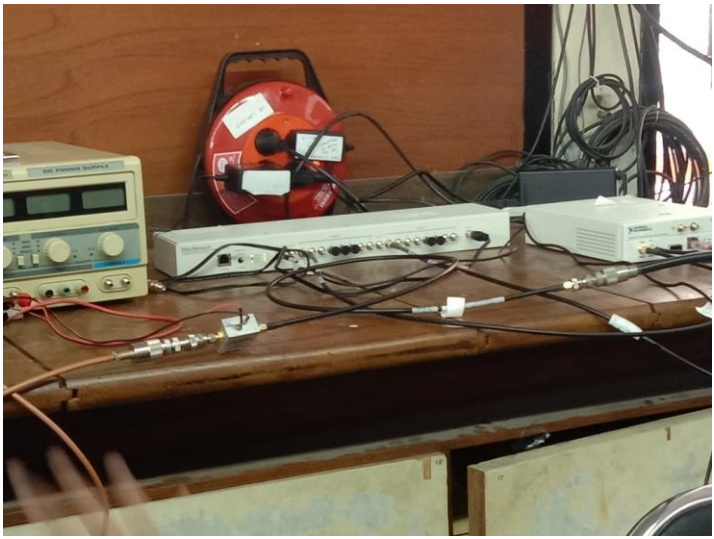
```
function y = MSB_LSB_swap_8bit(v)
  m ='55';%'0x55';
  n  = '33';
  j   ='0F';
  %swap odd and even bits
  %v = ((v >> 1) & 0x55) | ((v & 0x55) << 1);
  t = bitshift(v,-1);
  t = bitand(t,hex2dec(m));
  s = bitand(v, hex2dec(m));
  s = bitshift(s,1);
  v = bitor(t,s);
  %swap consecutive pairs
  %v = ((v >> 2) & 0x33) | ((v & 0x33) << 2);
  t = bitshift(v,-2);
  t = bitand(t,hex2dec(n));
  s = bitand(v, hex2dec(n));
  s = bitshift(s,2);
```

```
  v = bitor(t,s);
  %swap nibbles ...
  %v = ((v >> 4) & 0x0F) | ((v & 0x0F) << 4);
  t = bitshift(v,-4);
  t = bitand(t,hex2dec(j));
  s = bitand(v, hex2dec(j));
  s = bitshift(s,4);
  y = bitor(t,s);
end
```

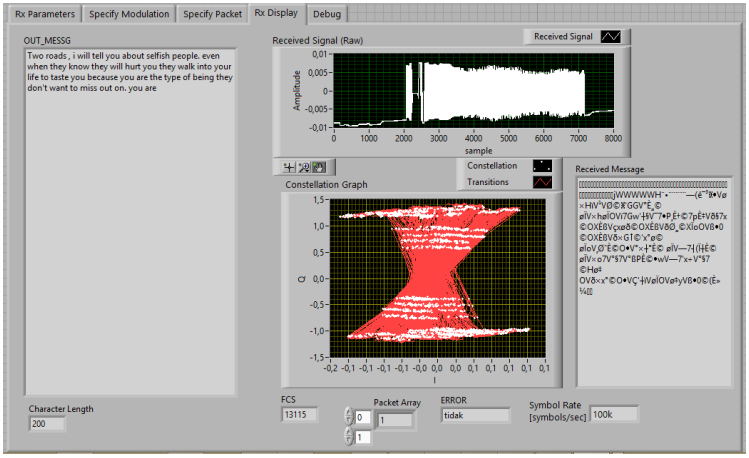
LAMPIRAN C
FOTO PENGUKURAN

Gambar antena dipole yang akan dipasang

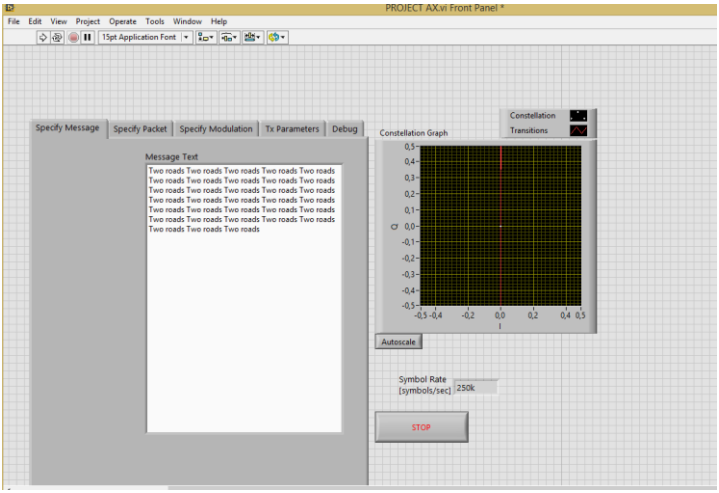
Gambar rangkaian perangkat pada *transmitter*



Gambar rangkaian perangkat pada *receiver*



Gambar tampilan keseluruhan dalam *receiver*



Gambar tampilan keseluruhan dalam *tranmitter*

[Halaman ini sengaja dikosongkan]

LAMPIRAN D
MONITORING KEGIATAN TUGAS AKHIR

KEMENTERIAN RISET, TEKNOLOGI, DAN PENDIDIKAN TINGGI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
FAKULTAS TEKNOLOGI ELEKTRO
DEPARTEMEN TEKNIK ELEKTRO
Gedung A,J , B & C, Kampus ITS Sukolilo, Surabaya 60111
Telp. (031) 5947302, 5935525, 5994251-55 (Ext.1206, 1239) Fax. (031) 5931237
Email: elits@ee.its.ac.id ; www.ee.its.ac.id

MONITORING KEGIATAN TUGAS AKHIR

Nama Mahasiswa : Vina Amalia Fitrianingrum Nrp 07111745000022
Judul Tugas Akhir : Perencanaan Protokol Data-*Link Layer* Untuk Sistem Komunikasi Radio *High Frequency* NVIS Berbasis USRP

Bulan Proposal Disahkan : Februari 2019
Dosen Pembimbing 1 : Prof. Ir. Gamantyo Hendrantoro, M.Eng., Ph.D NIP197011111993031002
Dosen Pembimbing 2 : Dr. Ir. Achmad Mauludiyanto, MT. NIP196109031989031001

No	Tanggal	Uraian Kegiatan	Tanda Tangan			Keterangan
			Pembimbing (1)	Pembimbing (2)	Mahasiswa	
1	18/02/2019	Pemaparan packet & frame				
2	25/02/2019	Laporan tentang paper yang didapat				
3	14/03/2019	AK-25 program dengan bahasa C Skema percobaan AK25 program				
4	21/03/2019	Presentasi AK-25 program				
5	28/03/2019	Pemaparan pengujian dari TA sebelum & perkembangan program				
6	4/04/2019	Program AK 25 karakter yang digunakan.				
7	11/04/2019	Penempatan AK-25 di sistem				
8	13/04/2019	Pemasthan skema percobaan AK-25				
9	18/04/2019	Progress karakter TxRx				
10	25/04/2019	Uji coba AK-25				
11	9/05/2019	Progress AK 25 dengan noise generator & pemasthan uji coba				
12	16/05/2019	Koreksi buku TA tentang perbandingan protokol.				
13	23/05/2019	Koreksi buku TA bab 4				
14	26/05/2019	Koreksi buku TA				
15	27/05/2019	Penyampaian hasil akhir				
16						

[Halaman ini sengaja dikosongkan]

RIWAYAT HIDUP

Vina Amalia F, lahir di Surabaya pada tanggal 01 Maret 1995. Telah menempuh pendidikan formal di SDI Al-Azhar 14 Semarang, SMP Islam Al Azhar 14 Semarang dan SMAN 4 Semarang, penulis melanjutkan melanjutkan ke studi Diploma 3 jurusan Teknik Elektro bidang studi Telekomunikasi di Politeknik Negeri Semarang dan lulus tahun 2016. Kemudian melanjutkan kuliah Lintas Jalur Sarjana di Institut Teknologi Sepuluh Nopember Surabaya dengan mengambil Jurusan Teknik Elektro, Bidang Studi Telekomunikasi Multimedia.

Kontak: vinaamaliaf1111@gmail.com