# IMPLEMENTASI PURWARUPA VANET DENGAN MIKROKOMPUTER MEMANFAATKAN NRF24L01

**SKRIPSI**

**KEMINATAN SISTEM KOMPUTER**

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Muhammad Wildan
NIM: 125150300111033


PROGRAM STUDI TEKNIK INFORMATIKA
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2016

# PENGESAHAN

IMPLEMENTASI PURWARUPA VANET DENGAN MIKROKOMPUTER MEMANFAATKAN NRF24L01

SKRIPSI

KEMINATAN TEKNIK KOMPUTER

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Muhammad Wildan
NIM: 125150300111033

Skripsi ini telah diuji dan dinyatakan lulus pada
11 Agustus 2016
Telah diperiksa dan disetujui oleh:

| Dosen Pembimbing I | Dosen Pembimbing II |
|---|---|
| Sabriansyah Rizqika Akbar, S.T, M.Eng | Adhitya Bhawiyuga, S.Kom, M.S |
| NIK: 19820809 201212 1 004 | NIK: 201405 890720 1 001 |

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T., M.T., Ph.D.
NIP: 19710518 200312 1 001

# PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata di dalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 16 Agustus 2016

Muhammad Wildan

NIM: 125150300111033

# KATA PENGANTAR

Puji syukur kehadirat Allah SWT yang telah melimpahkan rahmat, taufik dan hidayah-Nya sehingga laporan skripsi yang berjudul "Implementasi Purwarupa Vanet Dengan Mikrokomputer Memanfaatkan nRF24L01" ini dapat terselesaikan.

Penulis menyadari bahwa penyusunan skripsi ini tidak terlepas dari bantuan berbagai pihak. Oleh karena itu, penulis ingin menyampaikan rasa hormat dan terima kasih kepada:

1. Kedua orang tua penulis, Bapak Joko Kristioso dan Ibu Nihajah yang selalu memberikan doa, motivasi, kasih sayang, dukungan moril dan materil sebagai penyemangat dalam menyelesaikan skripsi ini.
2. Bapak Sabriansyah Rizqika Akbar, S.T, M.Eng dan Bapak Adhitya Bhawiyuga, S.Kom, M.S selaku dosen pembimbing skripsi yang telah dengan sabar membimbing dan mengarahkan penulis sehingga dapat menyelesaikan skripsi ini.
3. Bapak Issa Arwani, S.Kom., M.Sc. selaku ketua Program Studi Teknik Informatika Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya.
4. Bapak Wijaya Kurniawan, ST., MT. selaku dosen pembimbing dan penasihat akademik penulis.
5. Segenap Bapak dan Ibu dosen Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya atas segenap ilmu pengetahuan dan perhatian yang diberikan.
6. Kakak tercinta Havida Rahmah yang selalu memberi semangat dan support selama pengerjaan skripsi.
7. Saudara seperjuangan "Kelinci Corp", Rizky, Fajar, Apry, Muhlis, Pipit, Helmi, Icol, Putra, Andrika, Panji, Wildan, Ekky, Oddy, Gatut, Bela. Terimakasih atas segala bantuan dan perjuangan kita selama ini.
8. Teman topik skripsi yang sama Rizky Eka Putra yang selalu setia membantu dengan laporan paper dan ilmunya.
9. Partner duet sejak sekolah Vio Dhany Pramaysa yang selalu setia menghibur dikala sedih melanda dengan berbagai cara.
10. Teman sejak smp Putri Lenggogeni yang selalu mendengarkan curhatan dan kegalauanku selama ini, Yossinda dan Nur Syafira yang datang ketika seminar hasil.
11. Anita Dwi Puspitasari yang selalu mau mendengarkan kegelisahan dalam bimbingan serta sedikit membantu dan banyak mengganggu, dan tidak lupa anggota geng "pengajian" lainnya, Desy, Lili, Lariza, Istania, Brenda yang meskipun tidak membantu tetap bisa sedikit menghibur.
12. Seluruh teman-teman SISKOM angkatan 2012 atas dukungan dan kebersamaan dari awal perkuliahan sampai akhir kelulusan.

13. Semua pihak yang telah membantu terselesaikannya skripsi ini yang tidak dapat penulis sebutkan satu per satu.

Penulis menyadari bahwa dalam penyusunan skripsi ini masih banyak kekurangan, sehingga saran dan kritik yang membangun sangat penulis harapkan. Akhir kata penulis berharap skripsi ini dapat membawa manfaat bagi semua pihak yang menggunakannya.

Malang, 16 Agustus 2016

Penulis

Muh.wildan72@gmail.com

# ABSTRAK

Abstrak - Seiring dengan meningkatnya resiko berkendara yang semakin tinggi dibutuhkan sebuah sistem yang mampu membuat komunikasi antar kendaraan. Dengan mengimplementasikan Vanet yang memanfaatkan NRF24L01 dilakukan penelitian untuk merancang sistem yang dapat melakukan komunikasi tidak hanya antar kendaraan namun juga komunikasi antara kendaraan dengan infrastruktur sekitar. 2 *node* akan digunakan pada penelitian ini adalah RSU (Road Side Unit) yang diletakkan pada infrastruktur sekitar dan OBU (On Board Unit) yang diletakkan pada kendaraan, RSU memiliki peran sebagai sumber informasi yang akan mengirimkan informasi terkait dengan lalu lintas dan *node* OBU akan berperan sebagai penerima data, selain itu *node* OBU juga mampu mengirimkan pesan tanda bahaya. Kedua *node* tersebut melakukan komunikasi memanfaat nRF24L01 pengiriman datanya menggunakan radio frekuensi 2.4 GHz atau sama seperti Wi-Fi, pengalamatan yang dilakukan tidak menggunakan IP tapi menggunakan *pipe address*. Hasil yang didapatkan dalam penelitian ini sistem yang dibuat dapat berjalan cukup baik hingga jarak antar *node* 50 m setelah dilakukan pengujian dengan jarak dan kecepatan yang berbeda-beda. Penelitian ini belum bisa digunakan dengan jarak lebih dari 50 m karena keterbatasan jangkauan dari nRF24L01, jika ingin meningkatkan kemampuan sistem dapat dilakukan penambahan antena pada modul nRF24L01 untuk membuat pengiriman lebih jauh dan lebih cepat.

Kata kunci: VANET, nRF24L01, On Board Unit, Road Side Unit, lalu lintas, kendaraan.

# ABSTRACT

Along with high risk of driving, therefore needed system that can vehicle communicate with each other. Implement VANET using nRF24L01 is done research to devise system that isn't only communicate among vehicle but also can communicate with infrastructure. 2 *node* will be used in this research is RSU (Road Side Unit) which will be placed on infrastructure and OBU (On Board Unit) which will be placed on vehicle, RSU will play as information source who send traffic information and OBU will play as receiver, other than that OBU also can send a emergency message when in danger. Both *node* will communicate using nRF24L01 which data transmission use 2.4 radio frequency like WiFi, IP address is not used in this addressing, this research use *pipe address* as addressing for both *node*. Research result is system can work properly up to 50m gap between OBU and RSU after testing in difference distances. This can not be used more than 50m because range limit of nRF24L01, antenna is required if it want to improve system ability in order to faster and further delivery.

Keywords: VANET, nRF24L01, On Board Unit, Road Side Unit, Traffic, Vehicle.

# DAFTAR ISI

# DAFTAR TABEL

# DAFTAR GAMBAR

# BAB 1 PENDAHULUAN

## 1.1 Latar belakang

Dengan semakin majunya teknologi kendaraan menjadi sangat dibutuhkan oleh manusia, hal ini berakibat pada resiko berkendara yang semakin akan semakin tinggi seperti terus meningkatnya jumlah kemacetan dan kecelakaan yang terjadi pada lalu lintas. Bahkan juga *World Helath Orgaanization* (WHO) sempat melansir bahwa setalah tahun 2000 kecelakaan lalu lintas merupakan penyebab terbesar kematian hampir di seluruh dunia, setidaknya ada 17 kasus pada setiap 100.000 jiwa di dunia atau rata-rata ada 6.734 kasus yang ada di seluruh dunia(WHO, 2013). Banyak faktor yang menyebabkan kecelakaan bisa terjadi mulai dari kurang berhati-hatinya pengendara sampai karena pengendara yang ugal-ugalan, dan masih banyak faktor lainnya yang menyebabkan kecelakaan bisa terjadi. Maka dari itu dibutuhkan sistem yang bisa menanggulangi atau setidaknya meminimalisir terjadinya kecelakaan, hal ini bisa dilakukan apabila pengendara bisa mengetahui bagaimana keadaan sekitarnya sehingga pengendara bisa lebih berhati-hati dan dapat menghindar seandainya sudah dalam keadaan mendesak.

Agar pengendara mampu mengetahui keadaan di sekitarnya maka dibutuhkan komunikasi yang terjadi antar kendaraan yang berada di sektiar. *Vehicular Ad-hoc Network* atau VANET merupakan model jaringan yang cocok untuk diterapkan pada lalu lintas yang mampu menghubungkan *Road Side Unit* (RSU) dengan kendaraan sekitar sehingga mampu saling bertukar informasi. Dengan model jaringan VANET seperti ini informasi yang didapatkan oleh pengendara dapat digunakan untuk meningkatkan keamanan dalam berkendara.

VANET adalah salah satu aplikasi yang berkembang dengan dasar *Mobile Ad-hoc Network* (MANET). Sistem pada MANET adalah menghubungkan *mobile device* tanpa menggunakan kabel sehingga *mobile device* bisa bergerak secara bebas namun tetap terhubung dengan *mobile device* yang lain tanpa bergantung pada router. VANET memiliki sistem yang hampir sama dengan MANET hanya saja *node* yang digunakan bukan *mobile device* melainkan kendaraan yang berada di lalu lintas dan infrastruktur yang ada di sekitar lalu lintas. VANET memiliki 2 jenis *node* yaitu Road Side Unit (RSU) dan On-Borad Unit (OBU), RSU merupakan *node* yang berada di infrastruktur sekitar lalu lintas, OBU merupakan *node* yang terpasang di kendaraan yang berada di lalu lintas.

Dalam penerapannya VANET dengan menggunakan 802.11 memiliki satu masalah krusial yaitu konsumsi daya yang tinggi sehingga perlu dilakukan penghematan energi agar setiap host bekerja secara efektif. 802.11 sering mengonsumsi energi yang tidak perlu karena faktor *overhearing* (bhardwaj, 2012), atau terlalu banyak menerima data yang seharusnya tidak ditujukan kepada *node* tersebut akibat pengiriman secara *broadcast* (Basu, 2014) sehingga konsumsi daya yang dengan menggunakan 802.11 mencapai 16W per *node*nya (Bhardwaj, 2012) sehingga dibutuhkan *device* yang bisa memanfaatkan daya seefisien mungkin, dan *device* yang sesuai dengan kriteria tersebut adalah nRF24L01 yang memiliki

konsumsi daya hanya 5V untuk setiap *node* nya, tetapi tetap mampu melakukan komunikasi antar *node*. NRF24L01 merupakan modul komunikasi jarak jauh yang menggunakan *single-chip RF-transceiver* pada gelombang 2.4 GHz ISM band (Nordic, 2007).

Penelitian ini dilakukan dengan cara melakukan percobaan pengiriman data antara RSU dengan OBU dan antara OBU dengan OBU. Data yang dikirimkan disesuaikan dengan aplikasi yang diterapkan, yang mana data yang dikirimkan merupakan data yang berkaitan dengan keadaan lalu lintas dan pesan bahaya untuk memberikan peringatan apabila terjadi keadaan yang darurat pada lalu lintas.

## 1.2 Rumusan masalah

1) Bagaimana perancangan VANET menggunakan NRF24L01?
2) Bagaimana cara mengirimkan dan menerima data dengan NRF24L01?
3) Bagaimana pengaruh jarak antar kendaraan agar terhadap komunikasi data?
4) Bagaimana pengaruh kecepatan kendaraan terhadap komunikasi data?

## 1.3 Tujuan

1) Mengetahui cara VANET melakukan komunikasi dengan mikrokomputer menggunakan NRF24L01
2) Mengetahui jarak antar kendaraan agar mampu melakukan komunikasi
3) Mengetahui kecepatan maksimal agar kendaraan mampu melakukan komunikasi

## 1.4 Manfaat

Manfaat dari penelitian ini diharapkan dapat diterapkan pada sistem lalu lintas demi meningkatkan kewaspadaan dalam berkendara di jalanan, selain itu menjadi dasar dari pengembangan penelitian *vehicular network* dalam pengolahan informasi ke tahap berikutnya

## 1.5 Batasan masalah

1) Komunikasi dilakukan oleh 2 jenis *node* saja yaitu RSU (*Road Side Unit)* dan OBU(*On-Board Unit).*
2) Kecepatan kendaraan maksimal adalah 30km/jam
3) Jarak antar *node* adalah 10m sampai 50m

## 1.6 Sistematika pembahasan

**BAB I PENDAHULUAN**

Mengeruakan latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat dan sistematika penulisan.

**BAB II LANDASAN KEPUSTAKAAN**

Menguraikan tinjauan pustaka dan dasar teori yang mendasari teknologi VANet, NRF24l01, dan beberapa aspek yang digunakan dalam penelitian.

**BAB III METODOLOGI**

Menguraikan tentang metode dan langkah kerja yang terdiri dari studi literature, analisis kebutuhan simulasi, perancangan sistem, implementasi dan analisis serta pengambilan kesimpulan.

**BAB IV REKAYASA PERSYARATAN**

Menguraikan tentang deskripsi umum yang berisi perspektif sistem, kegunaan dan karakteristik pengguna. Menguraikan juga tentang kebutuhan, kebutuhan fungsional dan kebutuhan non fungsional.

**BAB V PERANCANGAN DAN IMPLEMENTASI**

Menguraikan proses implementasi dari dasar teori yang telah dipelajari sesuai analisis dan perancangan sistem.

**BAB VI PENGUJIAN DAN ANALISIS**

Memuat hasil pengujian dan analisis terhadap sistem yang telah direalisasikan.

**BAB VII PENUTUP**

Memuat kesimpulan yang diperoleh dari pembuatan dan pengujian, serta saran-saran untuk pengembangan lebih lanjut.

# BAB 2 LANDASAN KEPUSTAKAAN

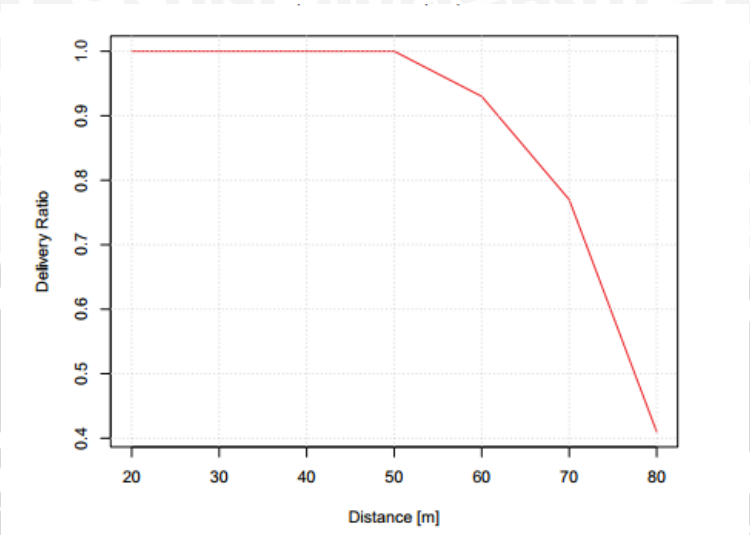
Bab ini berisi uraian serta pembahasan teori, masalah, konsep, metode yang berkaitan dengan penelitian yang akan dilakukan. Dalam bab ini terdapat landasan teori dari berbagai sumber yang terkait dengan teori dan metode yang akan digunakan.

## 2.1 Tinjauan Pustaka

Pada Paper "*A comprehensive survey on vehicular Ad Hoc network*" milik Saif Al-Sultan menjelaskan banyak hal tentang VANET seperti apa itu VANET, arsitektur VANET seperti apa, bagaimana melakukan komunikasi pada VANET, media apa saja yang dapat dimanfaatkan pada VANET, tantangan yang ada pada VANET, dan aplikasi apa saja yang mampu diterapkan oleh VANET. Saif menjelaskan arsitektur VANET terdiri dari banyak komponen seperti *Road-side Unit*(RSU) yang merupakan *node* yang berada pada infrastruktur lalu lintas dan memiliki sifat statis atau tidak berpindah, *On-Board Unit*(OBU) merupakan *node* yang berada pada kendaraan dan *node* yang melakukan komunikasi dengan RSU, dan *Application Unit*(AU) merupakan penyedia layanan untuk OBU dan RSU agar dapat *connect* ke dalam sebuah jaringan. Pada Paper ini juga dijelaskan banyak media yang bisa digunakan dalam mengimplementasikan VANET seperti jaringan *cellular* atau jaringan *Mobile* yang sudah diterapkan dalam *Mobile device* dan MANET, karena VANET sendiri merupakan salah satu perkembangan dari MANET maka dari itu VANET bisa menggunakan media jaringan *cellular*, VANET juga bisa memanfaatkan *Wi-Fi* dalam komunikasi antar kendaraan atau komunikasi dengan infrastruktur sekitar, dan masih banyak media yang bisa digunakan seperti *WiMAX* dan DSRC/WAVE. Dalam papernya Saif juga membahas bahwa VANET sendiri masih memiliki banyak tantangan seperti keamanan jaringan, *bandwidth*, sampai catu daya yang dibutuhkan, mungkin catu daya memang bukan masalah yang terlalu serius namun jika *node* menggunakan daya yang tinggi nanti juga akan timbul masalah.

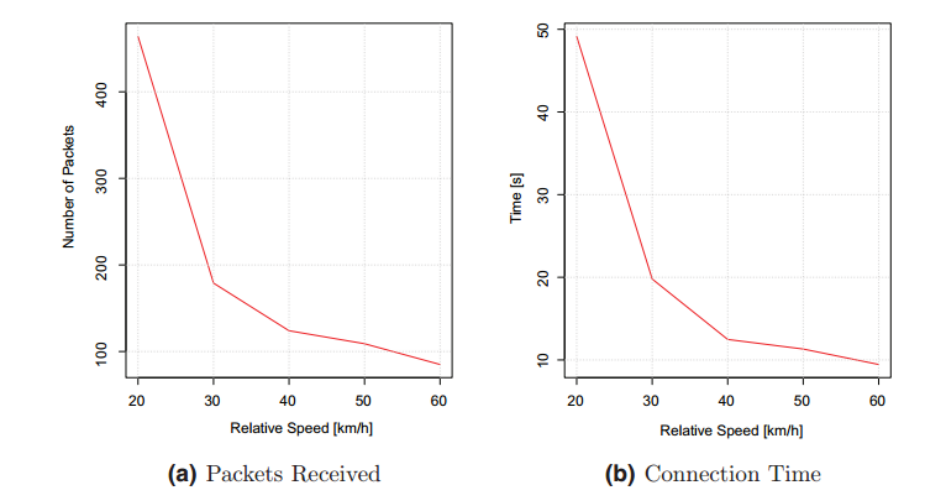
Liming Zheng melakukan penelitian terhadap implementasi VANET yang ditulis dalam papernya yang berjudul *Research on Communications over VANET under Different Scenes and Implementation of Vehicle Terminal.* penelitiannya tersebut membahas bagaimana implementasi VANET pada 3 jenis adegan lalu lintas yang berbeda yaitu keadaan jalan raya, jalan perkotaan, dan lalu lintas sesungguhnya menggunakan simulasi dengan mengirimkan video MPEG yang dikirimkan melalui VANET dengan protokol *routing* yang berbeda. Dari hasil pengujian yang dilakukan Zheng dapat menyimpulkan bahwa pada keadaan jalan raya apabila kendaraan berjalan terlalu cepat maka akan membuat informasi posisi kendaraan tidak akurat dan akan terjadi *packet loss* yang sangat tinggi, dan pada keadaan jalan perkotaan ketika lalu lintas sangat padat akan terjadi *collision* pada *wireless channel* yang mengakibatkan banyaknya *packet loss* yang terjadi, sementara pengujian pada lalu lintas sesungguhnya menghasilkan performa yang buruk karena jarak antar *node* yang terlalu jauh.

Pada penelitian *Bluetooth Low Energy performance and robustness analysis for Inter-Vehicular Communication* (Bronzi, et al., 2016) menggunakan BLE atau *bluetooth low energy* sebagai media komunikasi *di Inter-Vehicular Communication*, peneliti melakukan penyelidikan tentang kemampuan BLE dalam konteks *vehicular*.




**Gambar 2. 1 Rasio Pengiriman Data Pada Kendaraan Bergerak**

Gambar 2.1 merupakan hasil dari pengujian dengan kecepatan konstan yang pada jarak 20m hingga 80m, bisa dilihat bahwa 100% pengiriman bisa dilakukan hingga pada jarak 50m, namun pada jarak lebih jauh hingga jarak 80m rasio pengiriman data turun hingga 40% karena jarak yang terlalu jauh dan membutuhkan waktu untuk melakukan koneksi ulang di antara kedua *node* tersebut.




**Gambar 2. 2 Grafik Pengiriman dan Koneksi Pada Kecepatan Tertentu (Bronzi, et al., 2016)**

Pada Gambar 2.1 merupakan gambar dari hasil pengujian dengan kecepatan konstan untuk diketahui lama waktu *pairing* dan berapa paket yang berhasil dikirimkan selama *pairing*, seperti pada penjelasan dalam Paper dengan kecepatan konstan 20 Km/j di mana kendaraan mulai berjalan pada jarak 300m dari kendaraan

yang berhenti diketahui *pairing* terjadi pada jarak 110m dan berhasil *pairing* selama 50 detik sebelum terputus, paket yang berhasil dikirimkan 464 paket, pada saat kecepatan ditambah waktu terhubung menurun secara signifikan pada kecepatan 30Km/j menurun sampai 19.8 detik dan hanya mengirimkan 1796 paket. Peneliti juga menjelaskan kalau jarak maksimal terhubungnya dua *device* adalah 100m dan merupakan komunikasi yang kuat, dari penelitian ini dapat dikaji dengan komunikasi menggunakan BLE memiliki jarak maksimal yang cukup jauh sekitar 100m, dalam komunikasi antar kendaraan yang mungkin bertukar data dalam jarak yang dekat tidak dijadikan suatu permasalahan.

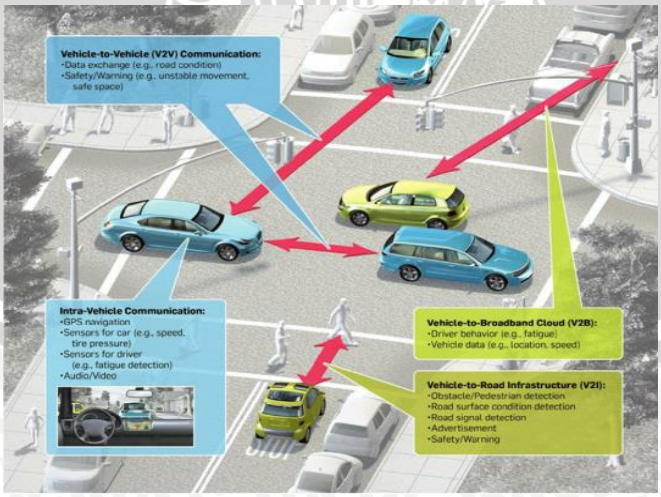
## 2.2 Dasar Teori

Pada bagian dasar teori menjelaskan tentang beberapa definisi tentang teori-teori yang berkaitan dengan sistem ini meliputi teori tentang VANET, *wireless sensor network,* NRF24L01, Arduino pro mini.

### 2.2.1 VANET

VANET merupakan salah satu aplikasi pengembangan dari MANET untuk komunikasi yang dilakukan pada kendaraan sehingga membentuk suatu *Intelligent Trnasportation System* (ITS). VANET menggunakan beberapa standart dalam pengembangannya seperti *Dedicated Short Range Communication* (DSRC) yang mana komunikasi yang dilakukan hanya dalam *coverage area* yang cukup kecil. Selain itu VANET juga menggunakan *Standard Wireless Access in Vehicular Area* (WAVE).

Dalam arsitekturnya VANET terdiri dari kendaraan-kendaraan yang sudah terinstall *On Board Unit* (OBU) dan infrastruktur yang ada atau bisa disebut *Road Side Unit* (RSU) yang digunakan untuk berkomunikasi. Biasanya di dalam OBU juga dipasang *Global Positioning System* (GPS) untuk mengirimkan informasi di mana kendaraan berada dan juga untuk mengetahui jejak dari kendaraan yang lain.




**Gambar 2. 3 Arsitektur VANET (Al *wireless* -Sultan, 2014)**

Komunikasi biasanya menggunakan strandar seperti IEEE 802.11. RSU akan berperan sebagai *router* yang memiliki *coverage area* yang luas dibandingkan

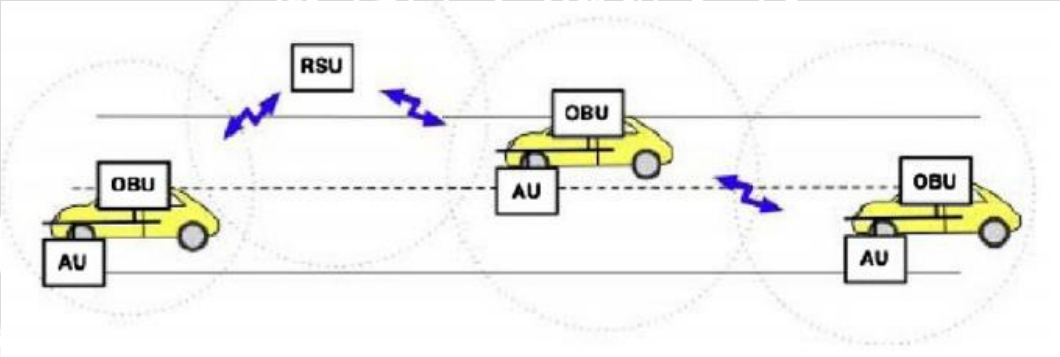
dengan OBU yang ada pada kendaraan sehingga RSU mampu berkomunikasi dengan kendaraan yang tidak masuk dalam *coverage area* kendaraan yang lain.

### 2.2.2 On Board Unit

*On board unit* (OBU) adalah *device* yang nantinya akan dipasang pada kendaraan dan digunakan untuk berkomunikasi dengan OBU yang lain atau dengan *road side unit* (RSU). OBU membuat jaringan dengan *coverage area* yang pendek berdasarkan *Standard* IEEE 802.11p. OBU melakukan koneksi dengan OBU yang lain atau dengan RSU melalui jaringan *wireless* berdasarkan IEEE 802.11p radio *frequency channel*, yang nantinya bertanggung jawab atas komunikasi yang terjalin antara OBU dengan OBU atau OBU dengan RSU(Al-Sultan, 2014).

### 2.2.3 Road Side Unit (RSU)

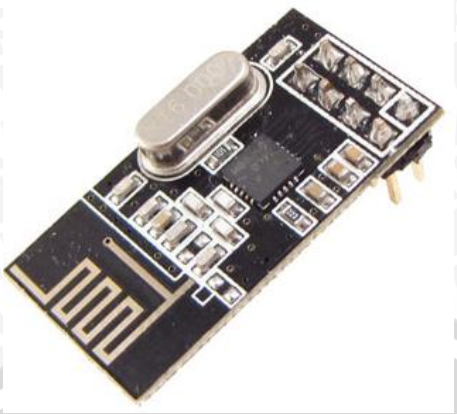
*Road side unit* (RSU) merupakan *device* yang berbeda dengan OBU yang bergerak secara *mobile*, RSU biasanya sudah terpasang di tempat tertentu seperti pada rambu jalan atau pada tempat parkir. RSU juga menggunakan *Standard* IEEE 802.11p radio *frequency channel* untuk membangun koneksi dengan OBU dan bertanggung jawab atas komunikasi data yang terjadi atau bahkan membangun koneksi antar infrastruktur. RSU juga bisa berfungsi untuk memperluas jaringan dengan cara mendistribusikan ulang data yang sudah diterima ke OBU atau ke RSU yang lain, selain itu RSU juga berperan untuk menyampaikan informasi penting seperti kecelakaan, kemacetan, dsb.




**Gambar 2. 4 Pengiriman Data RSU (Al-Sultan, 2014)**

### *2.2.4* NRF24L01

NRF24L01 adalah sebuah modul komunikasi jarak jauh hingga 100 m dengan *single-chip RF-transciver* yang ditujukan untuk aplikasi pada gelombang 2.4 GHz ISM band (Nordic, 2007). Modul NRF24L01 menggunakan antar muka SPI (*Serial Pheripheral Interface*) untuk komunikasi dengan tegangan kerja 5V DC. *Output power* dan saluran komunikasi dapat dengan mudah dikonfigurasikan dengan program. Konsumsi daya NRF24L01 rendah dan dengan berbagai mode daya rendah, NRF24L01 lebih hemat energi. (ZHU Jian-hong, 2011) Antarmuka modul NRF24L01 dapat dilihat pada gambar 2.5.



**Gambar 2. 5 modul nRF24L01 (Nordic, 2007)**

nRF24L01 sendiri merupakan salah satu *device* yang komunikasinya tidak menggunakan *IP addres*, namun menggunakan *channel*. Ada 126 radio frekuensi *channel* yang bisa digunakan untuk pengiriman data pada nRF24L01. Untuk dapat berkomunikasi, *node* yang digunakan harus berada pada *channel* yang sama. Pada implementasinya nRF24L01 memiliki 2 mode, yaitu *RX mode* yang merupakan mode di mana nRF24L01 radio yang berperan sebagai penerima data dan *TX mode* merupakan mode di mana nRF24L01 radio berfungsi untuk mengirimkan paket data.

### 2.2.5 RF24 Library

RF24 merupakan salah satu *library* yang ada pada nRF24L01, *library* RF24 sendiri merupakan *library* yang digunakan untuk melakukan komunikasi data. Untuk melakukan pengiriman data *library* RF24 dapat menggunakan 2 pengalamatan yaitu dengan menggunakan *channel* seperti yang biasa digunakan pada nRF24L01 atau bisa menggunakan *pipe address* yang penggunaannya sedikit berbeda dengan *channel. Pipe address* memiliki 2 jenis alamat yaitu *open.writingPipe* yang merupakan alamat untuk melakukan pengiriman data dan *open.readingPIpe* yang merupakan alamat untuk melakukan penerimaan data, biasanya untuk melakukan komunikasi data perlu digunakan 2 alamat *pipe address* yang digunakan untuk *open.writingPipe* dan untuk *open.readingPipe.* pengiriman data yang dilakukan oleh *library* RF24 ini juga fleksibel karena data yang dikirim ada sesuai dengan inisialisasi data sehingga tidak perlu dilakukan perubahan data menjadi *byte*.

**Tabel 2. 1 Contoh Script Pengiriman data Library RF24**

| Baris | *Source Code* |
|---|---|
| 1 | `#include <SPI.h>` |
| 2 | `#include <nRF24L01.h>` |
| 3 | `#include <RF24.h>` |
| 4 | `RF24 radio(8, 7);` |
| 5 | `const byte rxAddr[6] = "00001";` |
| 6 | `void setup(){` |

```
7    Serial.begin(9600);
8    radio.begin();
9    radio.setRetries(15, 15);
10   radio.openWritingPipe(rxAddr);
11   radio.stopListening();
12 }
13 void loop(){
14   const char text[] = "Hello World";
15   radio.write(&text, sizeof(text));
16   delay(1000);
17 }
```

Tabel 2.1 merupakan contoh script pengiriman data memanfaatkan library RF24, data yang dikirim adalah data String yang berisi “Hello World” yang dikirim melalui alamat *pipeaddress openWritingPipe* 00001. *Script* ini adalah khusus digunakan untuk mengirimkan data karena terdapat fungsi *radio.stopListening()* yang berarti *node* tidak akan menerima data apapun.

**Tabel 2. 2 Contoh Script Penerimaan data Library RF24**

| Baris | Algoritma |
|---|---|
| 1 | `#include <SPI.h>` |
| 2 | `#include <nRF24L01.h>` |
| 3 | `#include <RF24.h>` |
| 4 | `RF24 radio(8, 7);` |
| 5 | `const byte rxAddr[6] = "00001";` |
| 6 | `void setup(){` |
| 7 | `  while (!Serial);` |
| 8 | `  Serial.begin(9600);` |
| 9 | `  radio.begin();` |
| 10 | `  radio.openReadingPipe(0, rxAddr);` |
| 11 | `  radio.startListening();` |
| 12 | `}` |
| 13 | `void loop(){` |
| 14 | `  if (radio.available())  {` |
| 15 | `    char text[32] = {0};` |
| 16 | `    radio.read(&text, sizeof(text));` |
| 17 | `    Serial.println(text);  } }` |

Tabel 2.2 merupakan contoh *script* penerimaan data menggunakan *library* RF24, data yang akan diterima adalah data "*Hello World"* yang dikirim sebelumnya. Alamat yang digunakan sama seperti *script* pegiriman yaitu *pipeAddress* 00001 namun pada penerimaan data *pipe address* yang digunakan adalah *openReadingPipe*

### 2.2.6 Real-Time Clock

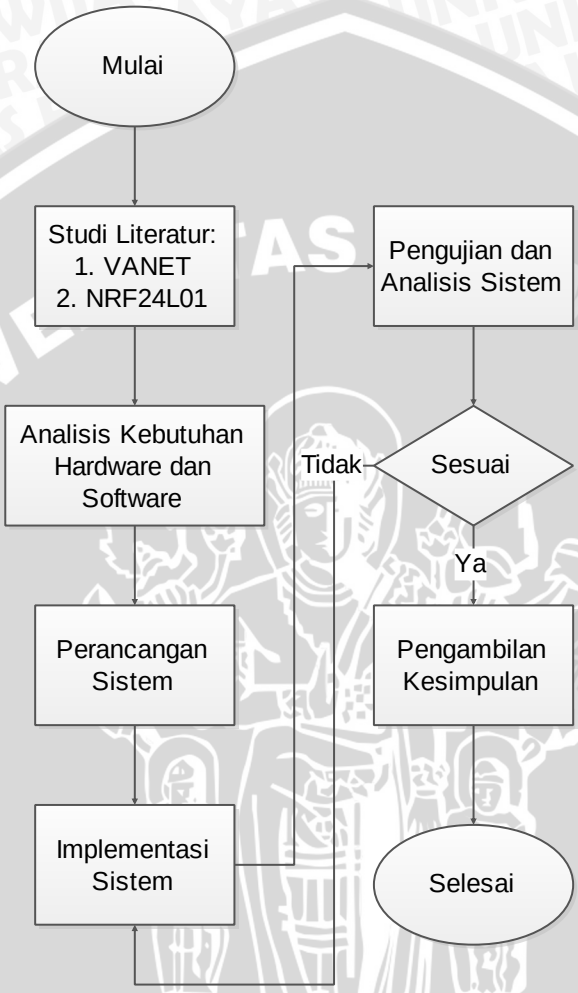
*Real-Time Clock* atau yang bisa disingkat RTC merupakan modul yang berfungsi untuk menghitung waktu dengan daya yang rendah dan memiliki 56 *bytes* NV RAM. Alamat dan data dikirim melalui 2 pin yang terhubung pada pin A4 dan A5 Arduino. Waktu yang bisa ditampilkan pada RTC adalah, detik, menit, jam, hari, tanggal, dan tahun. RTC juga bisa diatur untuk menampilkan jam secara 24 jam atau 12 jam, karena RTC hanya modul untuk menghitung waktu saja maka ketika ingin menampilkan waktu diperlukan *device* yang lain.

**Gambar 2. 6 Modul RTC (banggood.com)**

# BAB 3 METODOLOGI

## 3.1 Alur Metode Penelitian

Alur metode penelitian yang dilaksanakan secara umum dapat dilihat pada diagram alir yang seperti ditunjukkan pada gambar 3.1.




**Gambar 3. 1 Diagram Alir Metode Penelitian**

Yang dilakukan pada alur metode penelitian setelah mengetahui apa masalah yang diangkat adalah menggali informasi dari literatur-literatur yang terdahulu yang berkaitan dengan penelitian yang akan dilakukan. Selanjutnya dilakukan analisa untuk apa saja yang dibutuhkan oleh sistem agar sistem pada penelitian ini dapat berjalan. Setelah itu dilakukan perancangan sistem agar dalam implementasinya perancangan lebih mudah dilakukan. Kemudian setelah rancangan yang dilakukan sudah jadi, yang dilakukan adalah mengimplementasikan rancangan yang sudah dibuat dan memastikan sistem berjalan dengan baik. Setelah memastikan sistem dapat berjalan dengan baik, maka akan dilakukan pengujian terhadap sistem tersebut. Pada pengujian sistem ini juga akan dilihat apakah hasil dari sistem yang dibuat ini sesuai atau tidak dengan yang diharapkan. Ketika hasil masih belum sesuai

harapan maka proses implementasi akan kembali dilakukan untuk mendapatkan hasil yang diinginkan, namun ketika hasil pengujian sudah dianggap sesuai maka dapat dilakukan pengambilan kesimpulan dari hasil yang didapatkan.

## 3.2 Studi Literatur

Pada bagian ini dibahas mengenai dasar teori yang mendukung penelitian penerapan purwarupa *vehicular network* dengan menggunakan mikrokomputer memanfaatkan nRF24L01. Berikut merupakan dasar teori yang digunakan sebagai bahan studi:

1) Vehicular ad-hoc Network (VANET)

   Melakukan studi literatur terkait dengan *Vehicle ad-hoc Network* (VANET) dengan mencari paper-paper terkait yang menjelaskan bagaimana teori dan implementasi VANET dapat dilakukan. Tahapan ini mempelajari bagaimana infrastruktur dan topologi VANET bisa dibentuk.

2) NRF24L01

   Pada bagian ini akan dilakukan kajian terhadap penelitian terkait penerapan NRF24L01 pada sebuah topologi jaringan. Disini akan ditemukan bagaimana cara menerapkan NRF24L01 pada sebuah jaringan ad-hoc

## 3.3 Analisis Kebutuhan

### 3.3.1 Kebutuhan Pengguna

1) Pengguna dapat melihat data yang diterima
2) Pengguna dapat mengirim pesan darurat

### 3.3.2 Kebutuhan Sistem

1) RSU mampu mengirim data
2) OBU mampu menerima data
3) OBU mampu mengirim data.

### 3.3.3 Kebutuhan Perangkat Keras

Kebutuhan perangkat keras yang diperlukan untuk membangun sistem ini meliputi:

1) Perangkat Komputer / Laptop
2) Arduino ProMini
3) NRF24L01
4) FTDI

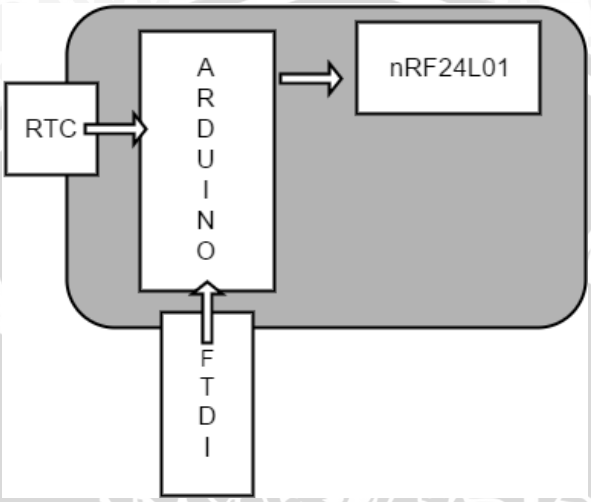
### 3.3.4 Kebutuhan Perangkat Lunak

1) Sistem Operasi Windows 10

2) Arduino IDE

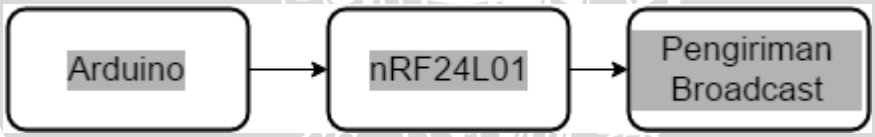
## 3.4 Perancangan Sistem

Tahap ini merupakan tahapan untuk membangun sistem dari penelitian setelah melakukan studi literatur dan analisa kebutuhan sistem. Perancangan sistem dapat dilihat pada diagram blok dibawah yang terdiri dari diagram blok *node* sistem dan rancangan kerja sistem.



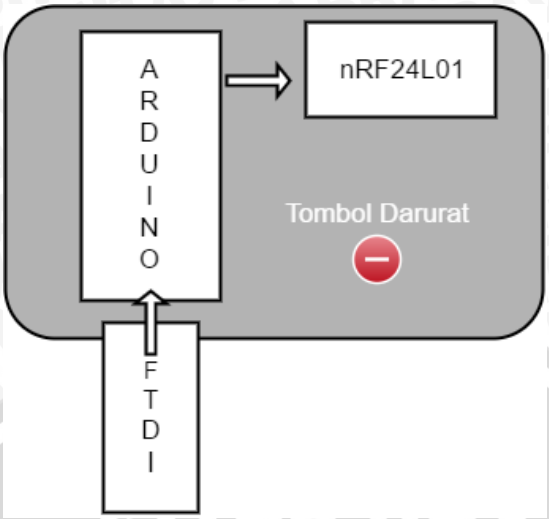

**Gambar 3. 2 diagram blok perancangan RSU**

Gambar 3.2 merupakan diagram blok perancangan RSU (*Road Side Unit)* yang nantinya akan dipasang pada infrastruktur lalu lintas. *Node* RSU dterdiri dari Arduino yang berperan sebagai mikrokontroler yang akan dibantu oleh FTDI untuk menerjemahkan algoritma, Arduino sendiri akan mendapatkan informasi waktu yang berasal dari RTC. Arduino juga terhubung dengan nRF24L01 yang bertugas melakukan pengiriman data secara broadcast.



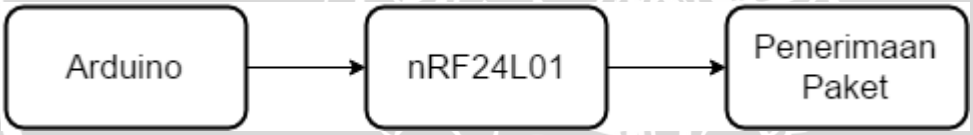

**Gambar 3. 3 Diagram Blok Kerja Sistem *Node* RSU**

Gambar 3.3 merupakan diagram blok dari perancangan kerja dari *node* RSU, *node* RSU pada sistem ini berperan sebagai sumber informasi sehingga *node* RSU hanya bisa melakukan pengiriman data. Proses pengiriman data diawali oleh arduino yang mengolah data yang nantinya akan dikirimkan. Setelah arduino selesai memproses data, lalu data tersebut dikirimkan ke nRF24L01 untuk dikirimkan secara broadcast ke *node* OBU di sekitarnya.




**Gambar 3. 4 Diagram Blok Perancangan *Node* OBU**

Gambar 3.4 adalah gambar diagram blok perancangan *node* OBU, seperti yang bisa dilihat perancangan *node* OBU tidak berbebda jauh dari perancangan *node* RSU. Pada *node* OBU tidak ada RTC untuk mengatur waktu, namun pada *node* OBU terdapat tambahan tombol darurat yang bisa digunakan saat dalam keadaan darurat. Arduino disini juga memiliki peran yang sama seperti pada *node* RSU yaitu sebagai mikrokontroler yang bertugas mengontrol keseluruhan rangkaian yang dibantu oleh FTDI untuk mendownload algoritma.




**Gambar 3. 5 Diagram Blok Penerimaan Data *Node* OBU**

Gambar 3.5 merupakan perancangan *node* OBU yang nantinya akan dipasang pada kendaraan memiliki fungsi sebagai penerima data yang dikirimkan oleh *node* RSU. Proses diawali dari arduino sebagai mikrokontroler untuk mempersiapkan untuk melakukan penerimaan data, setelah data siap maka radio dari nRF24L01 akan melakukan penerimaan paket yang dikirim oleh *node* RSU.


**Gambar 3. 6 Diagram Blok Pengiriman Data *Node* OBU**

Pada gambar 3.6 dijelaskan selain melakukan pengiriman data, *node* OBU juga mampu melakukan pengiriman darurat. pengiriman pesan darurat bisa dilakukan ketika tombol darurat ditekan oleh user, ketika tombol ditekan maka akan arduino akan secara otomatis memperisiapkan data untuk dikirim lalu akan memanfaatkan nRF24L01 untuk mengirimkan pesan darurat.

## 3.5 Implementasi Sistem

Impelemntasi sistem adalah tahap dimana sistem yang sudah dirancang sebelumnya akan diterapkan. Ada 2 jenis implementasi pada tahap implementasi sistem yaitu implementasi software dan implemtasi hardware.

1) Implementasi Software

Implementasi software yang dilakukan hanya untuk pembuatan program saja, tidak ada *GUI (Graphical User Interface)* untuk sistem yang dibuat ini. Pada pembuatan program aplikasi yang digunakan adalah Arduio IDE.

2) Implementasi Hardware

Pada implementasi *hardware*, menggunakan *minimum system* yang memanfaatkan Arduino ProMini sebagai otak dari komponen dimana Arduio ProMini ini akan dibantu oleh FTDI untuk mendownload algoritma yang nantinya akan diolah oleh Arduio ProMini itu sendiri. Hasil dari coding yang diolah oleh Arduino ProMini tadi akan diimplementasikan pada NRF24L01 yang berperan sebagai pengirim dan penerima data.

## 3.6 Pengujian Sistem

Pengujian sistem dilakukan dengan beberapa scenario yang ada untuk mendapatkan hasil yang sesuai. Scenario pengujian sistem meliputi:

1) RSU mengirim data ke OBU dengan jarak 50m, 40m, 30m, 20m, dan 10m
2) RSU mengirim data ke OBU yang bergerak mendekat dengan kecepatan 10km/jam, 20km/jam, 30km/jam
3) OBU mengirim data ke OBU pada jarak 50m, 40m, 30m, 20m, dan 10m
4) OBU mengirim data ke OBU dimana kedua OBU sama-sama bergerak mendekat dengan kecepatan 10km/jam, 20km/jam, 30km/jam

## 3.7 Pengambilan Kesimpulan

Jika semua pengujian sistem sudah dilakukan dan memiliki hasil yang sesuai, maka dapat ditarik kesimpulan dari hasil data yang dihasilkan. Kesimpulan dibuat berdasarkan hasil pengujian secara garis besar.

# BAB 4 REKAYASA PERSYARATAN

## 4.1 Pendahuluan

Pada penelitian perancangan VANET dengan menggunakan nRF24L01, telah ditentukan tujuan sistem, manfaat sistem, karakteristik pengguna, , batasan sistem, dan asumsi serta ketergantungan.

### 4.1.1 Tujuan

Sistem pada penelitian ini bekerja normal apabila setiap *node* yang ada yaitu RSU dan OBU dalam keadaan menyala dan berada dalam satu channel yang sama. RSU nantinya akan ditempatkan pada infrastruktur lalu lintas yang berada di sisi jalan, sementara OBU nantinya akan ditempatkan pada kendaraan yang melintas.

### 4.1.2 Manfaat

Penelitian ini berguna untuk menghasilkan sistem purwarupa VANET yang menggunakan NRF24L01 untuk alat komunikasinya. Selain itu sistem juga dapat diimplementasikan pada infrastruktur lalu lintas sehingga informasi yang dikirmkan bisa sesuai dengan keadaan lalu lintas pada saat itu, selain itu informasi yang dikirimkan juga bisa berkaitan dengan kendaraan yang ada sesuai dengan informasi dari kendaraan masing-masing. Hasil penelitian ini juga dapat dilanjutkan ketahap berikutnya dengan mengolah informasi yang dikirmkan ke setiap *node* sesuai dengan kebutuhan.

### 4.1.3 Karakteristik Pengguna

Pengguna dalam hal ini adalah pengendara yang sedang berada di lalu lintas. Tugas pengguna adalah mengaktifkan OBU dan mengirimkan pesan apabila terjadi keadaan darurat.

### 4.1.4 Batasan Sistem

1) Sistem diaplikasikan dalam bentuk purwarupa.
2) Purwarupa yang dimaksud tidak dalam bentuk kendaraan ataupun infrastruktur lalu lintas melainkan hanya satu set alat yang terhubung dengan PC/laptop.
3) Sistem yang dibuat hanya untuk melakukan pengiriman data yang dilakukan RSU dan penerimaan data yang dilakukan oleh OBU tanpa menggunakan sensor sebagai sumber informasinya sehingga informasi dalam paket dala adalah dummy.
4) Data yang dikirimkan tidak memiliki format baku tertentu sehingga format pengiriman data ditentukan sebelum melakukan pengiriman data.
5) Jarak maksimal antar *node* untuk melakukan pengiriman dan penerimaan data adalah 50m.

### 4.1.5 Asumsi dan Ketergantungan

1) *Baudrat*e yang digunakan oleh RSU dan OBU harus sama, agar data yang dikirim bisa dibaca oleh penerima.
2) *Pipe address* yang digunakan antara *transmitter* dan *receiver* harus sama sehingga dapat terjadi komunikasi.

## 4.2 Analisis Kebutuhan

Analisis kebutihan bertujuan untuk menggambarkan apa saja yang kebutuhan-kebutuhan dari sistem. Kebutuhan yang dimaksud terdiri dari kebutuhan fungsional dan kebutuhan non-fungsional yang akan dianalisis sesuai dengan kebutuhan sistem sehingga akan mempermudah untuk mendesain dan mengimplementasikan sistem.

### 4.2.1 Kebutuhan Fungsional

1) RSU mampu mengirimkan data secara broadcast

   Pada fungsi ini *node Road Side Unit* (RSU) dapat mengolah data yang dimilikinya dan mengirimkannya secara broadcast.

2) OBU mampu menerima data

   Pada fungsi ini *node On-Board Unit* (OBU) dapat menerima data yang sebelumnya sudah dikirm RSU secara broadcast atau dapat menerima pesan bahaya yang dikirimkan oleh OBU yang lain.

3) OBU mampu mengirimkan pesan bahaya.

   Pada fungsi ini *node* OBU dapat mengirimkan pesan bahaya apabila terjadi keadaan darurat dengan cara pengendara menekan tombol darurat yang ada pada OBU.

### 4.2.2 Deskripsi Non Fungsional

### 4.2.2.1 Kebutuhan Performa

Sistem yang dibuat pada penelitan ini akan berkerja dengan baik apabila jarak antar *node* 50m atau kurang, hal ini dikarenakan coverage area dari NRF24L01 sendiri yang tidak terlalu besar. Dan juga dengan keadaan jalan yang tidak memiliki banyak halangan seperti gedung.

### 4.2.3 Kebutuhan Perangkat Keras

Kebutuhan perangkat keras pada sistem ini antara lain:

1) Komputer/Laptop

   Komputer/laptop yang ada akan berguna sebagai catu daya dari Arduino Pro Mini, selain itu kompter juga berguna untuk melihat data yang dikirim atau diterima melalui *serial monitor.*

2) Arduino Pro Mini

Arduino Pro Mini berguna untuk menerapkan algoritma yang ada sehingga nRF24L01 mampu melakukan komunikasi.

3) FTDI FT232RL

FTDI berguna sebagai jembatan atara computer/laptop dengan Arduino, FTDI sendiri memiliki peran sebagai penerjemah algoritma agar dapat diterapkan oleh Arduino

4) NRF24L01

nRF24L01 berfungsi sebagai Modul pengiriman dan penerimaan data.

5) RTC

RTC berfungsi sebagai penunjuk waktu yang ditempatkan pada *node RSU.*

### 4.2.4 Kebutuhan Perangkat Lunak

Kebutuhan perangkat lunak pada sistem ini antara lain:

1) Sistem operasi Windows 10

Sebagai *platform* dalam menjalankan sistem

2) Arduino IDE

Sebagai tempat membangun *source code* pengiriman dan penerimaan data *node* RSU dan *node* OBU.

3) Library RF24

Sebagai *library* utama untuk membangun *source code* pengiriman dan penerimaan data *node* RSU dan *node* OBU.

# BAB 5 PERANCANGAN DAN IMPLEMENTASI

## 5.1 Perancangan Sistem

Pada bagian ini akan dijelaskan lebih mendetail tentang perancangan sistem untuk penelitian ini, perancangan sistem akan dibagi menjadi dua yaitu perancangan perangkat keras dan perancangan perangkat lunak.

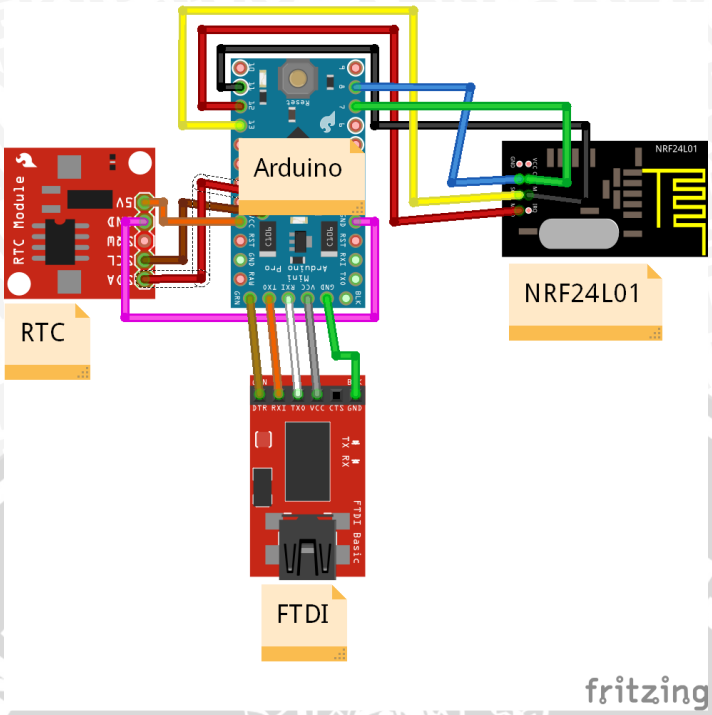
### 5.1.1 Perancangan Perangkat Keras

Perancangan perangkat keras pada sistem ini adalah perancangan untuk *node* RSU (Road Side Unit) dan *node* OBU (On-Board Unit). Penelitian ini berfokus pada pengiriman paket data yang memanfaatkan NRF24L01 sebagai modul pengirim dan penerima maka dari itu pada sistem ini tidak menggunakan sensor sebagai input untuk paket data. Sehingga perancangan perangkat keras untuk RSU dan OBU kurang lebih akan sama, diagram blok perancangan bisa dilihat pada gambar 3.2 dan gambar 3.4

Gambar 3.2 dan gambar 3.4 merupakan diagram blok untuk perancangan *node* RSU dan OBU. Perancangan untuk kedua *node* ini kurang lebih sama hanya sedikit bagian saja yang membedakan perancangan pada *node* RSU dan *node* OBU yaitu pada tombol darurat pada *node* OBU. Pada sistem ini *node* RSU maupun OBU akan dibagi menjadi 3 bagian yaitu FTDI yang berfungsi untuk mendownload algoritma yang ada dan memasukkannya ke bagian selanjutnya yaitu Arduino yang berfungsi untuk mengolah data dan membuat paket data sehingga paket data siap dikirim, bagian ketiga dari sistem ini ada NRF24L01 yang bertugas untuk mengirim dan menerima paket data.

#### 5.1.1.1 Perancangan Rangkaian Elektronik

Perancangan perangkat keras khususnya rangkaian elektronik dalam ini sistem ini memanfaatkan Arduino, NRF24L01, FTDI, dan RTC. Mikrokontroller yang digunakan adalah Arduino Pro Mini degan modul downloadernya yaitu FTDI. NRF24L01 digunakan sebagai modul komunikasi antar *node*.




**Gambar 5. 1 Rancangan Rangkaian Elektronik *Node* RSU**

**Tabel 5. 1 Keterangan Hubungan Pin nRF24L01 dan Arduino Pro Mini**

| Pin NRF24L01 | Pin Ardiono Pro Mini | Warna |
|---|---|---|
| CSN | 7 | |
| CE | 8 | |
| MOSI | 11 | |
| MISO | 12 | |
| SCK | 13 | |

**Tabel 5. 2 Keterangan Hubungan Pin FTDI dan Arduino Pro Mini**

| Pin FTDI | Pin Arduino Pro Mini | Warna |
|---|---|---|
| GND | GND | |
| VCC | VCC | |
| TX0 | RX1 | |
| RX1 | TX0 | |
| DTR | GRN | |

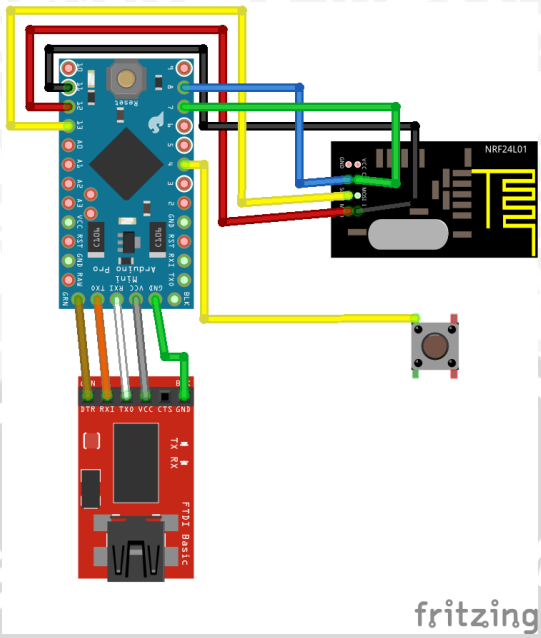
**Tabel 5. 3 Keterangan Hubungan Pin RTC dan Arduino Pro Mini**

| Pin RTC | Pin Arduino Pro Mini | Warna |
|---|---|---|
| 5V | VCC | |
| GND | GND | |
| SCL | A4 | |
| SDA | A5 | |

Pada gambar 5.1, sistem menggunakan arduino sebagai kontroler, FTDI sebagai modul *downloader* untuk Arduino. Pada tabel 5.1 FTDI memiliki 5 pin yang dihubungkan pada arduino antara lain adalah pin GND yang juga dihubungkan pada pin GND pada Arduino, pin VCC yang dihubungkan pada pin VCC Arduino untuk mengambil catu daya dari arduino dan pin TX0 dihubungkan pada RX1 Arduino untuk pengiriman data dari FTDI menuju arduino, pin RX1 dihubungkan pada TX0 Arduino untuk pengiriman data dari Arduino menuju FTDI.

Pada tabel 5.2 NRF24L01 memiliki 5 pin yang dihubungkan dengan Arduino Pro Mini seperti yang ditunjukkan pada tabel 5.1 pin tersebut antara lain CSN, SCK, MISO, MOSI, dan CE. Pin CSN, SCK, MISO, dan MOSI merupakan pin *untuk SPI interface* yang bertugas untuk tranmisi data dan penerimaan data, pin CSN sendiri merupakan pin yang berguna untuk melakukan listening melalui *SPI port* lalu memprosesnya, sementara 3 pin lainnya merupakan *user hardware SPI interface*. Pin CE digunkan untuk control data pengiriman dan penerimaan saat NRF24L01 berada pada mode TX dan RX. Pin CSN dihubungkan ke pin 7 arduino, pin CE dihubungkan ke pin 8 arduino, pin MOSI dihubungkan dengan pin 11 arduino, pin MISO dihubungkan dengan pin 12, dan pin SCK dihubungkan dengan pin 13 pada arduino.

Pada tabel 5.3 RTC memiliki 4 pin yang dihubungkan dengan Arduino antara lain pin VCC, GND, SDA, dan SCL. Pin VCC dan GND seperti biasa dihubungkan ke pin VCC dan pin GND pada Arduino untuk mengambil catu daya, sementara 2 pin yang lainnya SDA dan SCL merupakan pin untuk serial data yang berfungsi untuk mengirim data ke Arduino dan *serial clock* yang berfungsi mengatur waktu. Kedua pin tersebut memiliki tempat sendiri di Arduino yaitu pada pin A4 dan A5.




**Gambar 5. 2 Rancangan Rangkaian Elektronik *Node* OBU**

**Tabel 5. 4 Keterangan Hubungan Pin nRF24L01 dan Arduino Pro Mini**

| Pin NRF24L01 | Pin Ardiono Pro Mini | Warna |
|---|---|---|
| CSN | 7 | |
| CE | 8 | |
| MOSI | 11 | |
| MISO | 12 | |
| SCK | 13 | |

**Tabel 5. 5 Keterangan Hubungan Pin FTDI dan Arduino Pro Mini**

| Pin FTDI | Pin Arduino Pro Mini | Warna |
|---|---|---|
| GND | GND | |
| VCC | VCC | |
| TX0 | RX1 | |
| RX1 | TX0 | |
| DTR | GRN | |

Pada gambar 5.2 Diatas, sistem menggunakan arduino sebagai kontroler, FTDI sebagai modul downloader untuk Arduino, pada tabel 5.4 FTDI memiliki 5 pin yang dihubungkan pada arduino antara lain adalah pin GND yang juga dihubungkan pada pin GND pada Arduino, pin VCC yang dihubungkan pada pin VCC Arduino untuk

mengambil catu daya dari arduino, pin TX0 dihubungkan pada RX1 Arduino untuk pengiriman data dari FTDI menuju arduino, pin RX1 dihubungkan pada TX0 Arduino untuk pengiriman data dari Arduino menuju FTDI.

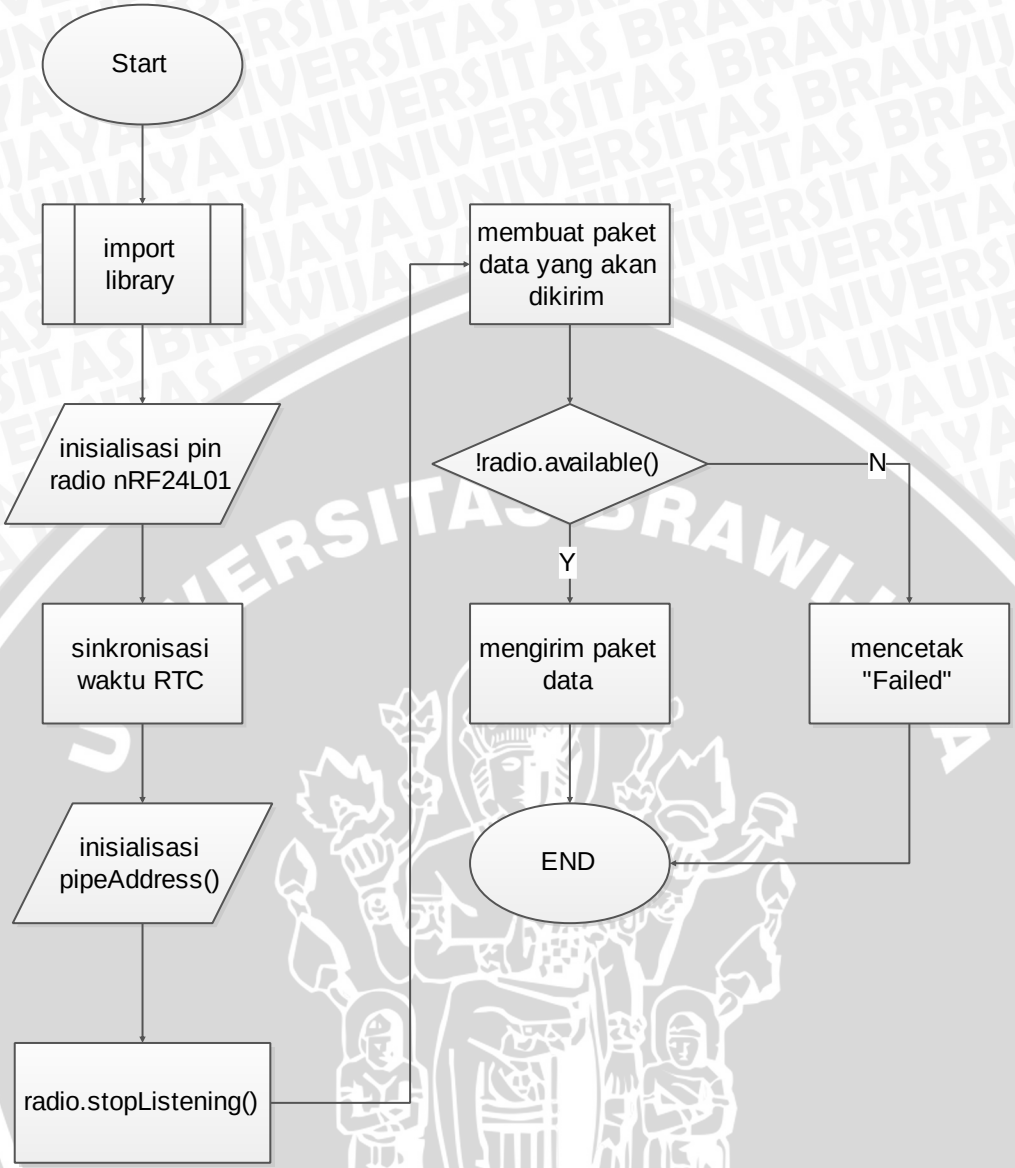
Pada tabel 5.5 NRF24L01 memiliki 5 pin yang dihubungkan dengan Arduino Pro Mini seperti yang ditunjukkan pada table 5.1 pin tersebut antara lain CSN, SCK, MISO, MOSI, dan CE. Pin CSN, SCK, MISO, dan MOSI merupakan pin untuk SPI interface yang bertugas untuk tranmisi data dan penerimaan data, pin CSN sendiri merupakan pin yang berguna untuk melakukan listening melalui SPI port lalu memprosesnya, sementara 3 pin lainnya merupakan user hardware SPI interface. Pin CE digunkan untuk control data pengiriman dan penerimaan saat NRF24L01 berada pada mode TX dan RX. Pin CSN dihubungkan ke pin 7 arduino, pin CE dihubungkan ke pin 8 arduino, pin MOSI dihubungkan dengan pin 11 arduino, pin MISO dihubungkan dengan pin 12, dan pin SCK dihubungkan dengan pin 13 pada arduino.

### 5.1.2 Perancangan Perangkat Lunak

Perancangan perangkat lunak yang akan digunakan pada sistem ini akan dibagi menjadi dua bagian yaitu perancangan perangkat lunak untuk RSU dan untuk OBU. Perangkat lunak RSU dirancang untuk membuat paket data yang berisikan informasi dan mengirimkan informasi secara broadcast dan terus menerus ke *node-node* OBU yang berada di sekitar areanya, sedangkan *node* OBU dirancang untuk menerima paket-paket data yang dikirimkan oleh RSU tadi. Selain itu OBU juga mampu mengirimkan pesan darurat apabila terjadi keadaan darurat.

#### 5.1.2.1 Perancangan Perangkat Lunak *Node* RSU (*Road Side Unit*)

*Node* RSU akan diprogram menggunakan bahasa pemgrograman C yang memanfaatkan library RF24 untuk metode pengiriman datanya. Program RSU dalam pembuatan dan pengiriman data dapat dilihat pada diagram alir pada gambar 5.3



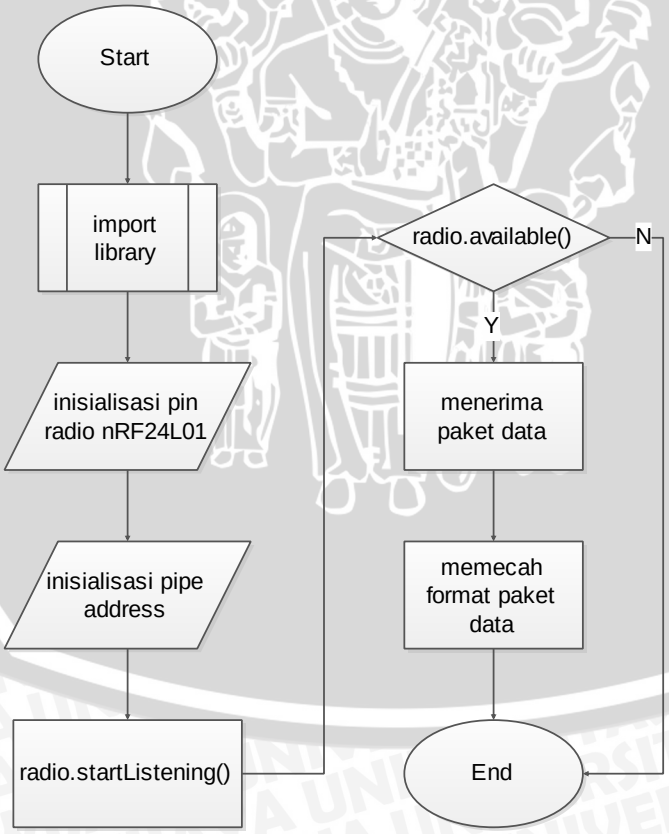

**Gambar 5. 3 Diagram Alir Algoritma *Node* RSU**

Gambar 5.3 merupakan diagram alir dari algoritma *node* RSU. Sebelum melakukan pengiriman data *node* RSU terlebih dahulu melakukan import library untuk semua hardware dan algoritma yang digunakan seperti library untuk nRF24L01, RTC, dan algoritma RF24. Kemudian perlu dilakukan inisialisasi pin radio nRF24L01 yaitu CE dan pin CSN untuk melakukan komunikasi melalui radio. Karena *node* RSU berperan sebagai sumber informasi maka *node* RSU perlu melakukan set waktu. Untuk bisa mengirimkan waktu maka arduino harus mengesetnya terlebih dahulu dengan cara mensinkronisasikannya dengan RTC. Untuk bisa melakukan pengiriman data diperlukan alamat untuk berkomunikasi. Alamat komunikasi yang digunakan pada sistem ini ada *pipe address* yang perlu dilakukan adalah menginisialisasi *pipe address* yang digunakan dalam pengiriman data. Ada 2 *pipe address* yang digunakan yaitu *openWritingPipe* merupakan alamat yang digunakan untuk mengirim data dan *openReadingPipe* merupakan alat yang digunakan untuk menerima data. Pada sistem ini sendiri hanya digunakan 1 *pipe address* sehingga

*pipe address* untuk *openWritingPipe* dan *openReadingPipe* memiliki alamat yang sama yaitu 0xF0F0F0F0E1LL. Pada NRF24L01 ini sebuah *node* tidak bisa melakukan pengiriman dan penerimaan data secara bersamaan sehingga penggunaannya harus bergantian antara *openWritingPipe* da *openReadingPipe* sehingga pada saat melakukan pengiriman data *openReadingPipe* tidak diaktifkan dengan cara memanggil fungsi *radio.stopListening()*. Ketika semua setting *node* RSU sudah dilakukan selanjutnya adalah mempersiapkan paket data yang akan dikirim. Format paket data yang dikirim tidak mengikuti format paket data IEEE 802.11 karena format paket data tersebut terlalu besar sehingga tidak cukup di dalam memory arduino dan nRF24L01 sehingga paket data yang dikirim hanya berisi pengirim, pesan, dan status lalu lintas. Sebelumnya melakukan pengiriman data perlu dilakukan pengecekan kembali apakah radio sudah dalam mode mengirim paket data atau tidak. Ketika radio tidak dalam mode mengirim data maka paket data tidak akan dikirim dan akan mencetak "Failed", sedangkan ketika radio dalam mode mengirim data maka paket data yang sudah disiapkan akan dikirimkan secara broadcast.

### 5.1.2.2 Perancangan Perangkat Lunak *Node* OBU (*On Board Unit*)

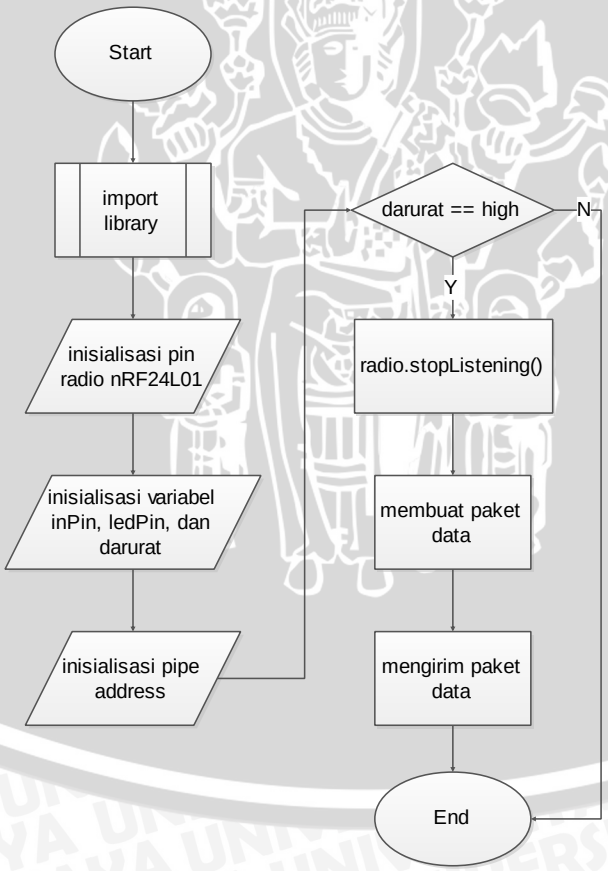
Perancangan *Node* OBU sama dengan *node* RSU. Program akan dibuat dengan mengunakan bahasa C. Algoritma program ada pada diagram alir pada Gambar 5.4.




**Gambar 5. 4 Diagram Alir Algoritma *Node* OBU menerima data**

Gambar 5.4 merupakan diagram alir algoritma program untuk *node* OBU dalam melakukan penerimaan data. Hal yang sama dilakukan pada *node* OBU yaitu

mengimport library untuk semua hardware dan algoritma yang digunakan dan melakukan inisialisasi untuk pin radio yang digunakan nRF24L01. Pada *node* OBU tidak perlu dilakukan pengaturan waktu seperti yang dilakukan pada *node* OBU sehingga langsung melakukan inisialisasi *pipe address* yang digunakan dan harus menggunakan *pipe address* yang sama seperti pada *node* RSU agar dapat melakukan penerimaan data dari *node* RSU. Seperti yang dijelaskan sebelumnya *node* pada nRF24L01 hanya bisa melakukan pengiriman data atau penerimaan data saja dalam satu waktu, karena *node* RSU sudah berperan sebagai pengirim data maka *node* OBU akan berperan sebagai penerima data yang dapat diaktifkan dengan cara menjalankan fungsi *radio.startListening()*. Sebelum melakukan penerimaan data perlu dilakukan kembali apakah radio sudah dalam mode menerima atau tidak dengan cara mengecek *radio.available()*, ketika radio siap menerima data maka data yang sudah dikirim akan diterima oleh *node* OBU melalui nRF24L01. Perlu juga dilakukan pemecahan paket data karena data yang diterima dalam bentuk string sehingga perlu dilakukan pemecahan data untuk mengetahui siapa pengirim pesan, pesan apa yang diberikan, dan bagaimana status lalu lintas. Ketika radio tidak dalam mode menerima data maka otomatis *node* OBU tidak akan menerima data apapapuun dan program berakhir.




**Gambar 5. 5 Diagram Alir Pengiriman Data *Node* OBU**

Gambar 5.5 menjelaskan fungsi lain *node* OBU selain melakukan penerimaan data, yaitu mengirim pesan darurat. Hal sama kembali dilakukan yaitu melakukan import library dan menginisialisasi pin radio yang digunakan nRF24L01, namun

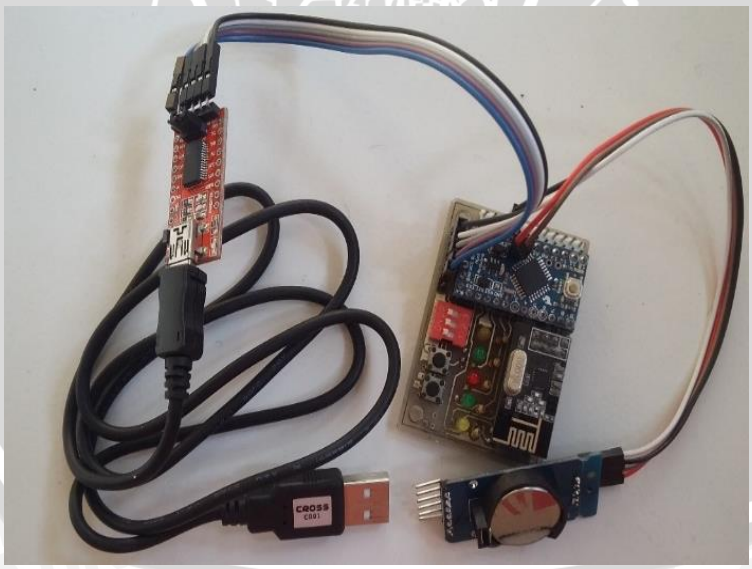
sebelum melakukan pengaturan *pipe address* perlu dilakukan inisialisasi variable yang digunakan sebagai input dan output saat keadaan darurat. Input yang digunakan adalah push button dan output yang digunakan ada LED. Kemudian melakukan pengaturan *pipe address* yang sama seperti yang digunakan oleh *node* RSU sebelumnya. Untuk perubahan mode yang dilakukan setelah melakukan pengecekan apakan tombol darurat sudah ditekan atau belum. Ketika tombol tidak ditekan maka tidak akan terjadi apa-apa dan program berakhir, sedangkan ketika tombol ditekan maka *node* OBU akan berubah mode menjadi siap mengirim data dengan menjalankan fungsi *radio.stopListening* dan kemudian *node* OBU membuat paket data yang siap dikirimkan, format data yang dikirimkan sama seperti format data yang dikirimkan *node* RSU. Saat data sudah selesai dibuat dan siap dikirim maka data tersebut akan dikirimkan secara broadcast.

## 5.2 Implementasi Sistem

Pada sup bab Implementasi Sistem dijelaskan tentang penerapan perancangan perangkat lunak dan perangkat keras. Selain itu juga dijelaskan batasan implementasi.

### 5.2.1 Implementasi *Node* RSU (Road Side Unit)

Implementasi *Node* RSU dilakukan sesuai dengan perancangan perangkat keras dan perangkat lunak yang sudah dijelaskan sebelumnya. Untuk implementasi perangkat keras dapat dilihat pada Gambar 5.6.



**Gambar 5. 6 Implementasi *Node* RSU**

Gambar 5.6 merupakan gambar perangkat keras *node* RSU, seperti yang sudah dijelaskan pada bab metodologi bagian perancangan sistem dan kembali dijelaskan pada bagian perancangan perangkat keras bab ini. Perangkat keras *node* RSU terdiri dari arduino pro mini, NRF24L01, FTDI, dan RTC. Semua perangkat keras harus

tersambung sesuai dengan keterangan yang sudah dijelaskan pada tabel 5.1, tabel 5.2 dan tabel 5.3

Untuk mengimplementasikan *source code* dari *node* RSU yang berperan sebagai sumber informasi perlu dilakukan implementasi perangkat lunak menggunakan arduino IDE dengan bahasa pemrograman C. Karena ada beberapa perangkat keras yang digunakan pada sistem ini maka diperlukan beberapa library dari masing-masing perangkat keras agar *source code* yang dijalankan bisa berjalan dengan baik. Library yang digunakan antara lain SPI.h dan nRF24L01.h digunakan untuk perangkat keras nRF24L01, time.h, Wire.h, dan DS1307RTC.h digunakan untuk perangkat keras RTC yang berguna untuk mengeset waktu pada arduino, ada pula libraru RF24.h yang merupakan library untuk format pengiriman data nRF24L01. Selain mengimport library yang digunakan perlu juga melakukan inisialisasi pin radio yang digunakan nRF24L01 dan juga format pengiriman data dari nRF24L01 itu sendiri.

**Tabel 5. 6 Import Library *node* RSU**

| Baris | Source Code |
|---|---|
| 1 | `#include <SPI.h>` |
| 2 | `#include "nRF24L01.h"` |
| 3 | `#include "RF24.h"` |
| 4 | `#include "printf.h"` |
| 5 | `#include <Time.h>` |
| 6 | `#include <Wire.h>` |
| 7 | `#include <DS1307RTC.h>` |
| 8 | `RF24 radio(8,7);` |

Sesuai dengan tabel 5.6 diatas library yang digunakan beragam untuk membuat setiap perangkat keras bekerja dengan semestinya. Baris 1-7 menjelaskan library apa saja yang digunakan dalam sistem ini, library yang digunakan antara lain SPI.h, nRF24L01.h, FR24.h, Time.h, Wire.h, DS1307RTC.h. ada pula inisialisasi RF24 radio yang merupakan insialisasi pin yang digunakan untuk melakukan transmisi data yang dapat dilihat pada baris 9, pin yang digunakan adalah pin 8 dan pin 7.

**Tabel 5. 7 Setup Code *Node* RSU**

| Baris | Source COde |
|---|---|
| 1 | `void setup(void)` |
| 2 | `{` |
| 3 | `  Serial.begin(9600);` |
| 4 | `  while (!Serial) ;` |
| 5 | `  setSyncProvider(RTC.get);` |
| 6 | `  if(timeStatus()!= timeSet)` |

```
7       Serial.println("Unable to sync with the RTC");
8     else
9       Serial.println("RTC has set the system
10  time");
11    radio.begin();
12    radio.openWritingPipe(0xF0F0F0F0E1LL);
13    radio.openReadingPipe(1,0xF0F0F0F0E1LL);
14    radio.stopListening();
15  }
```

Berdasarkan tabel 5.7 Diatas *void setup()* berisi pengaturan untuk arduino dalam menjalankan program *node* RSU. Pada baris 3 berisi pengaturan *baudrate* yang nantinya digunakan saat menjalankan program *node* RSU dan *baudrate* yang digunakan adalah 9600 karena *baudrate* yang digunakan akan berpengaruh pada waktu di arduino sehingga ketika digunakan terlalu besar waktu tidak akan pas. untuk melakukan setting waktu akan dilakukan ketika serial monitor sudah terbuka seperti yang terlihat pada baris 5, kemudian pada baris 6 dilakukan sinkronisasi waktu dengan memanfaatkan fungsi *setSyncProvider* dan RTC.get, setelah dilakukan sinkronisasi waktu kemudian dilakukan pengecekan pada baris 7-10 apakah waktu benar-benar sudah diset atau belum, ketika waktu sudah diset maka akan menampilkan "*Unable to sync with the RTC*" dan ketika waktu sudah diset maka akan menampilkan "*RTC has set the system time*". Karena pengiriman data oleh nRF24L01 memanfaatkan radio maka perlu menyalakan fungsi radio terlebih dahulu yang terlihat pada bariss 11. Seain itu dilakukan juga *setting pipe address* yang digunakan *node* RSU yang bisa dilihat pada baris 12 dan 13. Karena *node* RSU berperan sebagai sumber informasi yang mengirimkan data maka pada baris 14 ini menjelaskan bahwa radio pada *node* OBU tidak akan melakukan listening terhadap paket data karena dalam keadaan siap mengirim data.

**Tabel 5. 8 *Source Code* Pengiriman Data *Node* RSU**

| Baris | Source Code |
|---|---|
| 1 | `void loop(){` |
| 2 | `  struct dataStruct {` |
| 3 | `  int jam = hour();` |
| 4 | `  int menit=minute();` |
| 5 | `  int detik=second();` |
| 6 | `} waktu;` |
| 7 | `  i++;` |
| 8 | `  Serial.println(i);` |
| 9 | `  radio.stopListening();` |

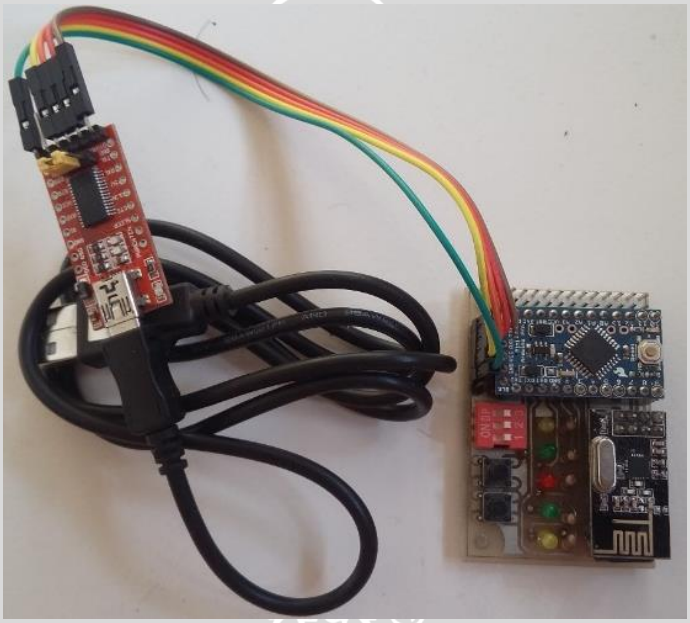
```
10  Char text[] = "RSU#Jalan Lancar#Tidak Berbahaya";
11    Serial.println("Mengirim ");
12    digitalClockDisplay();
13    const char separator[] = "#";
14    char *bacateks;
15    bacateks = strtok(text, s);
16    if ( bacateks != NULL ){
17      printf( " pengirim : %s\n", bacateks);
18      bacateks = strtok(NULL, s);
19    }
20    if (bacateks != NULL){
21       printf("pesan : %s\n",bacateks);
22       bacateks = strtok(NULL, s);
23    }
24    if (bacateks != NULL){
25        printf("status : %s\n", bacateks);
26        bacateks = strtok(NULL, s);
27    }
28    if(!radio.available()){
29      radio.write(&waktu, sizeof(waktu));
30      radio.write(&text, sizeof(text));
31    }
32    else{
33      Serial.println("Failed");
34      Serial.println("====================");
```

Berdasarkan tabel 5.8 diatas *void loop()* merupakan code yang berisi *source code* pengiriman data *node* RSU. Pada baris 2-6 dilakukan pembuatan *struct* untuk membuat data waktu yang sudah dilakukan sinkronisasi yang nantinya akan dikirim, pada baris 10 dilakukan terlebih dahulu pembuatan data yang akan dikirim, data yang dikirim akan berbentuk string yang berisi pengirim, pesan yang disampaikan, dan bagaimana status kendaraan. Ketiga data tersebut dimasukkan dalam 1 vaiable dan dipisahkan oleh tanda '#'. Pada baris 2 dan bari 3 akan mencetak tulisan "mengirim" dan akan menampilkan waktu yang sudah disinkronisasi pada RTC sebelumnya. Pada baris 3-23 dilakukan pemisahan data yang dikirim oleh *node* RSU untuk memastikan bagaimana data nanti diterima oleh *node* OBU, data yang dibaca adalah data *'text'* yang merupakan data yang akan dikirim, dalam setiap pembacaan data ketika bertemu dengan tanda '#' berarti sudah berada pada akhir data.

Sehingga ketika keseluruhan data dibaca akan menghasilkan data pengirim, pesan, dan status lalu lintas. data yang dibuat ini juga bersifat dumb karena data tidak berasal dari input manapun. karena data yang akan dikirim bersifat *dumb* atau data buatan sendiri tanpa input apapun maka data dimasukkan ke dalam variabel char, paket data tersebut berisi informasi pengirm, pesan yang dikirim dan bagaimana status lalu lintas. Pada baris 28 kembali dilakukan pengecekan apakah radio sedang dalam mode menerima atau tidak, ketika radio dalam posisi siap menerima data maka nRF24L01 tidak dapat melakukan pengiriman data, pengiriman data baru bisa dilakukan ketika nRF24L01 dalam mode pengiriman data, ketika nRF24L01 dalam mode pengirima data maka data yang berisi pesan dan waktu akan dikirim.

### 5.2.2 Implementasi *Node* OBU (On Board Unit)

Implementasi pada OBU tidak berbeda jauh dengan *node* RSU, implementasi dilakukan sesuai dengan yang sudah dijelaskan pada tahap perancangan. Perancangan perangkat keras *node* OBU dapat dilihat pada gambar 5.7.



**Gambar 5. 7 Implementasi *Node* OBU**

Gambar 5.7 merupakan gambar kerja *node* OBU, hampir sama dengan *node* RSU *node* OBU juga terdiri dari arduino sebagai mikrokontroler, NRF24L01 sebagai modul transmitter, dan FTDI sebagai downloader arduino. Rangkaian untuk *node* OBU ini juga bisa dilihat pada gambar *node* OBU dan tabel keterangan.

Untuk mengimplementasikan *source code* dari *node* OBU yang berperan sebagai penerima informasi yang perlu dilakukan sama seperti yang dilakukan pada RSU sebelumnya yaitu perlu dilakukan implementasi perangkat lunak menggunakan arduino IDE dengan bahasa pemrograman C. perangkat yang digunakan pada *node* OBU tidak sebanyak pada *node* RSU, pada *node* OBU tidak menggunakan RTC yang berfungsi untuk mensetting waktu sehingga *node* OBU tidak memerlukan library

time.h, wire.h, dan DS1307RTC.h, library yang digunakan pada *node* OBU antara lain SPI.h dan nRF24L01.h. Selain mengimport library yang digunakan perlu juga melakukan inisialisasi pin radio yang digunakan nRF24L01 dan juga format pengiriman data dari nRF24L01 itu sendiri.

**Tabel 5. 9 Import Library *Node* OBU**

| Baris | Source Code |
|---|---|
| 1 | `#include <SPI.h>` |
| 2 | `#include "nRF24L01.h"` |
| 3 | `#include "RF24.h"` |
| 4 | `RF24 radio(8,7);` |

Dapat dilihat dari tabel 5.9 ada beberapa library yang digunakan seperti yang terlihat pada baris 1-3, library yang digunakan antara lain SPI.h, nRF24L01.h, dan RF24.h. selain mengimport library dilakukan juga inisialisasi ping radio yang digunakan pada nRF24L01 pada baris 5, ping yang digunakan nRF24L01 adalah pin 8 dan pin 7.

**Tabel 5. 10 Setup Code *Node* OBU**

| Baris | Source Code |
|---|---|
| 1 | `void setup(void){` |
| 2 | `  Serial.begin(9600);` |
| 3 | |
| 4 | `  radio.begin();` |
| 5 | `  radio.openWritingPipe(0xF0F0F0F0E1LL);` |
| 6 | `  radio.openReadingPipe(1,0xF0F0F0F0E1LL);` |
| 7 | `  radio.startListening();` |
| 8 | `  }` |

Pada table 5.10 baris 2 berisi pengaturan baudrate yang nantinya digunakan saat menjalankan program *node* OBU dan baudrate yang digunakan adalah 9600 karena baudrate yang digunakan harus sama seperti baudrate yang digunakan oleh *node* RSU, ketika baudrate yang digunakan berbeda maka tidak akan bisa dilakukan penerimaan data atau data yang nanti ditema akan rusak. Karena pengiriman data oleh nRF24L01 memanfaatkan radio maka perlu menyalakan fungsi radio terlebih dahulu yang terlihat pada bariss 4. Seain itu dilakukan juga setting *pipe address* yang digunakan *node* OBU yang bisa dilihat pada baris 5 dan 6, piep address yang digunakan juga harus sama seperti yang digunakan oleh *node* RSU agar komunikasi antar *node* RSU dan *node* OBU bisa terjadi. Berbeda dengan *node* OBU yang berperan sebagai sumber informasi, *node* OBU berperan sebagai penerima maka

pada baris 7 ini menjelaskan diaktifkan fungsi radio.startListening() untuk siap menerima data.

**Tabel 5. 11 *Source Code* Penerimaan Data *Node* OBU**

| Baris | Source Code |
|---|---|
| 1 | `void loop(){` |
| 2 | `struct dataStruct {` |
| 3 | `  int jam;` |
| 4 | `  int menit;` |
| 5 | `  int detik;` |
| 6 | `} waktu;` |
| 7 | `  if ( radio.available() )` |
| 8 | `    {` |
| 9 | `      while (radio.available()) {` |
| 10 | `        char text[32] = {0};` |
| 11 | `        radio.read( &text, sizeof(text) );` |
| 12 | |
| 13 | `        const char separator[] = "#";` |
| 14 | `        char *bacateks;` |
| 15 | `        bacateks = strtok(text, s);` |
| 16 | `        if ( bacateks != NULL ){` |
| 17 | `          printf( " data : %s\n", bacateks );` |
| 18 | `          bacateks = strtok(NULL, s);` |
| 19 | `        }` |
| 20 | `          if (bacateks != NULL){` |
| 21 | `            printf("pesan : %s\n",bacateks);` |
| 22 | `            bacateks = strtok(NULL, s);` |
| 23 | `        }` |
| 24 | `          if (bacateks != NULL){` |
| 25 | `            printf("status : %s\n",bacateks);` |
| 26 | `            bacateks = strtok(NULL, s);` |
| 27 | `        }` |
| 28 | `        Serial.print(waktu.jam);` |
| 29 | `        Serial.print(":");` |
| 30 | `        Serial.print(waktu.menit);` |

```
31          Serial.print(":");
32          Serial.println(waktu.detik);
33          Serial.println("====================");
34          delay(20);
35        }
```

Dari tabel 5.11 dapat dilihat bagaimana *source code* penerimaan data berjalan pada *node* OBU. *Source code* penerimaan *node* OBU terletak pada void setup(), dimulai dari baris 2-6 dimana *node* OBU membuat *struct* untuk menerima data waktu yang dikirim *node* RSU, kemudian pada baris 7 dilakukan pengecekan ulang apakah *node* OBU dalam keadaan siap menerima data atau tidak, ketika *node* OBU siap menerima maka maka akan dibuat variable yang berguna untuk menampung data yang diterima seperti yang terlihat pada baris 10, saat variable sudah siap baris 6 akan menerima paket data yang dikirim. Saat data sudah diterima baris 13-27 merupakan *source code* yang digunakan untuk memecah paket data, sama seperti yang dilakukan *node* RSU sebelumya. Data akan dibaca sampai bertemu dengan tanda '#' yang berarti itu merupakan akhir dari satu paket data dan terus membaca seperti itu sampai akhir paket data sehingga nantinya akan menerima data pengirim, pesan yang disampaikan, dan status lalu lintas, serta waktu yang tadi sudah diterima akan ditampilkan pada baris 28-33. Pada baris 34 diberikan delay sebesar 20ms, hal ini dilakukan agar penerimaan data tidak berlebihan dan terjadi spam.

**Tabel 5. 12 *Source Code* Pengiriman Data *Node* OBU**

| Baris | Source Code |
|---|---|
| 1 | `int inPin = 6;` |
| 2 | `int ledPin = 9;` |
| 3 | `int darurat = 0;` |
| 4 | |
| 5 | `void setup(){` |
| 6 | `  pinMode(ledPin, OUTPUT);` |
| 7 | `  pinMode(inPin, INPUT);` |
| 8 | `}` |
| 9 | `void loop(){` |
| 10 | `  darurat = digitalRead(inPin);` |
| 11 | `  if (darurat == HIGH){` |
| 12 | `    //digitalWrite(ledPin, LOW);` |
| 13 | `    Serial.println("sedang darurat");` |
| 14 | `    radio.stopListening();` |

```
15      const char text[] = "Keadaan Darurat !!!";
16      Serial.println("==================");
17      Serial.println("Mengirim ");
18      radio.write(&text, sizeof(text));
19      Serial.println(text);
20      Serial.println("====================");
21      }
```

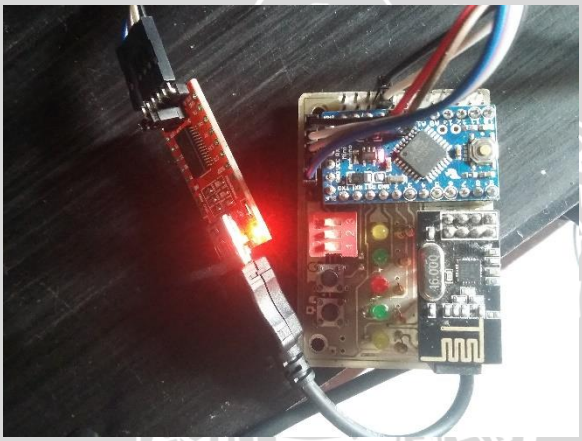
Tabel 5.12 merupakan *source code* lain untuk *node* RSU selain menerima paket data, yaitu untuk mengirimkan pesan darurat. Diawali dari inisialisasi variabel para baris 1-3 untuk input dan output, input berasal dari push button yang berada pada pin 6 arduino, sementara output merupakan yang berada pada pin 9, variabel darurat sendiri merupakan variabel yang menampung nilai dari push button nantinya. Kemudian perlu diinisialisasi bahwa inPin merupakan sebuah output dan ledPin merupakan sebuah out pada void setup seperti pada baris 6 dan baris 7. Saat program sudah berjalan akan langsung melakukan pembacaan input dari push button pada baris 10 dan dilanjutkan dengan pengecekan hasil pembacaan push button pada baris 11, ketika push button ditekan atau variable darurat bernilai high maka akan langsung mencetak "Sedang Darurat" dan merubah mode *node* OBU menjadi siap mengirim dengan menjalankan fungsi radio.startListening() pada baris 14. Setelah itu *node* OBU akan membuat paket data yang berisi pesan darurat untuk dikirimkan seperti yang terlihat pada baris 15, ketika data sudah siap kemudian pada baris 18 *node* OBU akan mengirimkan data secara broadcast.

# BAB 6 PENGUJIAN DAN ANALISIS

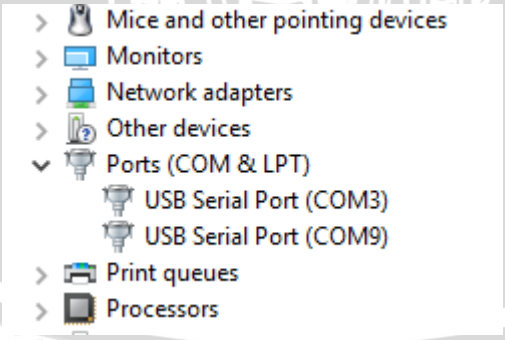
Pada bab ini akan dijelaskan bagaimana proses pengujian dari sistem yang sudah dirancang dan diimplementasikan pada bab sebelumnya. Pengujian akan dilakukan dari segi perangkat keras maupun perangkat lunak. Setelah melakukan pengujian akan didapatkan data hasil pengujian yang kemudian dilakukan analisis terhadap hasil pengujian dan masuk ke tahap penarikan kesimpulan berdasarkan hasil pengujian.

## 6.1 Pengujian Fungsionalitas Sistem

Pengujian ini dilakukan untuk memastikan *node* RSU dan *node* OBU sudah berfungsi sebagaimana mestinya. *Node* RSU dan *node* OBU dihubungkan dengan laptop melalui *port USB.* Jika *node RSU* dan *node* OBU sudah terhubung dengan computer, maka lampu indicator pada Arduino dan FTDI akan menyala dan pada *device manager* terlihat *serial port* yang terhubung seperti yang ditunjukkan pada gambar 6.1 dan gambar 6.2.

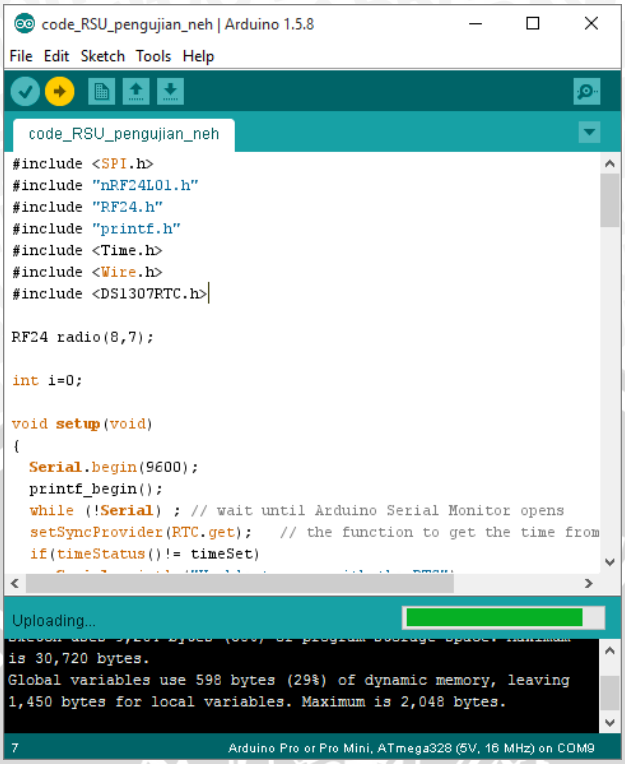


**Gambar 6. 1 *Node* Terhubung Dengan Komputer**

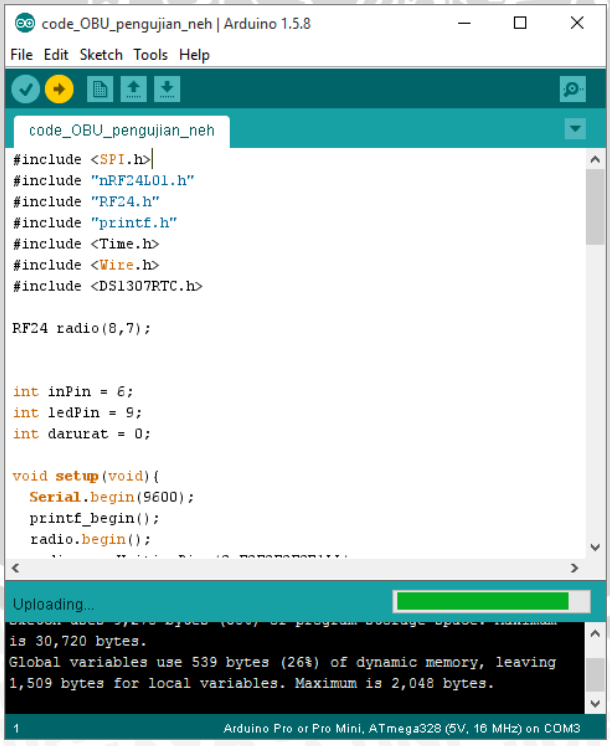



**Gambar 6. 2 *Serial Port* terhubung pada laptop**

Setelah kedua *node* terhubung, langkah selanjutnya adalah melakukan percobaan pengiriman data dari *node* RSU ke *node* OBU.

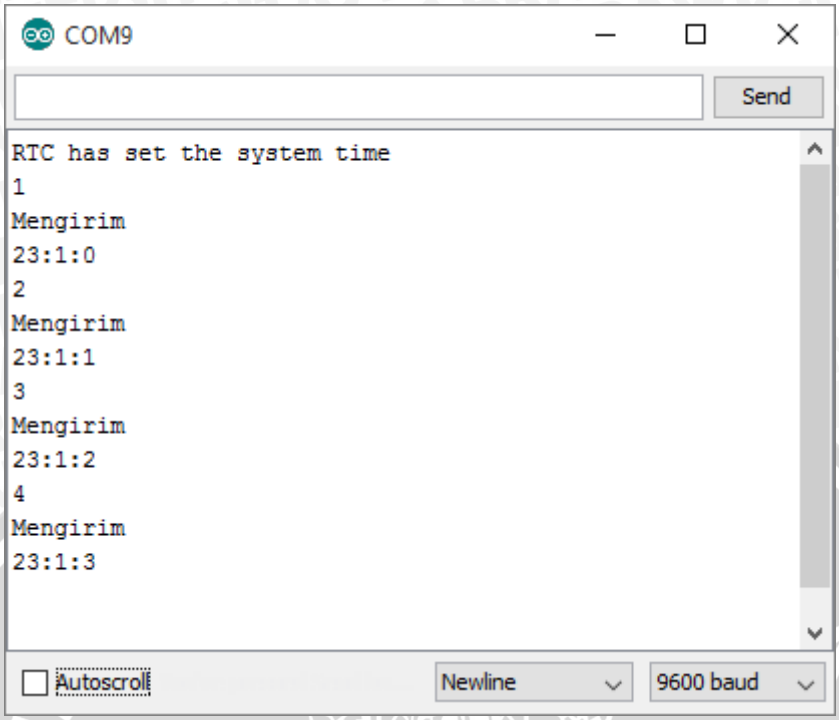



**Gambar 6. 3 Meng*upload Source Code Node* RSU**



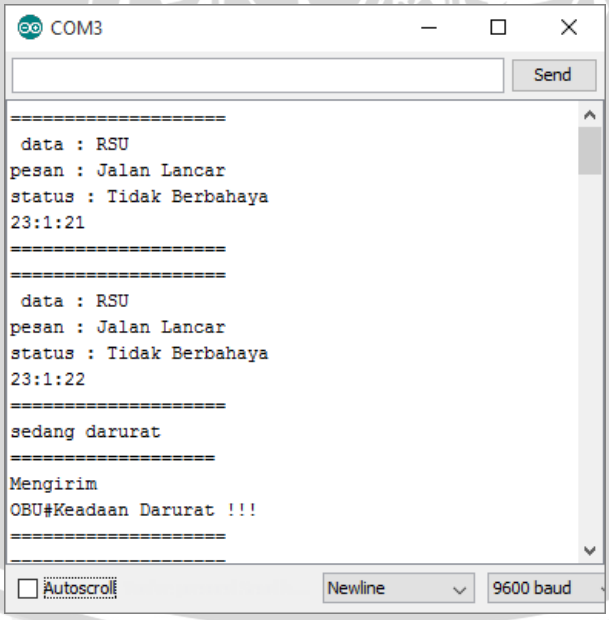

**Gambar 6. 4 Meng*upload Source Code Node* OBU**

Gambar 6.3 dan gambar 6.4 merupakan proses *upload source code node* RSU dan *node* OBU agar proses pengiriman dan penerimaan bisa terjadi.




**Gambar 6. 5 *Serial Monitor* pengiriman *node* RSU**

Gambar 6.5 merupakan tampilan *serial monitor* pada *node* RSU ketika melakukan pengiriman data ke *node* OBU




**Gambar 6. 6 *Serial Monitor* penerimaan *node* OBU**

Gambar 6.6 merupakan tampilan *serial monitor* pada *node* OBU ketika menerima data yang dikirim oleh *node* RSU, data yang diterma oleh OBU masih dalam satu format sehingga perlu dilakukan pemisahan data agar data yang diterima dapat dibaca secara jelas, setelah dilakukan pemisahan data lalu data ditampilkan dan dapat dibaca dengan baik.

## 6.2 Pengujian *Node* RSU

### 6.2.1 Tujuan Pengujian

Tujuan dari pengujian ini adalah untuk mengetahui apakah *node* RSU yang berperan sebagai sumber informasi mampu mengirimkan data ke *node* OBU dengan baik pada jarak 10m hingga 50m meskipun *node* OBU berjumlah lebih dari 1. Pengujian ini juga dilakukan karena merupakan kebutuhan fungsional dan kebutuhan non fungsional dari sistem yang dibuat, agar diketahui apakah kebutuhan bisa terpenuhi apa tidak pada hasil analisa.

### 6.2.2 Prosedur Pengujian

Pengujian ini dilakukan dengan cara menyalakan RSU dan melakukan pengiriman sebanyak 100x pada jarak 10m, 20m, 30m, 40m, dan 50m. hasil pengiriman data nanti akan dilihat seberapa besar packet loss, berapa kecepatan pengiriman data, dan berapa throughput nya.

### 6.2.3 Pelaksanaan Pengujian

Pengujian akan dilakukan dengan cara menyalakan *node* RSU dan *node* OBU, kemudian *node* RSU akan mengirimkan data secara broadcast yang nantinya akan diterima oleh *node* OBU. Dari data yang dikirim nanti dalam dilihat berapa waktu pengiriman data dan packet loss dari pengiriman tersebut, kemudian nanti juga akan dilakukan pengiriman terhadap besarnya troughput.

### 6.2.4 Hasil Pengujian

Hasil pengujian yang didapatkan akan dimasukkan kedalam tabel dan dihitung packet loss dan troughputnya dari hasil capture yang ada diwireshark dan jumlah paket data yang telah dibroadcast oleh *node* RSU, setelah itu dihitung rata – rata dari hasil pengujian packet loss dan troughput.

**Tabel 6. 1 Hasil Pengujian Layanan RSU**

| Jarak(meter) | Paket yang dikirim | Waktu Pengiriman (ms) | Packet loss | Throughput(Mbit/s) |
|---|---|---|---|---|
| 10 | 100 | 277.77 | 9.5 | 0.82 |
| 20 | 100 | 133.68 | 21.5 | 0.67 |
| 30 | 100 | 77.74 | 8 | 0.90 |
| 40 | 100 | 354.39 | 44.5 | 0.56 |
| 50 | 100 | 89.09 | 21 | 0.81 |
| Rata-rata | | 186.54 | 20.9 | 0.75 |

Pada Tabel 6.1 terdapat hasil dari pengujian yang telah dilakukan, pada bagian packet loss untuk mendapatkan persentasenya dilakukan perhitungan dengan menggunakan persamaan 6.1 dan.

$$Packet\ Loss = \frac{(Packe\_Transmitted - Packe\_Received)}{Packet\_Transmitted} \times 100\% \quad (6.1)$$

Untuk mendapatkan hasil pengujian troughput dapat digitung menggunakan persamaan 6.2.

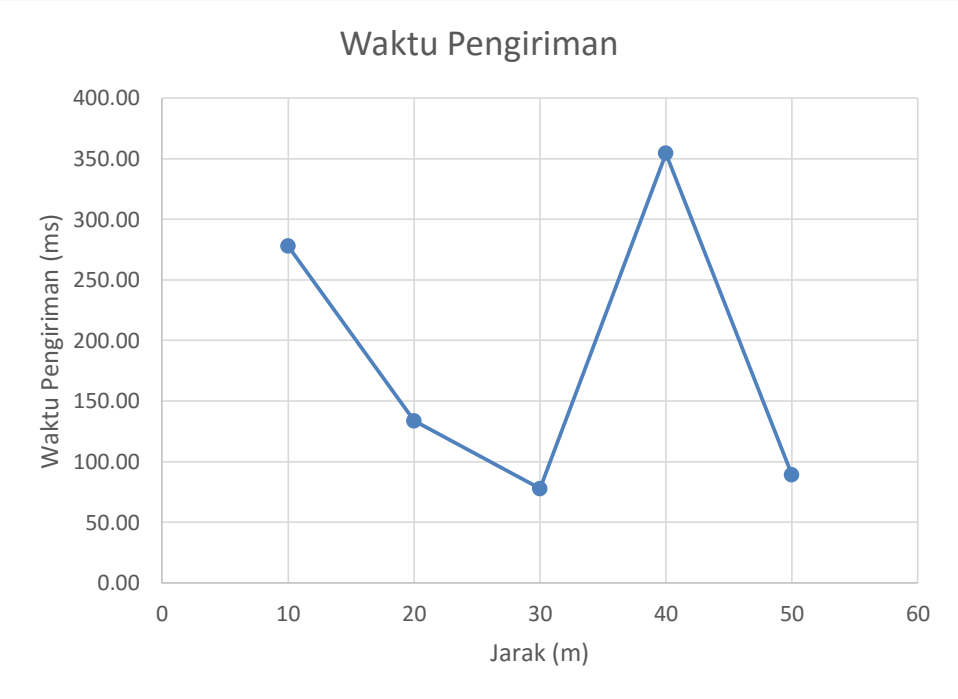
$$Troughput = \frac{Jumlah\ Data\ yang\ Dikirim}{Waktu\ Pengiriman\ Data} \quad (6.2)$$

Untuk mendapatkan nilai rata-rata dari pengujian dapat dihitung dengan persaan 6.3

$$Rata - Rata = \frac{\sum Hasil\ Perpengujian}{Banyak\ Pengujian} \quad (6.3)$$

### 6.2.5 Analisa Pengujian

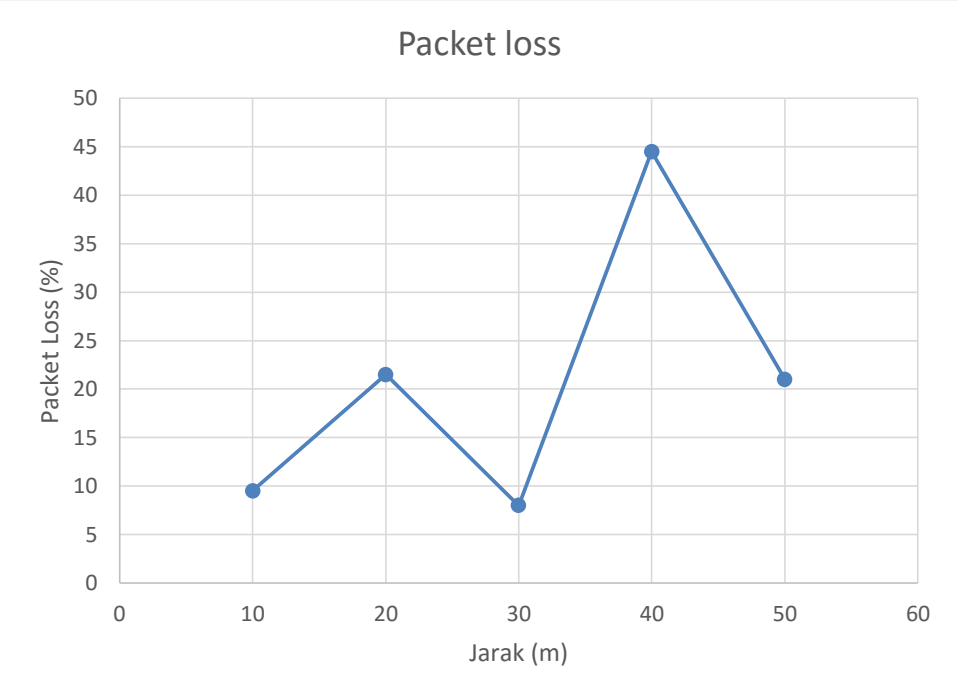
Dari hasil pengujian dapat dianalisa hasilnya dalam bentuk grafik seperti pada 3 grafik pada gambar 6.1, gambar 6.2, dan gambar 6.3




**Gambar 6. 7 Grafik Kecepatan Pengiriman RSU**

Gambar 6.7 merupakan gambar grafik dari hasil dari perhitungan round trip time (RTT) yang dilakukan oleh *node* RSU saat mengirimkan data sebanyak 100 kali ke 2 *node* OBU yang ada mulai dari 10m hingga 50m. bisa dilihat bahwa RTT pada jarak 10m cukup tinggi yaitu 277.77 ms hal ini dikarenakan saat melakukan pengujian terdapat angina dan juga area pengujian pada jarak ini sangat luas sehingga membuat pengiriman menjadi lebih lama, berbeda saat pengiriman pada jarak 20m yang memiliki area pengujian sedikit berbeda karena berada di sekitar perumahan yang tidak memiliki angin kencang yang membuat pengiriman menjadi

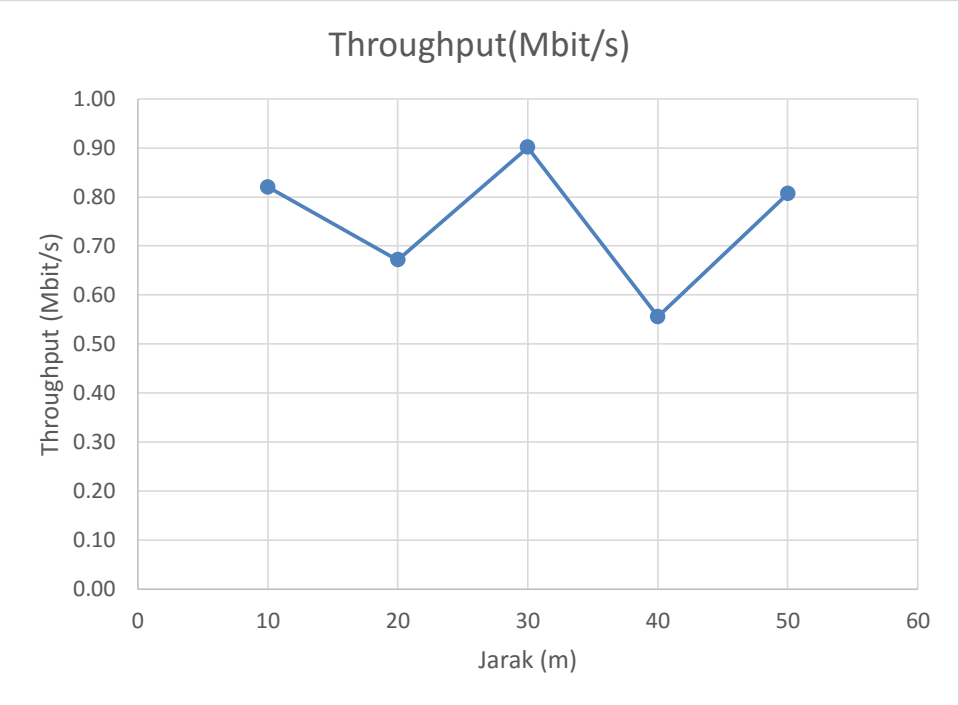
lebih cepat, pada pengujian jarak 30m dilakukan tidak ada angin sama sekali serta memiliki area yang sama seperti pengujian jarak 20m yang membuat waktu pengiriman data juga semakin cepat, sedangkan pada jarak 40m kecepatan pengiriman data kembali tinggi hal ini disebabkan karena jarak tersebut merupakan hampir jarak maksimal dari pengiriman menggunaka nRF24L01, pada jarak 50m kecepatan kembali meninggkat karena area yang digunakan sedikit landau sehingga pengiriman terjadi dari atas ke bawah yang ternyata berpengaruh besar terhadap kecepatan pengiriman data.




**Gambar 6. 8 Grafik *Packet Loss* Pengiriman RSU**

Gambar 6.8 merupakan grafik untuk packet loss saat *node* RSU melakukan pengiriman data sebanyak 100 kali pada 2 *node* OBU pada jarak 10m sampai 50m. dapat dilihat dari hasil pengujian mengalami packet loss dalam setiap pengujiannya. Pada jarak 10m terdapat packet loss yang sedikit hingga <10, hal ini bertolak belakang dengan kecepatan pengiriman data pada jarak 10m sebelumnya yang cukup tinggi biasanya akan menghasilkan *packet loss* yang tinggu juga namun *packet loss* pada jarak 10m cukup kecil hal ini bisa terjadi karena jarak antar *node* cukup dekat sehingga pengiriman hanya sedikit terganggu namun tidak sampai terputus, pada jarak 20m *packet loss* cukup banyak hingga >20 hal ini terjadi karena memang kecepatan pengiriman yang terjadi pada jarak 20m cukup tinggi sehingga proses komunikasi sedikit terganggu yang mengakibatkan banyaknya *packet loss,* pada jarak 30m *packet loss* yang terjadi sangat kecil karena memang kecepatan pengiriman sangat baik pada jarak 30m sehingga *packet loss* juga jarang terjadi, *packet loss* yang terjadi pada jarak 40m sangat tinggi karena memang kecepatan pengiriman yang cukup tinggi membuat kedua *node* sulit terhubung satu sama lain, sedangkan pada jarak 50m meskipun pengiriman data yang dilakukan cukup baik

dengan kecepatan yang juga baik tetap saja denga jarak yang cukup jauh *packet loss* juga cukup besar.




**Gambar 6. 9 Grafik *Throughput* Pengiriman RSU**

Gambar 6.9 merupakan grafik throughput dari pengujian *node* RSU yang nilainya didapatkan dari banyaknya data yang dikirim dibagi dengan waktu pengiriman data. Dapat dilihat bahwa throughput yang dihasilkan cukup bagus, pada pengujian jarak 10m throughput 0.82Mbit/s yang bisa dibilang bagus, hal ini tidak terlepad dari *packet loss* yang sedikit terjadi pada jarak 10m, pada jarak 20m *throughput* sedikit turun menjadi 0.67Mbit/s yang juga akibat dari *packet loss* yang terjadi pada jaram 20m cukup banyak, niali *throughput* kembali naik drastic pada jarak 30m karena memang hasil pengujian pada jarak ini sangat baik mulai dari kecepatan pengiriman dan *packet loss* yang sedikit membuat *throughput* yang dihasilkan juga menjadi sangat baik, berbanding terbalik pada jarak 40m yang merupakan hasil pengujian terburuk dilihat dari kecepatan pengiriman data dan *packet loss* yang banyak terjadi membuat *throughput* yang dihasilkan tidak terlalu tinggi, sementara pada jarak 50m *throughput* kembali naik karena *packet loss* yang terjadi memang tidak terlalu banyak pada jarak ini. Meskipun begitu secara keseluruhan dari hasil *throughput* bisa dibilang memuaskan karena memiliki nilai diatas 0.50 Mbit/s.

## 6.3 Pengujian *Node* RSU dengan *Node* OBU yang bergerak

### 6.3.1 Tujuan Pengujian

Pengujian ini dilakukan dengan tujuan untuk mengetahui apakah *node* RSU masih bisa mengirim data ke *node* OBU yang bergerak dan melihat apakah kecepatan *node* OBU bergerak mempengaruhi proses penerimaan data oleh *node*

OBU. Perhitungan yang dilakukan sama seperti pengujian sebelumnya yaitu menghitung berapa waktu pengiriman data, packet loss, dan troughput. Pengujian ini dilakukan karena ini merupakan salah satu kebutuhkan fungsional.

### 6.3.2 Prosedur Pengujian

Pengujian dilakukan dengan cara menyalakan *node* RSU dan melakukan pengiriman data, kemudian *node* OBU yang bertugas untuk menerima data bergerak dengan kecepatan 10km/jam, 20km/jam, dan 30km/jam pada jarak 10m, 20m, 30m, 40m, dan 50m. *node* OBU bergerak secara bebas namun pergerakannya mendekat ke *node* RSU.

### 6.3.3 Pelaksanaan Pengujian

Pengujian akan dilakukan dengan menyalakan *node* RSU terlebih dahulu dan kemudian menjalankan *node* OBU, *node* RSU akan mengirimkan data sebanyak 100 kali dan *node* OBU akan menerima paket data yang dikirimkan oleh *node* RSU. Hasil dari penerimaan *node* OBU akan dilihat seberapa besar packet loss, berapa RTT, dan berapa besar throughputnya.

Skenario pada pegujian kali ini pengujian adalah sebagai berikut

1. Pengiriman paket data oleh RSU dimana *node* OBU bergerak mendekat dengan kecepatan 10km/jam
2. Pengiriman paket data oleh RSU dimana *node* OBU bergerak mendekat dengan kecepatan 20km/jam
3. Pengiriman paket data oleh RSU dimana *node* OBU bergerak mendekat dengan kecepatan 30km/jam

### 6.3.4 Hasil Pengujian

**Tabel 6. 2 Hasil Pengujian Pengiriman *Node* RSU dengan OBU bergerak 10km/jam**

| Jarak(meter) | Paket yang dikirim | Waktu Pengiriman 10km/jam(ms) | packet loss (%) | Throughput (Mbit/s) |
|---|---|---|---|---|
| 10 | 100 | 117.27 | 14.5 | 0.81 |
| 20 | 100 | 85.44 | 14 | 0.77 |
| 30 | 100 | 73.06 | 10.5 | 0.71 |
| 40 | 100 | 112.39 | 38 | 0.71 |
| 50 | 100 | 76.81 | 20 | 0.81 |
| Rata-Rata | | 92.99 | 19.40 | 0.76 |

**Tabel 6. 3 Hasil Pengujian Pengiriman *Node* RSU dengan OBU bergerak 20km/jam**

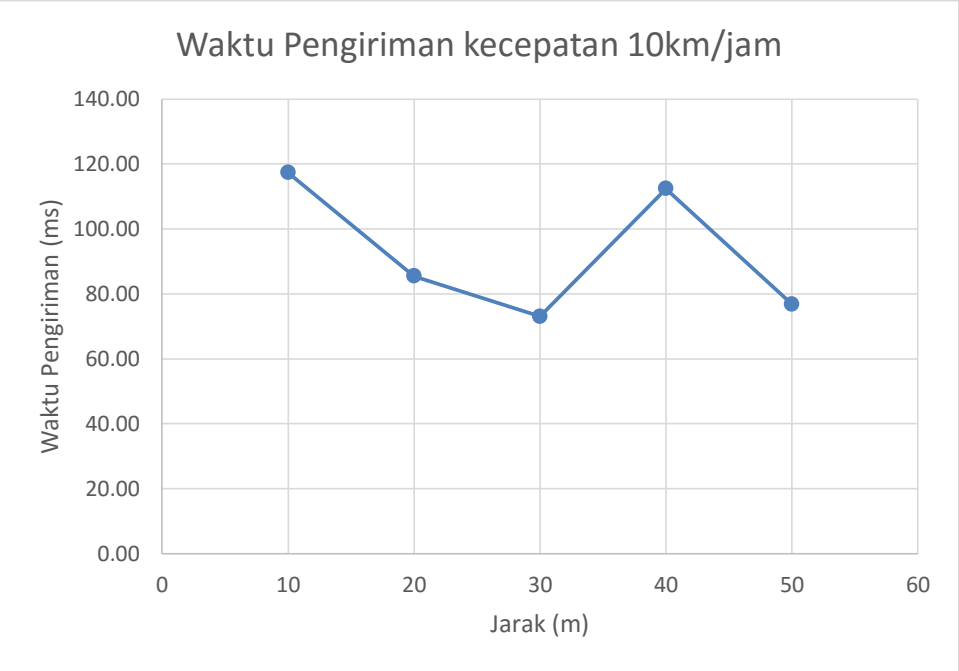
| Jarak(meter) | Paket yang dikirim | Waktu Pengiriman 20km/jam(ms) | packet loss (%) | Throughput (Mbit/s) |
|---|---|---|---|---|
| 10 | 100 | 160.15 | 18.50 | 0.77 |
| 20 | 100 | 90.42 | 11.00 | 0.67 |
| 30 | 100 | 73.92 | 10.00 | 0.77 |
| 40 | 100 | 158.95 | 30.00 | 0.71 |
| 50 | 100 | 78.05 | 20.00 | 0.81 |
| Rata-Rata | | 112.30 | 17.90 | 0.75 |

**Tabel 6. 4 Hasil Pengujian Pengiriman *Node* RSU dengan OBU bergerak 30km/jam**

| Jarak(meter) | Paket yang dikirim | Waktu Pengiriman 30km/jam(ms) | packet loss (%) | Throughput (Mbit/s) |
|---|---|---|---|---|
| 10 | 100 | 160.15 | 18.50 | 0.77 |
| 20 | 100 | 105.45 | 16.00 | 0.67 |
| 30 | 100 | 74.33 | 10.00 | 0.77 |
| 40 | 100 | 72.78 | 36.00 | 0.71 |
| 50 | 100 | 77.25 | 27.00 | 0.81 |
| Rata-Rata | | 97.99 | 21.50 | 0.75 |

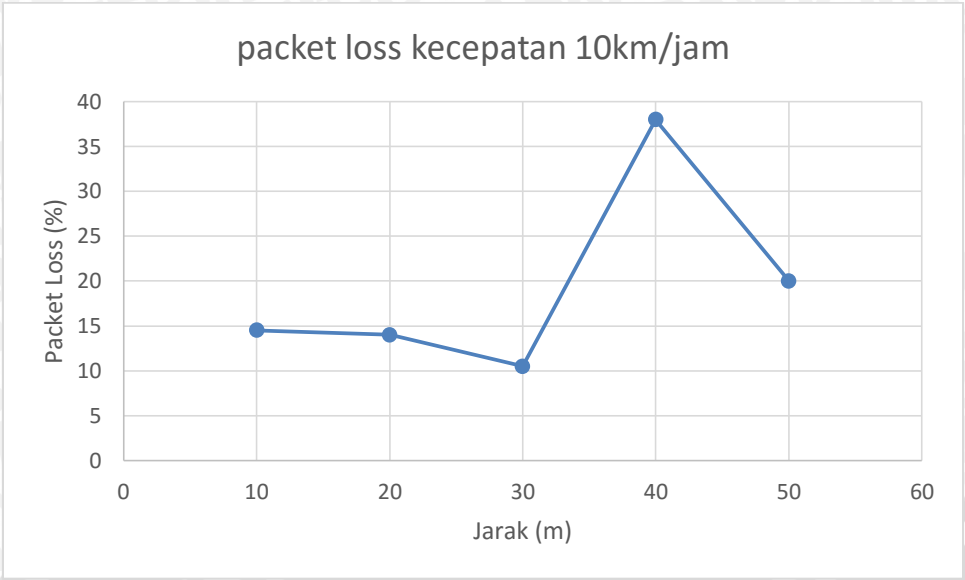
Table 6.2, table 6.3, dan table 6.4 merupakan table hasil pengujian pengiriman *node* RSU kepada *node* OBU dengan kecepatan 10km/jam, 20km/jam, dan 30km/jam. hasil yang perlu diperhatikan adalah waktu pengiriman data, besarnya *packet loss,* dan besarnya *throughput* yang perhitungannya dapat dilihat pada pengujian sebelumnya tepatnya pada persamaan 6.1 dan persamaan 6.2.

### 6.3.5 Analisa Pengujian



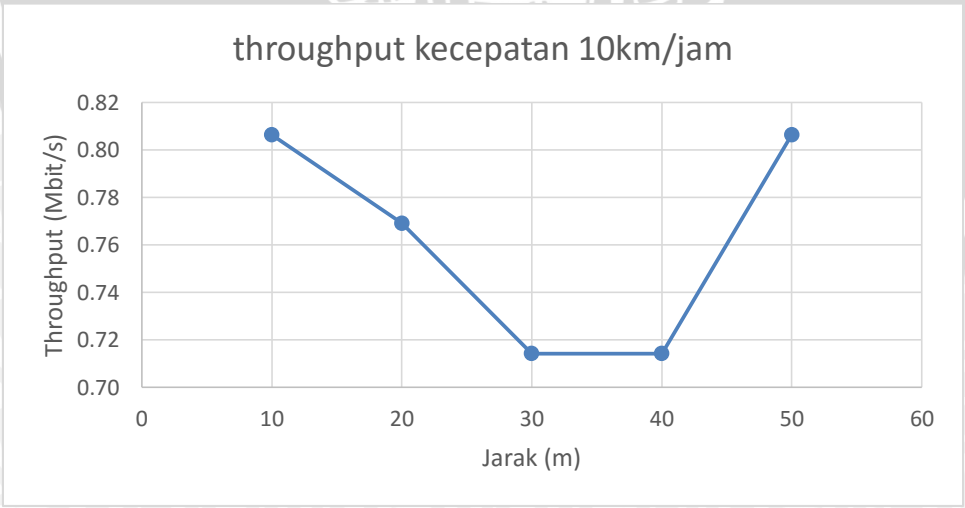

**Gambar 6. 10 Grafik Kecepatan Pengiriman RSU kecepatan 10km/jam**

Gambar 6.10 merupakan grafik RTT untuk pengiriman data yang dilakukan dapat dianalisa bahwa RTT memiliki nilai yang cukup bagus, pengujian pada jarak 10m memiliki waktu 117.27ms yang hasilnya lebih baik daripada pengujian ketika *node* OBU dalam posisi diam, hal ini terjadi karena *node* OBU bergerak mendekat yang membuat pengiriman lebih lancer dan membutuhkan waktu pengiriman data lebih sedikit. Pada jarak 20m dan 30m juga terjadi hal yang sama yaitu waktu pengiriman data lebih cepat dari pengujian dengan *node* diam dan pengujian jarak 10m karena *node* bergerak mendekat dan gangguan angin tidak terlalu besar. Seperti halnya pada percobaan sebelumnya, hasil pengujian pada jarak 40m memiliki waktu pengiriman data yang cukup lama, namun tetap waktu yang dihasilkan pada pengujian ini lebih baik daripada pengujian saat *node* diam, beigtu juga dengan hasil yang ditunjukkan pada jarak 50m. Secara keseluruhan waktu pengiriman lebih cepat karena *node* OBU bergerak mendekat pada *node* RSU.




**Gambar 6. 11 Grafik *Packet Loss* Pengiriman RSU kecepatan 10km/jam**

Grafik 6.11 merupakan grafik packet loss yang terjadi selama pengiriman data dilakukan pada kecepatan 10km/jam. Dapat dianalisa bahwa rata-rata *packet loss* pada jarak 10m adalah 14.5, meskipun waktu pengiriman lebih cepat dari pengujian saat *node* diam namun *packet loss* yang terjadi justru lebih besar hal ini terjadi karena saat *node* OBU bergerak mendekati *node* RSU posisi nRF24L01 pada *node* OBU tidak selalu menghadap nRF24L01 yang berada pada *node* RSU, selain itu juga ada gangguan dari angin seperti yang sudah dijelaskan pada pengujian saat *node* OBU diam, hal yang sama juga terjadi saat pengujian pada jarak 30m yang memiliki *packet loss* lebih tinggi daripada *packet loss* pengujian sebelumnya. Meskipun *packet loss* yang terjadi cukup besar pada jarak 40m, hasil yang ditunjukkan lebih baik daripada saat pengujian sebelumnya, sedangkan pada jarak 50m *packet loss* yang terjadi tidak berbeda dengan percobaan sebelumnya.



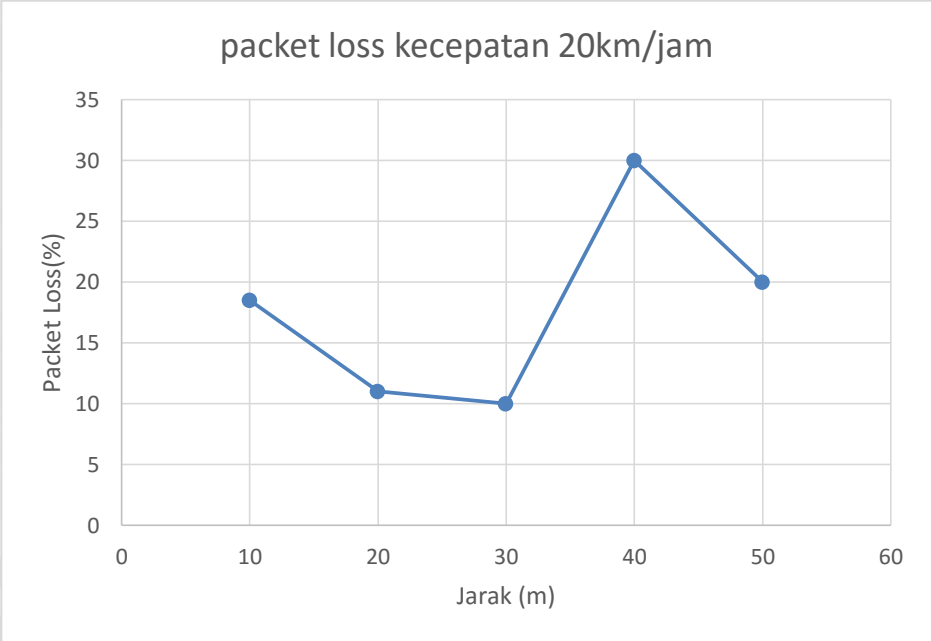

**Gambar 6. 12 Grafik *Throughput* Pengiriman RSU kecepatan 10km/jam**

Grafik 6.12 merupakan hasil perhitungan throughput pengiriman data. Dapat dianalisa bahwa *throughput* pada jarak 10m sangat baik, meskipun waktu

pengiriman data cukup lama dan memilik *packet loss* yang meningkat *throughput* yang dihasilkan ada jarak 10m tetap tinggi hal ini menjelaskan bahwa *node* RSU dan *node* OBU tetap terhubung satu sama lain meskipun memiliki beberapa gangguan dalam pengiriman data, *throughput* pada jarak 20m mengalami kenaikan dibandingkan dengan percobaan sebelumnya hal ini tidak terlepas dari *packet loss* yang terjadi pada jarak 20m yang tidak terlalu besar, berbeda dengan *throughput* pada jarak 30m yang menurun diakibatkan naiknya *packet loss* yang terjadi, pada jarak 40m *throughput* yang dihasilkan cukup bagus melihat *packet loss* yang terjadi cukup tinggi, sementara pada jarak 50m *throughput* yang dihasilkan tidak berubah karena memang *packet loss* yang terjadi pun tidak berbeda.

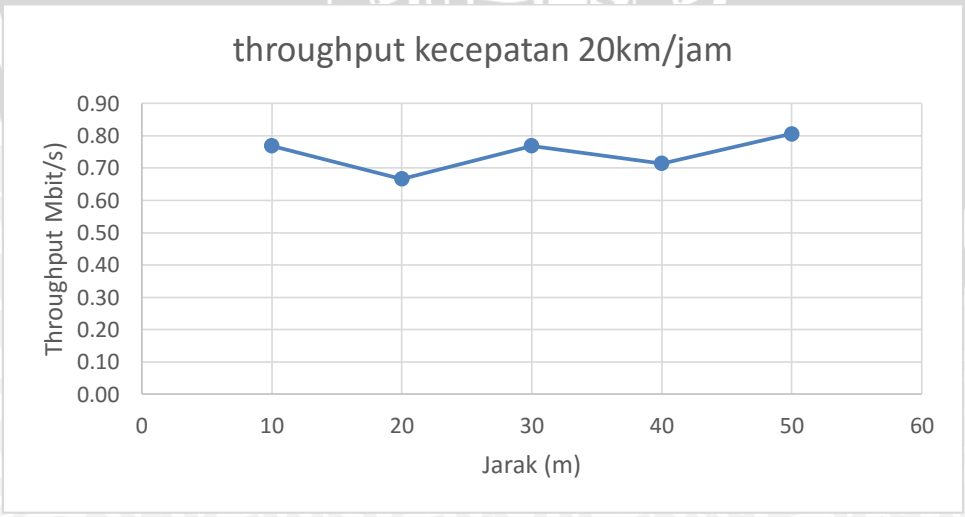

**Gambar 6. 13 Grafik Kecepatan Pengiriman RSU kecepatan 20km/jam**

Gambar 6.13 merupakan grafik hasil dari perhitungan RTT pengiriman data dengan kecepatan 20km/jam. Dapat dilihat bahwa grafik diatas tidak berbeda jauh dari grafik yang ditunjukkan pada gambar 6.6, hal ini terjadi karena memang percobaan yang dilakukan sama persis seperti yang dilakukan pada saat OBU bergerak dengan kecepatan 10km/jam hanya saja kecepatan ditingkatkan menjadi 20km/jam. Dari grafik diatas bisa dilihat bahwa waktu pengiriman data meningkat pada jarak 10m hingga 50m terutama pada jarak 10m dan 40m, hal ini terjadi dikarenakan angin yang dihasilkan pada kecepatan 20km/jam lebih besar dibandingkan dengan saat OBU bergerak dengan kecepatan 10km/jam terlebih lagi area pengujian jarak 10m memang sedikit berangin dan jarak 40m memang jarak yang cukup jauh untuk mengirimkan data.




**Gambar 6. 14 Grafik *Packet Loss* Pengiriman RSU kecepatan 20km/jam**

Grafik 6.14 merupakan hasil pengujian pengiriman *node* RSU pada *node* OBU yang bergerak dengan 20km/jam. Grafik *packet loss* juga tidak memiliki perbedaan yang signifikan dengan grafik yang ditunjukkan pada gambar 6.11, sama seperti hasil waktu pengiriman data jarak 10m dan 40m yang memiliki perbedaan cukup besar, sementara pada jarak 20m, 30m dan 50 memiliki hasil yang lebih stabil. Pada jarak 10m *packet loss* kembali naik menjadi 18.50 yang sebelumnya sudah naik dari nilai 9.5 ke 14.5, hal ini sedikit banyak dipengaruhi dari waktu pengiriman data yang cukup lama. Sementara *packet loss* pada jarak 40m justru mengalami penurunan menjadi 30 meskipun waktu pengiriman data lebih lama daripada percobaan dengan kecepatan 10km/jam.



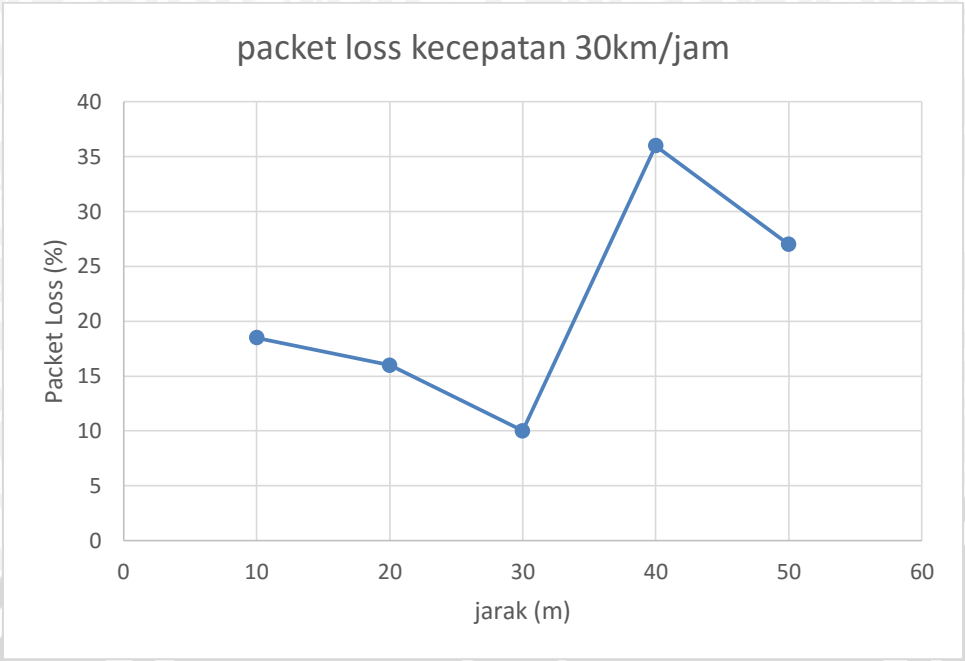

**Gambar 6. 15 Grafik *Throughput* Pengiriman RSU kecepatan 20km/jam**

Gambar 6.15 merupakan grafik *throughput* yang dihasilkan setelah percobaan pengiriman *node* RSU pada *node* OBU yang bergerak dengan keceptan

20km/jam. *Throughput* yang dihasilkan cukup stabil karena memiliki nilai yang tidak jauh berbeda dengan nilai *throughput* yang dihasilkan pada percobaan dengan kecepatan 10km/jam. Hanya saja *throughput* yang dihasilkan pada jarak 20m turun hingga 0.67 Mbit/s padahal hasil pengujian waktu pengiriman data dan *packet loss* pada jarak ini cukup bagus.

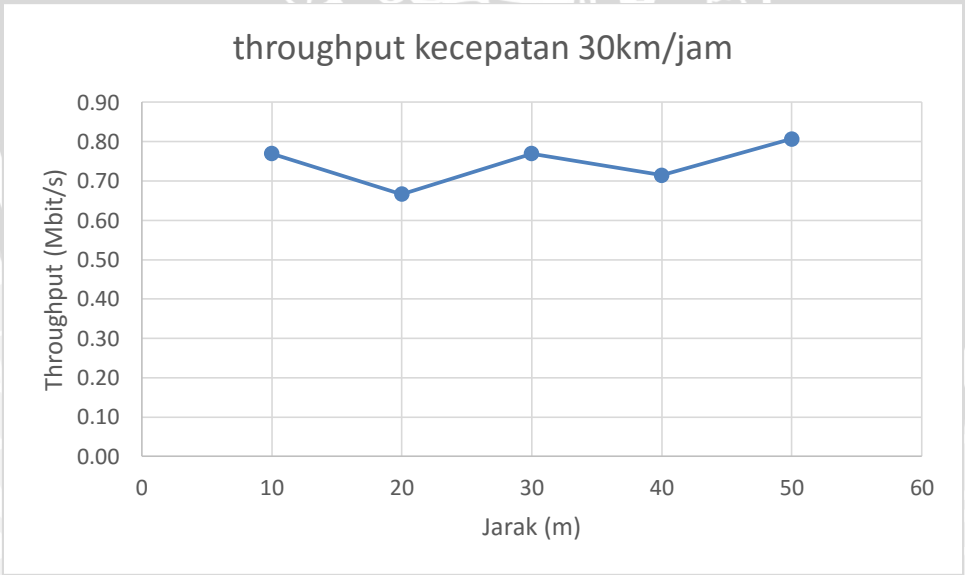

**Gambar 6. 16 Grafik Kecepatan Pengiriman RSU kecepatan 30km/jam**

Gambar 6.16 merupakan grafik dari hasil pengujian pengiriman *node* RSU pada *node* OBU yang bergerak dengan kecepatan 30km/jam. Grafik yang ditunjukkan berbeda dengan hasil kecepatan 10km/jam dan 20km/jam, pada grafik ini terlihat waktu pengiriman data pada jarak 10m memiliki waktu pengirman data yang paling lama yaitu 160.15ms yang semakin lama waktu pengiriman data semakin cepat seperti yang ditunjukkan pada jarak 20m memiliki waktu pengiriman data 105.45ms kemudian pada jarak 30m waktu pengiriman menjadi 74.33ms, pada jarak 40m waktu menjadi 72.78ms, dan pada jarak 50m 77.25ms. Terjadi anomaly pada hasil pengujian pada kecepatan 30m karena ketika jarak semakin jauh justru waktu pengiriman semakin cepat, dan waktu pengiriman pada jarak 10m memiliki waktu pengiriman yang paling lama.




**Gambar 6. 17 Grafik *Packet Loss* Pengiriman RSU kecepatan 30km/jam**

Gambar 6.17 merupakan grafik dari hasil pengujian pengiriman *node* RSU pada *node* OBU yang bergerak dengan kecepatan 30km/jam. Grafik *packet loss* tidak menunjukkan perbedaan besar seperti yang terjadi pada grafik waktu pengiriman, nilai yang dihasilkan juga tidak berbeda jauh dengan percobaan pada kecepatan 20km/jam, hanya ada sedikit kenaikan *packet loss* yang terjadi pada jarak 20m, 40m, dan 50m.




**Gambar 6. 18 Grafik *Throughput* Pengiriman RSU kecepatan 20km/jam**

Gambar 6.18 berikut merupakan grafik dari hasil pengujian pengiriman *node* RSU pada *node* OBU yang bergerak dengan kecepatan 30km/jam. Grafik *throughput* menunjukkan tidak ada perbedaan antara *throughput* pada kecepatan 20km/jam dan 30km/jam, nilai yang dimiliki dari kedua pengujian juga bernilai sama.

## 6.4 Pengujian Pengiriman Paket Data *Node* OBU

### 6.4.1 Tujuan Pengujian

Tujuan dari pengujian ini adalah untuk mengetahui apakah *node* OBU mampu melakukan pengiriman data dengan baik.

### 6.4.2 Prosedur Pengujian

Prosedur dari pengujian adalah dengan mengirimkan paket data dari *node* OBU ke *node* OBU yang lain sebanyak 100 kali sehingga *node* OBU akan melakukan pengiriman data dan penerimaan data secara bergantian satu sama lain dalam keadaan statis. Pengujian ini dilakukan dengan cara yang sama seperti 2 pengujian sebelumnya yaitu dilihat berapa waktu pengiriman data dan packet loss dari pengiriman tersebut, kemudian nanti juga akan dilakukan pengiriman terhadap besarnya troughput

### 6.4.3 Pelaksanaan Pengujian

Pelaksanaan pengujian akan dilakukan dengan cara menyalakan kedua *node* OBU dan menjalankan proses pengiriman data dan penerimaan data, sehingga kedua *node* OBU akan mengirim paket data dan langusng menerima paket data dari *node* OBU yang lain.

### 6.4.4 Hasil pengujian

**Tabel 6. 5 Hasil Pengujian Pengiriman Paket Data *Node* OBU**

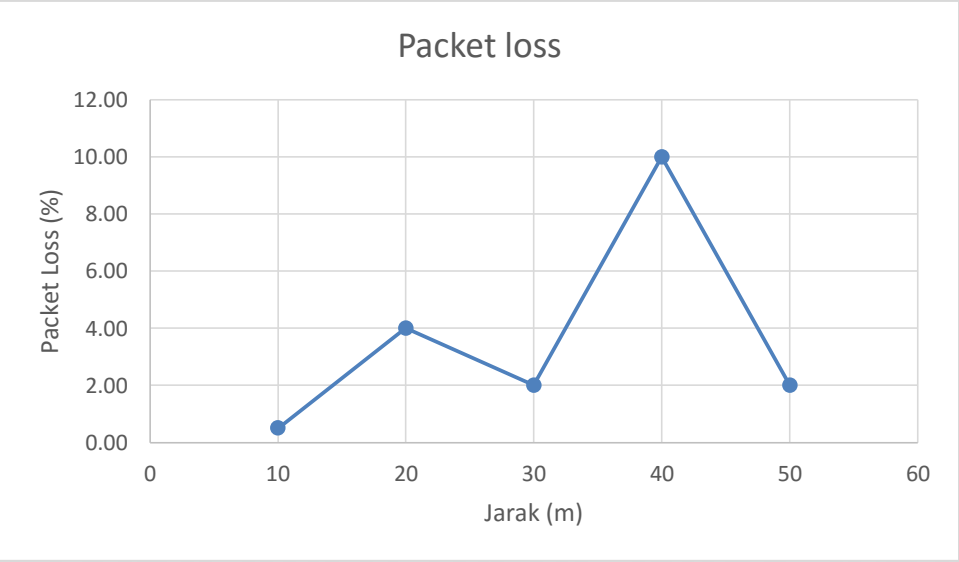
| Jarak(meter) | Paket yang dikirim | Waktu Pengiriman | packet loss (%) | Throughput (Mbit/s) |
|---|---|---|---|---|
| 10 | 100 | 149.87 | 0.50 | 2.77 |
| 20 | 100 | 301.64 | 4.00 | 1.91 |
| 30 | 100 | 174.28 | 2.00 | 1.74 |
| 40 | 100 | 491.65 | 10.00 | 1.05 |
| 50 | 100 | 129.04 | 2.00 | 4.30 |
| Rata-Rata | | 249.29 | 3.70 | 2.35 |

Tabel 6.5 merupakan hasil dari pengujian dari pengiriman *node* OBU kepada *node* OBU yang lainnya. Hasil yang percobaaan yang dilakukan adalah lamanya waktu pengiriman data yang dilakukan, besarnya *packet loss,* besarnya *throughput* yang dihasilkan. Analisa dari table 6.5 dapat dilihat pada gambar 6.13 sampai gambar 6.15

### 6.4.5 Analisa Pengujian


**Gambar 6. 19 Grafik Kecepatan Pengiriman Data *Node* OBU**

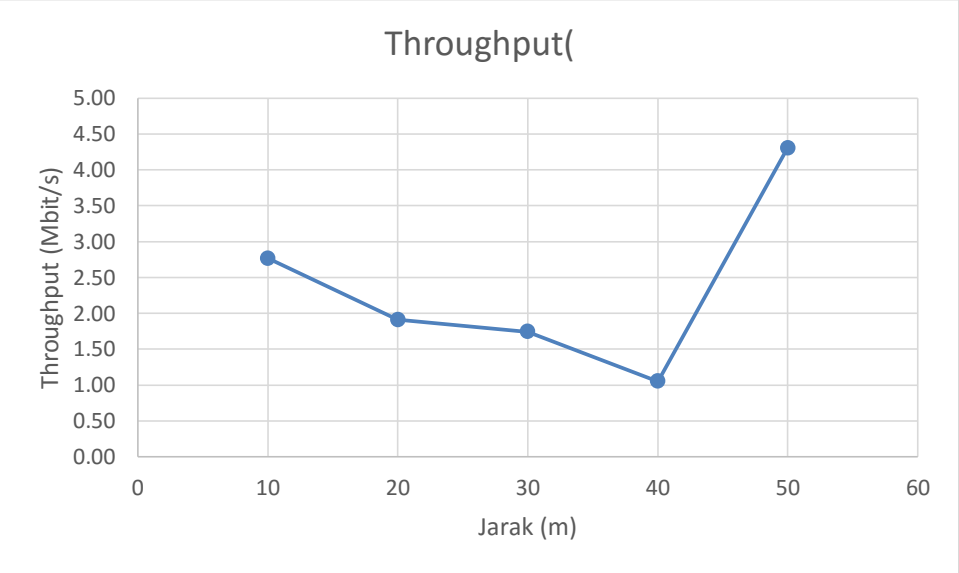
Gambar 6.19 merupakan grafik waktu pengiriman data pada pengujian pengiriman *node* OBU pada *node* OBU. Terlihat grafik menunjukkan bahwa waktu pengiriman data lebih lama dibandingkan dengan waktu pengiriman saat *node* RSU mengirimkan data, hal ini terjadi karena pengambilan data dilakukan tidak bersamaan dengan saat pengiriman *node* RSU sehingga hasil yang ditunjukkan pada grafik juga sedikit berbeda. Terlihat pada jarak 10m pengujian pengiriman *node* RSU memiliki waktu pengiriman yang cukup lama sementara pada pengujian pengiriman *node* OBU ini waktu pengiriman data pada jarak 10m adalah salah satu yang tercepat.




**Gambar 6. 20 Grafik *packet loss* Pengiriman Data *Node* OBU**

Gambar 6.20 merupakan grafik *packet loss* pada pengujian pengiriman *node* OBU pada *node* OBU. Grafik *packet loss* yang terbentuk tidak berbeda jauh dengan

grafik waktu pengiriman yang mana membuktikan bahwa waktu pengiriman data berkaitan dengan besarnya *packet loss* yang terjadi. *Packet loss* yang terjadi pada jarak 10m sangat kecil bahkan hampir tidak ada, sementara pada jarak 40m terjadi *packet loss* paling banyak yaitu 10, dari hasil *packet loss* dapat dilihat bahwa *packet loss* yang terjadi saat pengiriman *node* OBU lebih sedikit dibandingkan dengan pengiriman *node* RSU yang memiliki nilai *packet loss* hingga 44.




**Gambar 6. 21 Grafik *Throughput* Pengiriman Data *Node* OBU**

Gambar 6.21 merupakan grafik *throughput loss* pada pengujian pengiriman *node* OBU pada *node* OBU. Karena *packet loss* yang terjadi sedikit maka nilai *throughput* yang dihasilkan menjadi sangat baik, terlihat bahwa nilai *throughput* paling rendah terjadi saat berada pada jarak 40m, namun hasil *throughput* tersebut masih lebih baik daripada hasil *throughput* yang dihasilkan saat pengiriman nose RSU.

## 6.5 Pengujian Pengiriman Paket Data *Node* OBU yang bergerak

### 6.5.1 Tujuan Pengujian

Tujuan dari pengujian ini adalah untuk menlihat apakah pergerakan *node* OBU dengan kecepatan tertentu mempengaruhi proses pengiriman dan penerimaan data pada *node* OBU itu sendiri.

### 6.5.2 Prosedur Pengujian

Prosedur dari pengujian adalah dengan mengirimkan paket data dari *node* OBU ke *node* OBU yang lain sebanyak 100 kali sehingga *node* OBU akan melakukan pengiriman data dan penerimaan data secara bergantian satu sama lain dalam keadaan statis. Pengujian ini dilakukan dengan cara yang sama seperti 2 pengujian sebelumnya yaitu dilihat berapa waktu pengiriman data dan packet loss dari pengiriman tersebut, kemudian nanti juga akan dilakukan pengiriman terhadap besarnya troughput

### 6.5.3 Pelaksanaan Pengujian

Pengujian ini dilaksanakan dengan cara menyalakan kedua *node* OBU dan membiarkan kedua *node* OBU tersebut saling mengirim paket data sebanyak 200 kali. Pada pengujian ini ada beberapa skenarian yang akan dilakukan, antara lain:

1. Pengiriman paket data oleh kedua *node* OBU dimana keuda *node* OBU bergerak saling mendekat dengan kecepatan 10km/jam
2. Pengiriman paket data oleh kedua *node* OBU dimana keuda *node* OBU bergerak saling mendekat dengan kecepatan 20km/jam
3. Pengiriman paket data oleh kedua *node* OBU dimana keuda *node* OBU bergerak saling mendekat dengan kecepatan 30km/jam

### 6.5.4 Hasil Pengujian

**Tabel 6. 6 Pengujian Pengiriman Paket Data *Node* OBU yang bergerak dengan kecepatan 10km/jam**

| Jarak(meter) | Paket yang dikirim | Waktu Pengiriman 10km/jam (ms) | packet loss (%) | Throughput (Mbit/s) |
|---|---|---|---|---|
| 10 | 100 | 150.38 | 1.5 | 4.55 |
| 20 | 100 | 81.39 | 8.5 | 1.91 |
| 30 | 100 | 75.99 | 7.5 | 1.38 |
| 40 | 100 | 454.76 | 14.5 | 1.05 |
| 50 | 100 | 79.99 | 3 | 4.29 |
| Rata-Rata | | 168.50 | 7.00 | 2.64 |

**Tabel 6. 7 Pengujian Pengiriman Paket Data *Node* OBU yang bergerak dengan kecepatan 20km/jam**

| Jarak(meter) | Paket yang dikirim | Waktu Pengiriman 20km/jam (ms) | packet loss (%) | Throughput (Mbit/s) |
|---|---|---|---|---|
| 10 | 100 | 106.04 | 2.50 | 1.91 |
| 20 | 100 | 81.42 | 6.50 | 1.47 |
| 30 | 100 | 172.68 | 4.00 | 1.36 |
| 40 | 100 | 480.58 | 10.00 | 0.85 |
| 50 | 100 | 79.99 | 3 | 4.29 |
| Rata-Rata | | 126.95 | 4.50 | 1.94 |

**Tabel 6. 8 Pengujian Pengiriman Paket Data *Node* OBU yang bergerak dengan kecepatan 30km/jam**

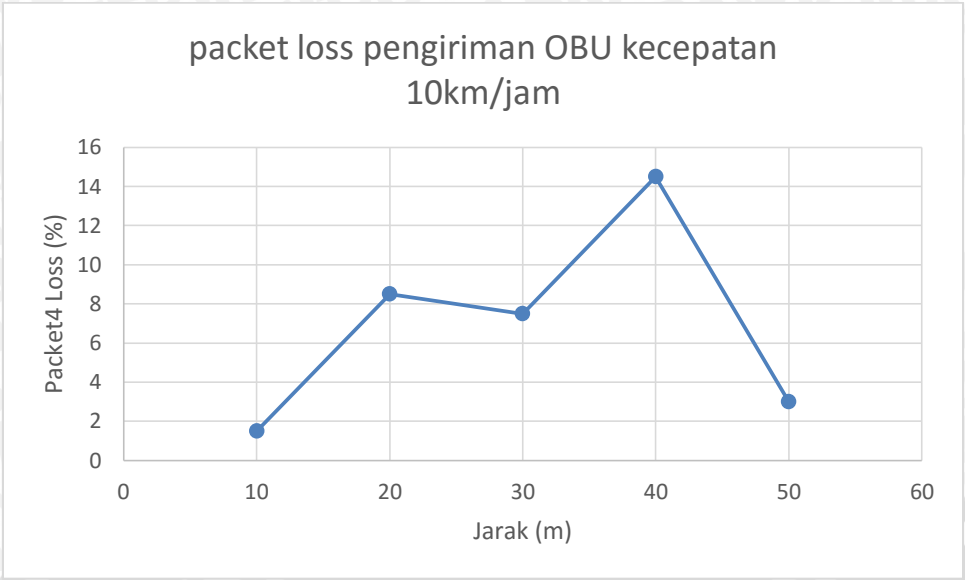
| Jarak(meter) | Paket yang dikirim | Waktu Pengiriman 20km/jam (ms) | packet loss (%) | Throughput (Mbit/s) |
|---|---|---|---|---|
| 10 | 100 | 106.99 | 1.50 | 1.91 |
| 20 | 100 | 81.42 | 8.00 | 1.64 |
| 30 | 100 | 159.54 | 6.00 | 1.22 |
| 40 | 100 | 208.19 | 13.50 | 0.82 |
| 50 | 100 | 248.06 | 4.50 | 3.08 |
| Rata-Rata | | 160.84 | 6.70 | 1.73 |

Tabel 6.6, tabel 6.7, dan tabel 6.8 merupakan hasil dari pengujian dari pengiriman *node* OBU kepada *node* OBU yang lainnya. Hasil yang percobaaan yang dilakukan adalah lamanya waktu pengiriman data yang dilakukan, besarnya *packet loss,* besarnya *throughput* yang dihasilkan. Analisa dari table 6.5 dapat dilihat pada gambar 6.16 sampai gambar 6.25

### 6.5.5 Analisa Pengujian

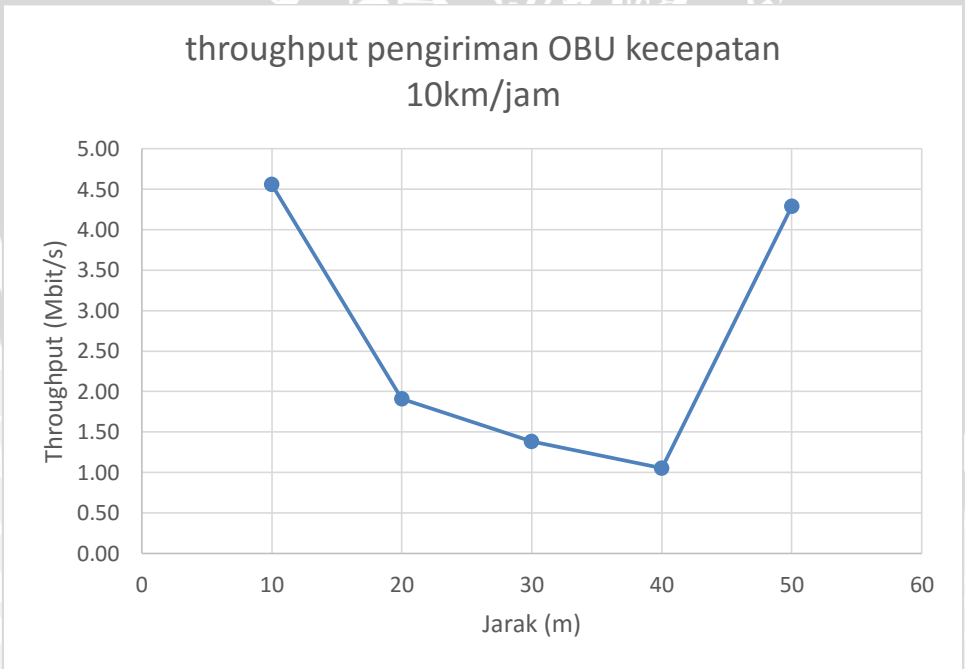

**Gambar 6. 22 Grafik Kecepatan Pengiriman OBU kecepatan 10km/jam**

Gambar 6.22 merupakan grafik waktu pengiriman data dari pengujian pengiriman *node* OBU pada *node* OBU dengan kecepatan 10km/jam. Dapat dilihat bahwa pada jarak 10m waktu pengiriman adalah 106.04ms yang sedikit lebih tinggi dibandingkan dengan jarak 20m, 30m, dan 50, hal ini seperti ini terjadi seperti saat pengiriman data *node* RSU. Pada jarak 20m, 30m, dan 50m pengiriman data lebih stabil dengan waktu pengiriman data yang berada dibawah 100ms, sementara pada jarak 40m waktu pengiriman data sangat lama hingga mencapai 454.76ms, hal ini terjadi karena factor jarak antar *node* yang gangguan angin yang ada.




**Gambar 6. 23 Grafik *packet loss* Pengiriman OBU kecepatan 10km/jam**

Gambar 6.23 merupakan grafik *pcket loss* dari pengujian pengiriman *node* OBU pada *node* OBU dengan kecepatan 10km/jam. Dapat dilihat bahwa *packet loss* yang terjadi juga memiliki nilai yang kecil, *packet loss* pada jarak 40m yang memiliki waktu pengiriman yang sangat tinggi juga tidak terlalu besar yaitu hanya 14,5. Sementara pada jarak 10m,20m,30m, dan 50m *packet loss* yang terjadi tidak lebih dari 10.



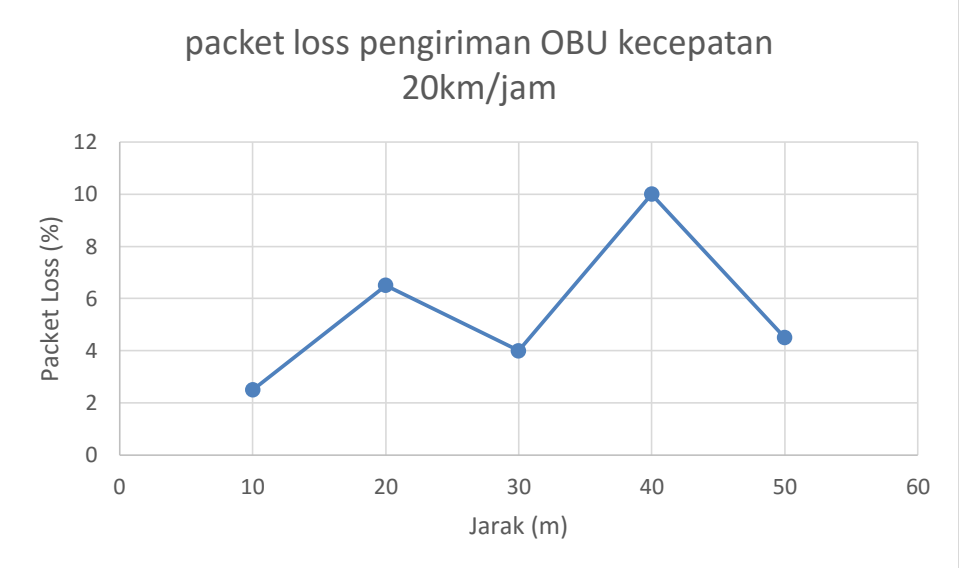

**Gambar 6. 24 Grafik *Throughput* Pengiriman OBU kecepatan 10km/jam**

Gambar 6.24 merupakan grafik *throughput* dari pengujian pengiriman *node* OBU pada *node* OBU dengan kecepatan 10km/jam. Dapat dilihat juga *throughput* yang dihasilkan sangat baik hingga mencapai 4.50Mbit/s pada jarak 10m, hal ini

terjadi karena proses pengiriman data pada *node* OBU memang sangat cepat dikarenakan delay saat pengiriman data kecil dan *packet loss* yang terjadi juga tidak terlalu besar.


**Gambar 6. 25 Grafik Kecepatan Pengiriman OBU kecepatan 20km/jam**

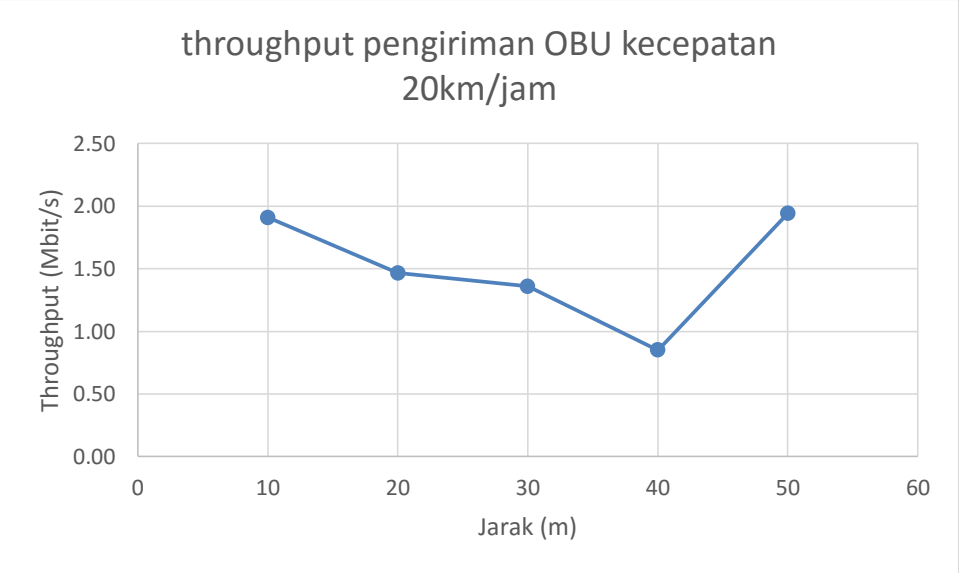
Gambar 6.25 merupakan grafik waktu pengiriman data dari pengujian pengiriman *node* OBU pada *node* OBU dengan kecepatan 20km/jam. Dapat dilihat dari grafik tidak berbeda jauh dengan hasil pengujian saat kecepatan 10km/jam, waktu pengiriman pada jarak 10m, 20m, 30m, dan 50m tidak berbeda jauh dan pada jarak 40m waktu pengiriman data tetap tinggi yang mencapai 480.58ms.




**Gambar 6. 26 Grafik *packet loss* Pengiriman OBU kecepatan 20km/jam**

Gambar 6.26 merupakan grafik *packet loss* dari pengujian pengiriman *node* OBU pada *node* OBU dengan kecepatan 20km/jam. Grafik *packet loss* juga tidak menunjukkan perbedaan yang besar dengan *packet loss* pada kecepatan 10km/jam,

pada pengujian ini memiliki hasi yang sedikit lebih baik karena *packet loss* yang terjadi tidak lebih dari 10.



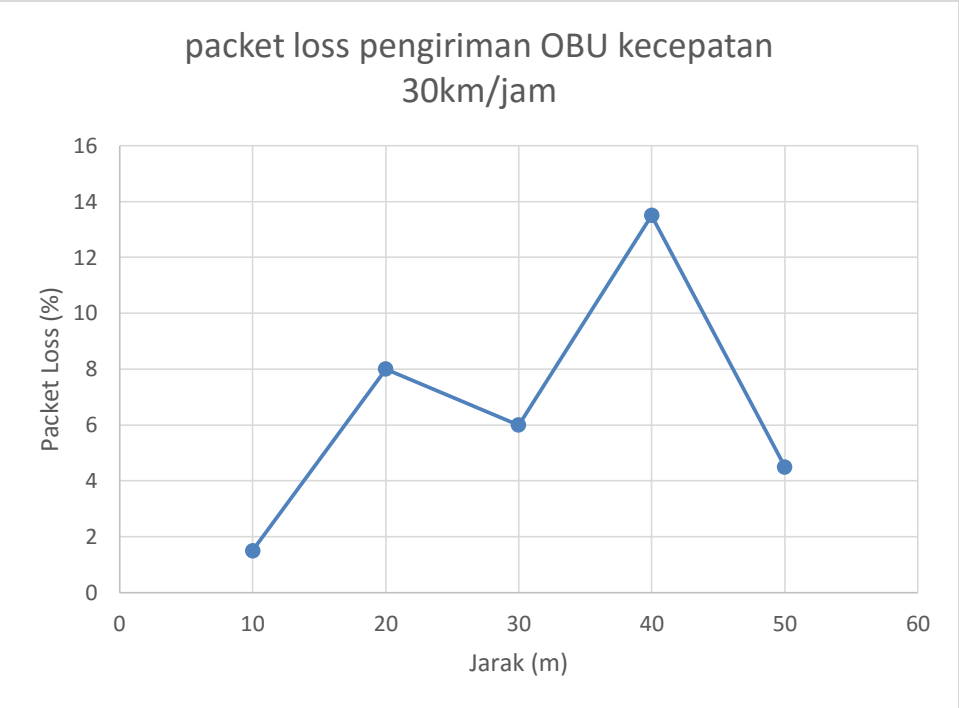

**Gambar 6. 27 Grafik *Throughput* Pengiriman OBU kecepatan 20km/jam**

Gambar 6.27 merupakan grafik *throughput* dari pengujian pengiriman *node* OBU pada *node* OBU dengan kecepatan 20km/jam. Hasil *throughput* pada pengujian ini tidak sebaik seperti pada percobaan dengan kecepatan 10km/jam, hal ini dikarenakan waktu pengiriman data yang sedikit meningkat meskipun *packet loss* bernilai kecil. Namun nilai *throughput* yang dihasilkan tetap lebih baik dari hasil pengiriman *node* RSU.


**Gambar 6. 28 Grafik Kecepatan Pengiriman OBU kecepatan 30km/jam**

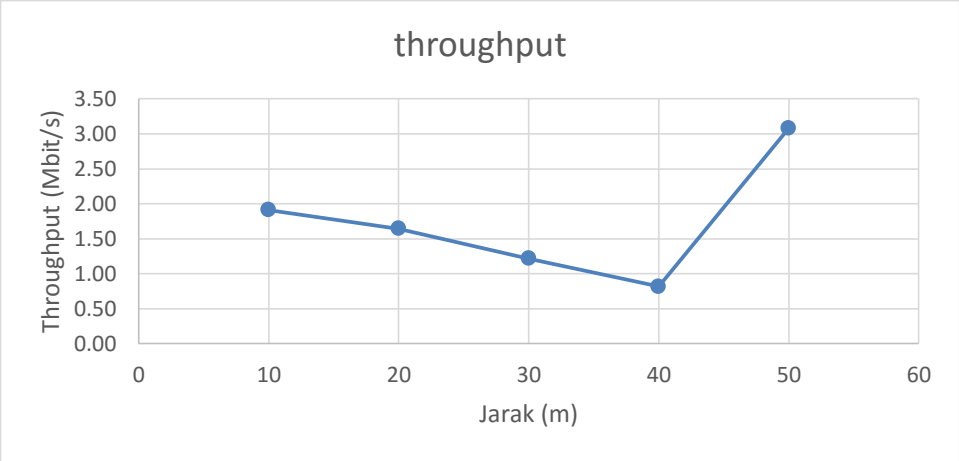
Gambar 6.28 merupakan grafik waktu pengiriman data dari pengujian pengiriman *node* OBU pada *node* OBU dengan kecepatan 30km/jam. Grafik yang

ditampilkan berbeda dari grafik yang ditunjukkan pada pengujian dengan kecepatan 10km/jam dan 20km/jam, pada grafik ini terlihat jarak sangat mempengaruhi waktu pengiriman data. Jarak 10m dan 20m memiliki waktu pengiriman 106.99ms dan 81.42 yang merupakan waktu pengiriman tercepat, setelah itu waktu pengiriman terus naik dari jarak 30m hingga 50m.




**Gambar 6. 29 Grafik *packet loss* Pengiriman OBU kecepatan 30km/jam**

Gambar 6.29 merupakan grafik *packet loss* dari pengujian pengiriman *node* OBU pada *node* OBU dengan kecepatan 30km/jam. Meskipun grafik waktu pengiriman data berbeda dengan pengujian pada kecepatan 20km/kam dan 10km/jam grafik *packet loss* yang ditunjukkan pada pengujian 30km/jam tidak berbeda jauh dengan percobaan denga kecepatan 10km/jam dan 20km/jam, hanya saja ada sedikit kenaikan pada jarak 10m, 20m, 30m, dan 40m, sementara pada jarak 50m *packet loss* yang terjadi sama.




**Gambar 6. 30 Grafik *Thrpughput* Pengiriman OBU kecepatan 10km/jam**

Gambar 6.30 merupakan grafik *throughput* dari pengujian pengiriman *node* OBU pada *node* OBU dengan kecepatan 20km/jam. Dari grafik dapat dilihat bahwa *throughput* yang dihasilkan tidak berbeda jauh dengan yang dihasilkan pada percobaan dengan kecepatan 20km/jam, namun pada jarak 50m *throughput* yang dihasilkan mencapai 3.08Mbit/s, hal ini terjadi karena meskipun waktu pengiriman data pada jarak 50m cukup tinggi namun *packet loss* yang terjadi cukup kecil sehingga *throughput* yang dihasilkan juga cukup bagus.

# BAB 7 PENUTUP

Bab ini berisi kesimpulan yang diambil dari hasil dan analisis penelitian dan juga saran untuk penelitian selanjutnya.

## 7.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi, pengujian dan analisis yang telah dilakukan, maka dapat diambil kesimpulan sebagai berikut:

1) Penerapan nRF24L01 pada VANET dilakukan dengan cara menghubungkan nRF24L01 dengan Arduino Pro Mini dan FTDI dan membuat dua *node* yang berperan sebagai RSU dan sebagai OBU.
2) Proses pengiriman data yang dilakukan oleh *node* RSU maupun proses penerimaan data yang dilakukan oleh *node* OBU memanfaatkan library RF24 dengan menggunakan *pipe address*.
3) Sesuai dengan hasil pengujian yang sudah dilakukan, jarak antar *node* sedikit mempengaruhi karena pada setiap jarak kekuatan sinyal dari NRF24L01 berbeda sehingga hasil waktu pengiriman data, *packet loss*, dan *throughput* juga berbeda.o
4) Kecepatan juga mempengaruhi proses pengiriman data pada nRF24L01 karena dengan adanya pergerakan *node* sehingga membutuhkan waktu pengiriman yang berbeda.
5) Hasil penelitian yang dilakukan dengan 4 skenario yaitu :
    a. RSU mengirim data ke OBU yang bergerak mendekat dengan jarak 50m, 40m, 30m, 20m, dan 10m
    b. RSU mengirim data ke OBU yang bergerak mendekat dengan kecepatan 10km/jam, 20km/jam, 30km/jam
    c. OBU mengirim data ke OBU yang bergerak saling mendekat dengan jarak 50m, 40m, 30m, 20m, dan 10m
    d. OBU mengirim data ke OBU yang bergerak saling mendekat dengan kecepatan 10km/jam, 20km/jam, 30km/jam.

    Adalah rata-rata waktu pengiriman saat *node* dalam keadaan diam adalah 217.92ms dan pada saat *node* bergerak adalah 137.69ms, sementara rata-rata *packet loss* saat *node* dalam keadaan diam adalah 12.30% dan pada saat *node* bergerak adalah 13.00%, sedangkan besar *throughput* saat *node* dalam keadaan diam adalah 1.55 Mbit/s dan saat *bode* bergerak adalah 1.35 Mbit/s.

## 7.2 Saran

Dari kesimpulan penelitian ini, terdapat beberapa saran yang bisa menjadi acuan dalam pengembangan penelitian selanjutnya, antara lain:

1. Perlu dilakukan penambahan antenna pada nRF24L01 agar pengiriman data bisa lebih jauh dan memiliki sinyal yang lebih kuat.
2. Perlu dibuat standart perngiriman data yang cocok untuk diterapkan pada arduino dan nRF24L01 yang memiliki memory terbatas.

# DAFTAR PUSTAKA


Bhoi, Sourav Kumar., Khilar, Pabrita Mohan. 2013. Vehicular communication: a survey. Rourkela : Department of Computer Science and Engineering, National Institute of Technology

Amudhavel, J., dkk, 2015. An Robust Recursive Ant Colony OptimizationStrategy in Vanet for Accident Avoidance (RACO-VANET). Pondicherry: International Conference on Circuit, Power and Computing Technologies [ICCPCT]

Zhang, Liren., dkk., 2013. Mobility analysis in vehicular ad hoc network (VANET). UAE. College of Information Technology, United Arab Emirates University.

GitHub, 2011, RF24 [online] tersedia di: < https://github.com/maniacbug/RF24 > [diakses pada tanggal 12 september 2015].

Al-Sultan, Saif., dkk., 2013. A comprehensive survey on vehicular Ad Hoc network. Leicester. Software Technology Research Laboratory, De Montfort University, Bede Island Building.

ASA, Nordic Semiconductor, *nRF24L01 Product Specification V2.0*,[online] tersedia di:< http://www.nordicsemi.com/eng/nordic/download_resource/8041/1/62435711 > [diakses tanggal 12 September 2015].

Arduino, *Arduino Pro Mini,* [online] tersedia di: < http://arduino.cc/en/Main/ArduinoBoardProMini > [diakses tanggal 13 September 2015].

C.C. Communication Consortium. Car 2 car communication consortium manifesto. [online] tersedia di: < Http://car-to-car.org/index.php?id=31 > [diakses pada tanggal 4 november 2015].

WHO, Number of road traffic deaths [online] tersedia di: < http://www.who.int/gho/road_safety/mortality/number_text/en/ > [diakses tanggal 17 mei 2016].

Bhardwaj, Ankit., dkk., 2012. An Efficient Energy Conserving Scheme For IEEE 802.11 Adhoc Networks. India. Punjab Engineering College

Lee, Seungbae., Alvin Lim., 2012., Reliability and Performance of IEEE 802.11n for Vehicle Networks with Multiple *Node*s. Auburn. Auburn University.

Zheng, Liming., dkk., 2012., Research on Communications over VANET under Different Scenes and Implementation of Vehicle Terminal. Guangzhou. Jinan University

Semiconductor, Dallas., DS1307 64 x 8 Serial Real-Time Clock [online] tersedia di: < https://www.sparkfun.com/datasheets/Components/DS1307.pdf > [diakses pada 5 Juni 2016].

Putra, Rizki Eka., 2016., Implementasi Purwarupa Vehicular Network Dengan Mikrokomputer Memanfaatkan Protocol 802.11 Untuk Diterapkan Pada Rambu Lalu Lintas., Malang., Universitas Brawijaya.

Bronzi., Walter., dkk., 2015., Bluetooth Low Energy performance and robustness analysis for Inter-Vehicular Communications., Luxemburg., Luxemburg University.

Christ, Peter., dkk., 2011., Performance Analysis of the nRF24L01 Ultra-Low-Power Transceiver in a Multi-Transmitter and Multi-Receiver Scenario., Bielfield., CITEC Bielfield University.