TUGAS AKHIR – KI14150

RANCANG BANGUN SISTEM KOMUNIKASI *MOBILE PEER-TO-PEER* BERBASIS ANDROID DAN NRF24L01 DENGAN MENGIMPLEMENTASIKAN KONSEP *DELAY TOLERANT NETWORK* SERTA ENKRIPSI

DELY TEJA MUKTI
NRP 05111540000003

Dosen Pembimbing I
Waskitho Wibisono, S.Kom., M.Eng.,Ph.D.

Dosen Pembimbing II
Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

TUGAS AKHIR – KI14150

RANCANG BANGUN SISTEM KOMUNIKASI *MOBILE PEER-TO-PEER* BERBASIS ANDROID DAN NRF24L01 DENGAN MENGIMPLEMENTASIKAN KONSEP *DELAY TOLERANT NETWORK* SERTA ENKRIPSI

DELY TEJA MUKTI
NRP 05111540000003

Dosen Pembimbing I
Waskitho Wibisono, S.Kom., M.Eng.,Ph.D.

Dosen Pembimbing II
Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

(Halaman ini sengaja dikosongkan)

UNDERGRADUATE THESIS – KI14150

DEVELOPMENT OF MOBILE PEER-TO-PEER COMMUNICATION BASED ON ANDROID AND NRF24L01 WITH DELAY TOLERANT NETWORK AND ENCRYPTION IMPLEMENTATION

DELY TEJA MUKTI
NRP 05111540000003

Supervisor I
Waskitho Wibisono, S.Kom., M.Eng.,Ph.D.

Supervisor II
Royyana Muslim Ijtihadie, S.Kom., M.Kom., PhD.

DEPARTMENT OF INFORMATICS
Faculty of Information and Communication Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2019

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

RANCANG BANGUN SISTEM KOMUNIKASI *MOBILE PEER-TO-PEER* BERBASIS ANDROID DAN NRF24L01 DENGAN MENGIMPLEMENTASIKAN KONSEP *DELAY TOLERANT NETWORK* SERTA ENKRIPSI

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Rumpun Mata Kuliah Komputasi Berbasis Jaringan
Sistem Studi S-1 Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh
Dely Teja Mukti
NRP. 05111540000003

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Waskitho Wibisono, S.Kom., M.Eng., Ph.D.
NIP: 19741022 200003 1 001 ..... (pembimbing 1)

Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D.
NIP: 19770824 200604 1 001 ..... (pembimbing 2)

SURABAYA
JANUARI, 2019

(Halaman ini sengaja dikosongkan)

RANCANG BANGUN SISTEM KOMUNIKASI *MOBILE PEER-TO-PEER* BERBASIS ANDROID DAN NRF24L01 DENGAN MENGIMPLEMENTASIKAN KONSEP *DELAY TOLERANT NETWORK* SERTA ENKRIPSI

Nama	**: Dely Teja Mukti**
NRP	**: 05111540000003**
Departemen	**: Informatika - FTIK**
Dosen Pembimbing I	**: Waskitho Wibisono, S.Kom., M.Eng.,Ph.D.**
Dosen Pembimbing II	**: Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D.**

Abstrak

Komunikasi merupakan salah satu kebutuhan dasar manusia. Dimanapun dan kapanpun manusia membutuhkan komunikasi untuk saling berinteraksi. Namun terkadang terdapat suatu keadaan seseorang sulit berkomunikasi dengan orang lain karena sedang berada pada daerah yang memiliki keterbatasan jaringan seluler. Dengan sebab itulah diperlukan sebuah sistem komunikasi yang independen yang dapat digunakan oleh seseorang ketika berada di daerah yang memiliki keterbatasan jaringan seluler.

Untuk menghadapi permasalahan tersebut, dalam tugas akhir ini telah diimplementasikan komunikasi mobile peer to peer dengan memanfaatkan Arduino dan modul nRF24L01 sebagai media transmisi antar node satu dengan node yang lainnya. Modul nRF24L01 merupakan sebuah modul komunikasi yang dapat digunakan sebagai alat transmisi pada jaringan mobile peer to peer. Untuk menjamin reliabelitas dari komunikasi tersebut telah diimplementasikan konsep Delay Tolerant Network. Dari sisi keamanan komunikasi, telah diimplementasikan algoritma simple encryption.

Sistem komunikasi mobile peer to peer ini telah dilakukan uji coba untuk menguji fungsionalitas dan performanya. Jarak efektif antar communication node untuk pengiriman pesan dalam sistem komunikasi ini hingga 20 meter. Ketika jarak antar communication node 30 meter tingkat keberhasilan pengiriman pesan sebesar 80% dan semakin mengecil seiring bertambahnya jarak antar node.

Kata kunci : ***Delay Tolerant Network, Peer-to-peer***

PEER-TO-PEER MOBILE COMMUNICATION SYSTEM BASED ANDROID AND NRF24L01 BY IMPLEMENTING DELAY TOLERANT NETWORK CONCEPT AND ENCRYPTION

Student Name : Dely Teja Mukti
Id Student : 05111540000003
Major : Informatics - FTIK
Supervisor I : Waskitho Wibisono, S.Kom., M.Eng., Ph.D.
Supervisor II : Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D.

Abstract

Communication is one of the basic human needs. Wherever and whenever humans need communication to interact with each other. But sometimes there is a condition that human difficult to interact because they are in an area which has limited cellular networks. That is why an independent communication system which can be used by someone when in an area that has cellular network limitations is needed.

To solve that problem, in this final project, the mobile peer to peer communication has been implemented by utilizing Arduino and nRF24L01 modules as transmission media between one node and the other nodes. The nRF24L01 module is a communication module that can be used as a transmission tool on peer to peer mobile networks. To ensure reliable communication, the Delay Tolerant Network concept has been implemented. From the security aspect, a simple encryption algorithm has been implemented.

This peer to peer mobile communication system has been tested to ensure its functionality and performance. The

Effective distance between one communication nodes and another communication node for sending messages in this communication system up to 20 meters. At a distance of 30 meters between one communication to another communication nodes the success rate of sending messages is 80% and decreases with increasing distance between communication nodes.

Keywords : ***Delay Tolerant Network, Peer-to-peer***

KATA PENGANTAR

بسم الله الرحمن الرحيم

Puji syukur kehadirat Allah *Subhanahu wa Ta'ala* dengan rahmat dan kemudahan yang diberikan oleh-Nya, penulis dapat menyelesaikan tugas akhir dengan judul "**Rancang Bangun Sistem Komunikasi *Mobile Peer-To-Peer* Berbasis Android Dan NRF24L01 Dengan Mengimplementasikan Konsep *Delay Tolerant Network* Serta Enkripsi** " sesui dengan waktu yang telah ditentukan.

Pengerjaan tugas akhir ini menjadi sebuah sarana bagi penulis untuk memperdalam ilmu yang telah didapatkan di Departemen Informatika Institut Teknologi Sepuluh Nopember. Penulis menyampaikan terima kasih kepada:

1. Allah Subhana wa Ta'ala yang telah melimpahkan rahmat, hidayah dan kemudahan kepada penulis untuk menyelesaikan Tugas Akhir ini.
2. Nabi Muhamad Shallallahu 'alaihi wa sallam yang telah menunjukkan jalan hidup yang benar.
3. Orang tua penulis, kepada Ibu yang telah mendoakan panulis agar selalu diberikan kemudahan oleh-Nya dalam menghadapi dinamika dunia perkuliahan dan dalam pengerjaan tugas akhir ini, serta Bapak yang telah memberikan dukungan moral maupun material kepada penulis.
4. Bapak Waskitho Wibisono, S.Kom, M.Eng, Ph.D. dan Bapak Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D. sebagai dosen pembimbing yang telah meluangkan waktunya memberikan arahan serta bimbingan dalam pengerjaan tugas akhir ini.
5. Bapak dan Ibu dosen Departemen Informatika ITS yang telah membimbing dan mengajarkan ilmunya kepada penulis
6. Seluruh elemen Ma'had Thaybah Surabaya, khusunya Ustadz Muhammad Nur Yasin, Lc. yang selalu mendoakan,

membimbing dan menasehati penulis untuk menjadi seorang pemuda yang tumbuh dalam ketaatan terhadap syariat islam.
7. Seluruh teman-teman angkatan 2015 dan seluruh mahsiswa Departemen Informatika ITS khusunya para admin Laboratorium *Net-Centric Computing* yang telah memberikan dukungan dan bantuan selama penulis menempuh pendidikan S1 di Departemen Informatika ITS.
8. Pihak-pihak yang tidak sempat penulis sebutkan yang telah membantu dalam pengerjaan Tugas Akhir ini.

Buku ini tidaklah sempurna, oleh sebab itulah penulis mohon maaf atas segala kesalahan dalam penyusunan buku ini, serta penulis mengharapkan saran serta kritik yang membangun dari pembaca.

Surabaya, Januari 2019

Dely Teja Mukti

DAFTAR ISI

DAFTAR GAMBAR

(Halaman ini sengaja dikosongkan)

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

KODE SUMBER

BAB I
PENDAHULUAN

Bab ini menjelaskan hal-hal yang mendasari pengerjaan Tugas Akhir, meliputi latar belakang, tujuan, rumusan dan batasan masalah, metodologi pengerjaan Tugas Akhir, dan sistematika penulisan Tugas Akhir.

1.1 Latar Belakang

Komunikasi merupakan salah satu kebutuhan dasar manusia. Dimanapun dan kapanpun manusia membutuhkan komunikasi untuk saling berinteraksi. Namun terkadang terdapat suatu keadaan dimana manusia sulit berinteraksi karena keterbatasan media untuk berkomunikasi. Salah satu contohnya ialah ketika seseorang berada di daerah yang memiliki keterbatasan jaringan seluer. Dalam kondisi ini, seseorang sulit berinteraksi dengan orang lainnya.

Komunikasi *peer to peer* (P2P) merupakan salah satu model jaringan komputer yang terdiri dari dua atau beberapa komputer, dimana setiap komputer yang terdapat di dalam jaringan berperan sebagai *client* dan juga *server*.

Modul Wireless nRF24L01 adalah sebuah modul komunikasi yang memanfaatkan pita gelombang RF 2.4 GHz ISM (*Industrial Scientific and Medical*) dan menggunakan antarmuka SPI (*Serial Peripheral Interface*) untuk berkomunikasi. Modul ini didesain untuk jaringan nirkabel yang membutuhkan daya rendah. Selain itu, modul nRF24L01 mempu menjangkau jarak hingga satu kilo meter. Sehingga modul ini cocok digunakan pada *node-node* dalam komunikasi *peer to peer*. Dengan keterbatasan jaringan komunikasi seluler itulah diperlukan sebuah skema komunikasi yang independen yang dapat digunakan oleh seseorang ketika berada di daerah yang memiliki keterbatsan jaringan seluler. Skema komunikasi tersebut dapat berupa komunikasi *peer to peer* dengan

memanfaatkan Arduino dan modul nRF24L01 sebagai media transmisi antar *node* satu dengan *node* yang lainnya. Untuk menjamin reliabelitas dari komunikasi tersebut, diimplementasikan konsep *Delay Tolerant Netwok*. Dari sisi kemanan komunikasi telah diimplementasikan algoritma *simple encryption*. Pemilihan algoritma *simple encryption* dikarenakan adanya keterbatasan jumlah karakter pada setiap pengiriman pesan pada modul nRF24L01, yaitu 32 karakter.

Dengan skema komunikasi *peer to peer* yang mengimplementasikan *Delay Tolerant Network* serta enkripsi sederhana seperti yang dijelaskan diatas, diharapkan seseorang yang sedang berada di daerah yang memiliki keterbatasan jaringan seluler dapat perkomunikasi dengan orang lain secara reliabel, aman dan tidak tergantung dengan adanya jaringan seluler di daerah tersebut

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam Tugas Akhir ini adalah sebagai berikut:

1. Bagaimana menerapkan *Delay Tolerant Network* pada komunikasi *mobile peer to peer* menggunakan modul nRF24L01?
2. Bagaimana menerapkan algoritma *simple encryption* pada komunikasi *mobile peer to peer* menggunakan modul nRF24L01?
3. Berapakah jarak efektif antar *communication node* untuk pengiriman pesan pada komunikasi *mobile peer to peer* menggunakan modul nRF24L01?
4. Berapa banyak pesan dengan enkripsi per detik yang mampu diolah oleh *communication node* pada komunikasi *mobile peer to peer* menggunakan modul nRF24L01?

1.3 Batasan Permasalahan

Permasalahan yang dibahas dalam Tugas Akhir ini memiliki beberapa batasan, di antaranya sebagai berikut:

1. Menggunakan Arduino Uno
2. Menggunakan modul nRF24L01
3. Menggunakan *Delay Tolerant Network*
4. Menggunakan Algortima *simple encryption*
5. Menggunakan perangkat *mobile* dengan sistem operasi Android.

1.4 Tujuan

Tujuan dari pembuatan tugas akhir ini adalah untuk merancang sebuah sistem komunikasi *mobile peer to peer* dengan jangkauan yang luas menggunakan modul nRF24L01 yang reliabel dan aman.

1.5 Manfaat

Tugas akhir ini diharapkan dapat memberikan manfaat kepada seseorang ketika berada di daerah yang memiliki keterbatasan jaringan seluler untuk berkomunikasi dengan orang lain secara reliabel dan aman.

1.6 Metodologi

Langkah-langkah yang ditempuh dalam pengerjaan Tugas Akhir ini yaitu sebagai berikut:

1.6.1 Penyusunan Proposal

Proposal Tugas Akhir ini berisi tentang diskripsi pendahuluan dari Tugas Akhir yang akan dikerjakan. Secara detail, proposal tugas akhir ini berisi tentang beberapa bagian, yaitu latar belakang diajukannya Tugas Akhir ini, rumusan masalah yang diangkat, batasan masalah untuk Tugas Akhir, tujuan dari pembuatan Tugas Akhir, dan manfaat dari Tugas Akhir. Selain itu, dijabarkan pula tinjauan pustaka yang digunakan sebagai referensi pendukung pembuatan tugas akhir

dan ringkasan isi yang membahas metode yang akan digunakan dalam tugas akhir. Sub bab metodologi merupakan penjelasan mengenai tahapan penyusunan tugas akhir. Terdapat pula sub bab jadwal pengerjaan yang menjelaskan jadwal pengerjaan tugas akhir dan di akhir bagian terdapat daftar pustaka untuk mencantumkan referensi yang digunakan dalam tugas akhir.

1.6.2 Studi Literatur

Pada tahap ini dilakukan pencarian informasi dan studi literatur sejumlah referensi tentang implementasi *Delay Tolerant Network* pada komunikasi *mobile peer to peer* menggunakan nRF24L01 sebagai media transmisi. Informasi dan studi literatur tersebut berasal dari buku, internet, serta materi perkuliahan yang berhubungan dengan motede yang digunakan.

1.6.3 Analisis dan Desain Perangkat Lunak

Tahap ini meliputi perancangan sistem berdasarkan studi literatur dan pembelajaran konsep teknologi dari perangkat lunak yang ada. Tahap ini mendefinisikan alur implementasi dan langkah-langkah yang akan dikerjakan. Pada tahapan ini dilakukan desain sistem dan desain proses-proses yang ada. Tahapan dari analisis dan desain perangkat lunak yang akan dilakukan dengan tahapan sebagai berikut:

1. Pemilihan arduino, modul nRF24L01 dan modul Bluetooth HC-06 yang akan digunakan.
2. Melakukan uji coba komunikasi antara perangkat android dengan arduino menggunakan modul Bluetooth.
3. Melakukan uji coba transmisi dari arduino satu dengan arduino yang lainnya mengguanakan modul nRF24L01.
4. Implementasi *delay tolerant network* pada rangkaian yang sudah dibuat.
5. Implementasi algoritma enkripsi pada rangkaian yang telah diimplementasikan *delay tolerant network.*

1.6.4 Implementasi Perangkat Lunak

Pembangunan aplikasi akan dilakukan menggunakan Bahasa pemrograman C++ pada arduino IDE. Adapun pembangunan aplikasi pada perangkat android akan menggunakan Bahasa pemrograman Java menggunakan Android Studio IDE.

1.6.5 Pengujian dan Evaluasi

Pengujian akan dilakukan dengan menguji coba fungsionalitas dan performa dari proses pengiriman pesan dari perangkat android satu ke perangkat android yang lainnya sesuai dengan skenario yang telah ditentukan. Hasil evaluasi menjabarkan tentang rangkuman hasil pengujian yang telah dilakukan. Pengujian fungsionalitas meliputi uji coba setiap bagian perangkat keras yang dirangkai pada arduino dan juga uji coba keseluruhan sistem. Pengujian performa dapat dilakukan dengan pengujian dengan penambahan jumlah *communication node*, memvariasikan jarak antar *communication node,* pengiriman pesan pada *communication node* dalam jumlah yang besar

1.6.6 Penyusunan Buku

Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas akhir serta hasil dari implementasi aplikasi perangkat lunak yang telah dibuat.

1.7 Sistematika Penulisan Buku

Buku Tugas Akhir ini bertujan untuk mendapatkan gambaran dari pengerjaan Tugas Akhir. Selain itu, diharapkan dapat berguna bagi pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku Tugas Akhir ini terdiri atas beberapa bagian. Antara lain :

Bab I Pendahuluan

Bab ini berisi latar belakang masalah, tujan dan manfaat dari Tugas Akhir, permasalahan, batasan masalah,

metodologi yang digunakan, dan sistematika penyusunan Tugas Akhir.

Bab II Tinjauan Pustaka

Bab ini membahas beberapa teori penunjang yang berhubungan dengan pokok pembahasan dan menjadi dasar dari pembuatan Tugas Akhir.

Bab III Desain dan Perancangan

Bab ini membahas desain dari jaringan yang akan dibuat meliputi arsitektur dan proses perangkat lunak serta perangkat keras.

Bab IV Implementasi

Bab ini berisi implementasi dari perencanaan perangkat lunak yang telah dibuat pada bab sebelumnya. Implementasi berupa *pseudocode* dari fungsi utama dan screenshot perangkat lunak.

Bab V Hasil Uji Coba dan Evaluasi

Bab ini membahas uji coba dari jaringan yang dibuat dengan melihat keluaran yang dihasilkan, analisa dan evaluasi untuk mengetahui kemampuan jaringan

Bab VI Kesimpulan dan Saran

Bab ini merupakan bab yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan, masalah-masalah yang dialami pada proses pengerjaan Tugas Akhir, dan saran untuk pengembangan solusi kedepannya.

Daftar Pusataka

Merupakan daftar referensi yang digunakan untuk mengembangkan Tugas Akhir.

Lampiran

Lapiran yang ada berisi kelengkapan-kelengkapan yang diperlukan dalam menyusun buku Tugas Akhir.

BAB II
TINJAUAN PUSTAKA

2.1 Komunikasi *Peer to Peer*

Komunikasi *peer to peer* (P2P) merupakan salah satu model jaringan komputer yang terdiri dari dua atau beberapa komputer, dimana setiap komputer yang terdapat di dalam jaringan berperan sebagai *client* dan juga *server*. Hal ini sangat berbeda dengan sistem *client-server*. Jaringan *peer to peer* tidak memiliki sistem kontrol yang terpusat. [1]

Dalam sistem komunikasi *peer to peer*, yang diutamakan adalah *sharing resource* dan *service*, seperti penggunaan program, data dan printer secara bersama-sama. Misalnya pemakai komputer A dapat memakai program yang dipasang di komputer B, dan pengguna A dan pengguna B dapat mencetak ke printer yang sama pada saat yang bersamaan.

2.2 *Mobile Ad-Hoc Networks* (MANET)

Mobile Ad-Hoc network (MANET) adalah suatu jaringan tanpa infrastruktur pada perangkat *mobile/node* yang terhubung dengan *link* nirkabel dan membentuk jaringan bersifat sementara. setiap perangkat *mobile* pada MANET dapat bergerak bebas secara acak. oleh karena itu, topologi jaringan pada MANET berubah secara dinamis. setiap *node* pada MANET dapat berkomunikasi dengan *node* lainnya ketika berada dalam satu cakupan jaringan komunikasi.

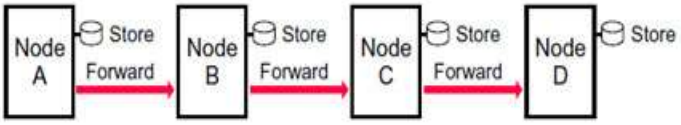
MANET cocok diimplementasikan pada keadaan darurat seperti dalam suatu daerah yang tertimpa bencana dimana terdapat kebutuhan untuk membangun jaringan komunikasi tanpa infrastruktur yang bersifat sementara. [2]

2.3 *Delay Tolerant Network* (DTN)

Delay Tolerant Network merupakan sebuah metode membangun arsitektur jaringan yang berupaya mengatasi

masalah teknis dalam jaringan heterogen yang tidak ada koneksi *end to end* atau tidak ada konektivitas jaringan secara permanen. Karena tidak adanya koneksi *end to end,* sangat besar kemungkinan terjadi *paket lost* atau pesan yang dikirim oleh *transceiver* tidak diterima oleh *receiver*. Contoh jaringan tersebut adalah yang beroperasi di jaringan seluler atau lingkungan terrestrial yang ekstrim. DTN juga telah diimplementasikan pada jaringan di ruang angkasa.[3]

Berbeda dengan sistem jaringan pada umumnya, DTN menerapkan sistem *store* dan *forward* dimana ketika suatu *node* dalam DTN menerima paket pesan, paket tersebut tidak langsung dikirim ke *node* yang lain. Melainkan ketika suatu *node* yang telah menerima paket pesan, maka paket pesan tersebut disimpan dalam media penyimpanan masing-masing *node*. Ketika terdapat keneksi yang reliabel ke *node* yang lain, maka paket pesan yang telah tersimpan dalam media penyimpanan di kirim ke *node* yang lain. Konsep *store* dan *forward* pada DTN terdapat pada gambar 2.1

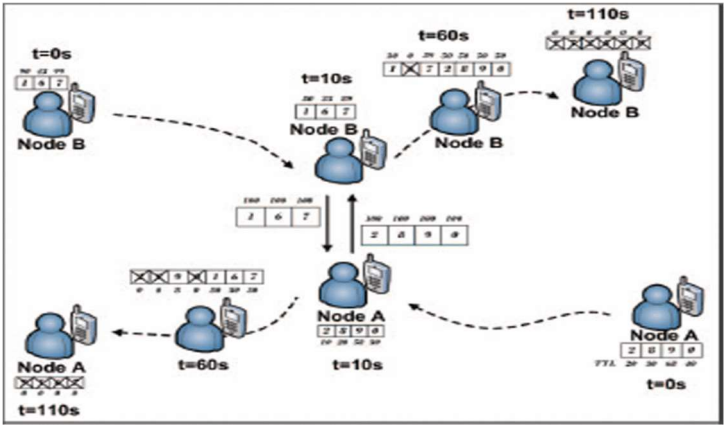


Gambar 2.1 *Store* dan *Forward* Pada *Delay Tolerant Network* [4]

Terdapat beberapa *routing protocol* yang telah diimplementasikan pada DTN salah satunya adalah *pure epidemic routing protocol*. Cara kerja dari *pure epidemic routing protocol* adalah suatu *node* akan menduplikasi pesan dan melakukan *broadcast* pesan ketika *node* tersebut mengetahui bahwa ada *node* lain disekitarnya. Terdapat kekurangan dan kelebihan mengenai *pure epidemic routing protocol*, salah setu kelebihan dari *pure epidemic routing protocol* ini adalah memiliki waktu *delay*

pengiriman yang lebih singkat daripada beberapa *routing protocol* pada DTN lainnya. Namun terdapat beberapa kekurangan dari *pure epidemic routing protocol*, yaitu pada *pure epidemic routing protocol* terdapat kepadatan jaringan yang tinggi karena jaringan dipenuhi dengan tukar menukar paket antar *node* dan *pure epidemic routing protocol* membutuhkan *resource* penyimpanan yang besar. Karena beberapa kekurangan dari *pure epidemic routing protocol* pada DTN, munculah beberapa optimasi pada *pure epidemic routing protocol* diantaranya adalah

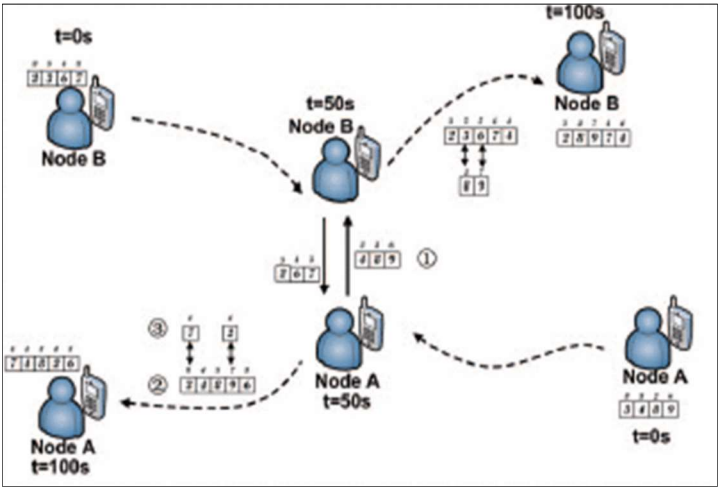
1. **Epidemic *with Time-to-Live***
   Epidemic with Time-to-Live merupakan optimasi dari *pure epidemic routing protocol* dengan memberikan *lifetime* pada masing masing pesan. Setiap pesan yang memiliki *lifetime* akan selalu di *broadcast* oleh *node* yang menerimanya sampai *lifetime* pesan tersebut habis. *Epidemic with time to live* dapat mengurangi kepadatan jaringan karena pesan akan di *broadcast* ke *node* lain sampai batas *lifetime* pesan habis. *Epidemic with Time-to-Live* ditunjukkan pada gambar 2.2



Gambar 2.2 *Epidemic with Time to Live* [5]

2. *Epidemic with Encounter Count*

Epidemic with Encounter Count merupakan optimasi dari *pure epidemic routing protocol* dengan memberikan *max encounter count* pada masing-masing masing pesan. Pemberian *max encounter count* yang dimaksud ialah memberikan batas maksimal pesan yang bersangkutan sampai pada *node* lain. Encounter count akan ditambahkan satu ketika pesan tersebut sampai di *node* lain. Pesan yang memiliki *encounter count* tinggi berarti pesan tersebut telah banyak terduplikasi dalam jaringan. Setiap pesan yang memiliki *max encounter count* akan senantiasa di *broadcast* ketika mencapai suatu *node* tertentu hingga *encounter count* pada pesan mencapai batas maksimal *encounter count* masing-masing pesan yang telah ditentukan. Batas maksimal *encounter count* harus lebih besar atau sama dengan jumlah *node*. *Routing protocol Epidemic with encounter count* ditunjukkan pada gambar 2.3



Gambar 2.3 *Epidemic with Encounter Count*[5]

3. ***Epidemic with Max Distance***

Epidemic with max distance merupakan optimasi dari *pure epidemic routing protocol* dengan memberikan maksimal jarak pada masing-masing pesan. Ketika pesan sampai pada suatu *node*, akan dihitung jarak antara pengirim pesan yang bersangkutan dengan *node* saat ini, apabila jarak yang didapatkan lebih kecil dari *max distance* maka pesan tersebut akan di *broadcast* ke *node* yang lain, namun apabila jarak yang didapatkan lebih besar dari *max distance*, maka pesan tersebut tidak akan di broadcast ke *node* yang lain.

2.4 *Global Positioning System* (GPS)

Global Positioning System (GPS) adalah sistem navigasi satelit berbasis ruang yang menyediakan informasi lokasi dan waktu dalam semua kondisi cuaca. GPS menyediakan kapabilitas bagi pengguna militer, sipil, dan komersial di seluruh dunia. GPS dikelola oleh pemerintah Amerika Serikat dan dapat diakses secara bebas oleh siapa saja yang memiliki alat yang dapat menerima sinyal GPS. GPS merupakan sistem navigasi berbasis satelit yang terdiri dari jaringan 24 satelit yang ditempatkan sesuai dengan orbitnya oleh Departemen Pertahanan Amerika Serikat. GPS pada mulanya ditujukan untuk kepentingan militer, tetapi pada tahun 1920, pemerintah Amerika Serikat membuat GPS tersedia untuk penggunaan sipil.

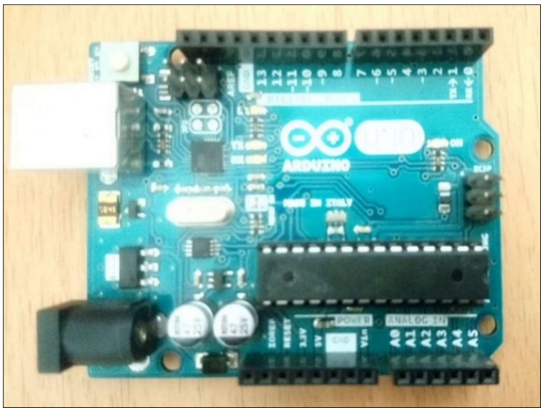
2.5 Algoritma *Simple Encryption*

Algoritma *Simple Encryption* adalah sebuah algoritma untuk melakukan enkripsi pesan secara sederhana. Algortima ini dapat berupa melakukan pergeseran setiap karakter pada kalimat atau kumpulan karakter , kombinasi operasi logika seperti xor, or , dan operasi logaritma log, dan sebagainya. Algoritma ini cocok diimplementasikan pada pesan yang pendek.

2.6 Arduino

Arduino adalah sebuah *hardware* dan *software open source* yang mendesain *single-board microcontroller* dan kit mikrokontroler untuk membangun alat digital yang bisa merasakan lingkungan dan mengontrol objek fisik maupun digital. Produk Arduino didistribusikan secara *open source* dan dilisensikan dibawah *GNU Lesser General Public License (LGPL)* atau *GNU General Public License (GPL)*, membuat produk Arduino dapat didistribusikan oleh siapapun tanpa harus membayar lisensi maupun royalti.[6]

Board arduino menggunakan banyak jenis mikroprosesor dan mikrokontroler. *Board* Arduino dilengkapi dengan set pin *input/output* (I/O) digital dan analog yang bisa digunakan untuk berkomunikasi dengan *board* lainnya atau modul expansi yang terdapat di pasaran. *Board* Arduino mempunyai antarmuka komunikasi *serial*, termasuk *Universal Serial Bus* (USB) yang digunakan untuk memprogram mikrokontroler Arduino melalui komputer. Mikrokontroler Arduino secara kusus diprogram menggunakan dialek dan fitur bahasa C dan C++ . Arduino dapat ditunjukkan pada gambar 2.4.

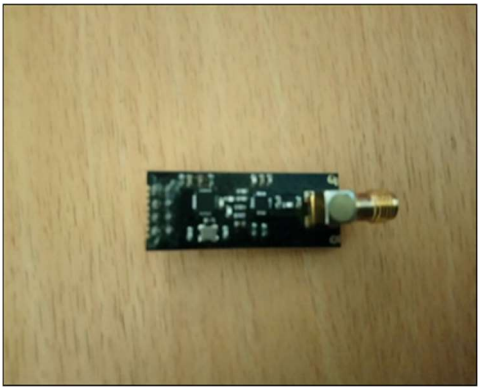


Gambar 2.4 Arduino

2.7 Modul *Wireless* nRF24L01

Modul *Wireless* nRF24L01 merupaan modul komunikasi jarak jauh yang memanfaatkan pita gelombang RF (*Radio Frequency*) 2.4GHz ISM (*Industrial, Scientific and Medical*) yang didesain untuk jaringan nirkabel yang membutuhkan penggunaan daya sangat rendah. Modul ini berupa sebuah *chip transreceiver* tunggal yang memiliki *baseband logic Enhanced Shockburst.* [7]

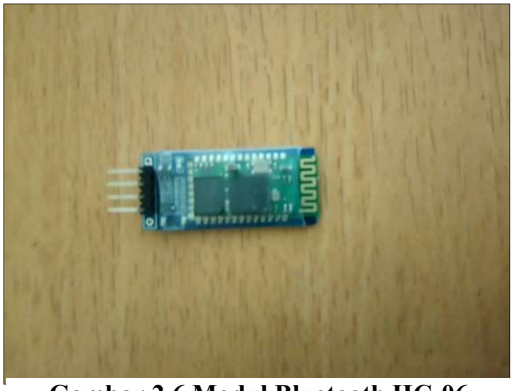
Modul ini cocok digunakan dengan *Microcontroller* Arduino Modul nRF24l01 dapat bekerja sebagai *transmitter* dan juga sebagai *receiver*. Namun nRF tidak bisa menjadi *transmitter* dan *receiver* dalam waktu yang bersamaan. Perlu adanya pergantian mode bila membutuhkan komunikasi dua arah antar *node* nRF. Selain itu melakukan pergantian mode dengan *timing* yang pas diperlukan karena paket yang datang akan *drop* bila modul dalam mode transmisi saat bersamaan. Diluar dari kelemahan tersebut modul nRF24L01 memiliki kelebihan dalam kehandalan, portabilitas, dan pemakaian daya. Pemakaian daya yang rendah dan dimensi yang kecil membuat modul ini banyak dipakai dalam bidang yang memerlukan komunikasi nirkabel tidak terkecuali jaringan sensor nirkabel. Modul *Wireless* nRF24L01 ditunjukkan pada gambar 2.5.



Gambar 2.5 Modul nRF24L01

2.8 Modul Bluetooth HC-06

HC-06 merupakan modul Bluetooth *slave* kelas dua yang di desain untuk komunikasi serial nirkabel. Untuk dapat berkomunikasi dengan Bluetooth lainnya, modul Bluetooth master seperti pada personal computer maupun perangkat android harus melakukan *pairing* dengan modul Bluetooth ini. Semua data yang diterima melalui input serial akan di transmisikan melalui udara [8].Modul Bluetooth HC-06 ditunjukkan pada gambar 2.6.



Gambar 2.6 Modul Bluetooth HC-06

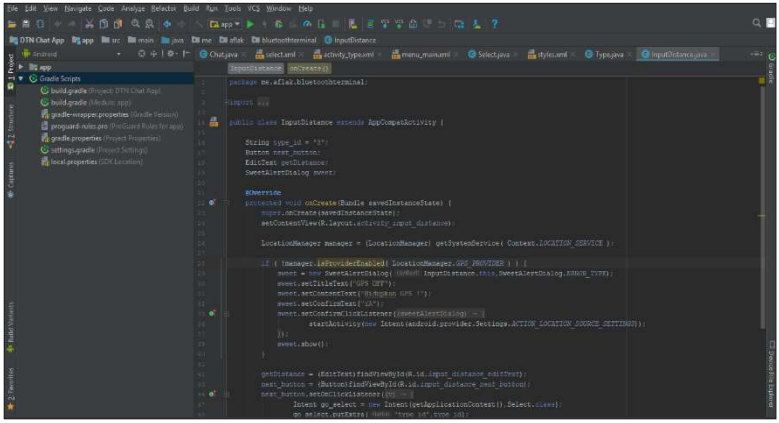
2.9 Arduino IDE

Arduino IDE merupakan sebuah perangkat lunak yang digunakan sebagai tempat untuk menulis logika-logika dari suatu skema rangkaian yang terhubung dengan *board* Arduino. Arduino IDE dibangun dengan bahasa pemrograman Java dan bersifat *cross-platform*. Barisan kode dalam Arduino IDE ditulis mengikuti aturan dari C/C++ dan baris kode ini disebut dengan istilah *sketch*.

Arduino IDE dipakai karena memiliki kompatibilitas baik dengan semua perangkat Arduino. Arduino IDE mempunyai fitur deteksi otomatis bila ada Arduino yang dihubungkan ke komputer. Untuk melakukan *debugging* Arduino IDE memiliki fitur *serial monitor* untuk komunikasi dengan Arduino.

2.10 Android Studio IDE

Android Studio adalah Lingkungan Pengembangan Terpadu -*Integrated Development Environment* (IDE) untuk pengembangan aplikasi Android. Selain merupakan editor kode IntelliJ dan alat pengembang yang berdaya guna, Android Studio menawarkan fitur lebih banyak untuk meningkatkan produktivitas Anda saat membuat aplikasi Android, misalnya Sistem versi berbasis Gradle yang fleksibel dan *Emulator* yang cepat dan kaya fitur. Android studio IDE ditunjukkan pada gambar 2.7.



Gambar 2.7 Android Studio IDE

(Halaman ini sengaja dikosongkan)

BAB III
DESAIN DAN PERANCANGAN

Bab ini akan menjelaskan mengenai dasar dari perancangan perangkat lunak pada sistem yang akan dibangun dalam Tugas Akhir. Perancangan perangkat lunak yang dibahas mengenai diskripsi umum sistem, arsitetur sistem, proses perancangan, alur dan implementasinya serta antar muka pengguna. Pada bab ini dan beberapa bab setelahnya akan menggunakan istilah-istilah asing yang artinya dapat dilihat pada tabel 3.1

Tabel 3.1 Daftar Istilah

Istilah	Arti
Node/Communication Node	Sebuah titik redistribusi atau titik akhir dari suatu komunikasi
DTN(*Delay Tolerant Network*)	Sebuah jaringan yang toleran dengan adanya *delay*
L*ifetime*	Interval waktu dari suatu pesan yang bersangkutan akan di *broadcast* oleh *communication node*
TTL(*Time to Live*)	*Lifetime* pada pesan yang akan dikirimkan
EC(*Encounter Count*)	Angka yang menunjukkan berapa kali suatu pesan telah melewati suatu *hop*
MD(*Max Distance*)	Jarak maksimal sebagai batas suatu pesan akan di *broadcast* ke *node* lain
Hop	Salah satu bagian dari jalur antara sumber dan tujuan

Preffix	Bagian awal dari suatu kata atau *string*
Suffix	Bagian akhir dari suatu kata atau *string*

3.1 Diskripsi Umum Sistem

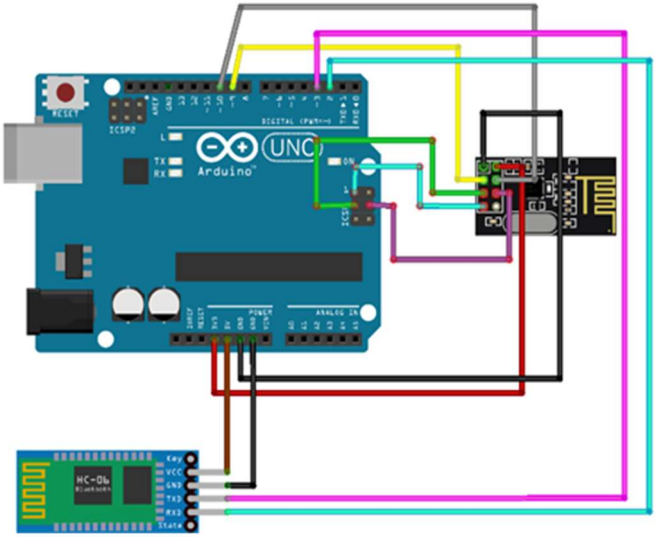
Sistem yang dibangun pada Tugas Akhir ini berupa sistem komunikasi *mobile peer to peer* dengan menerapkan konsep *Delay Tolerant Network* dan enkripsi. Dalam sistem komunikasi ini terdapat beberapa *communication node* yang akan saling mengirim pesan dengan *communication node* yang lain. Masing-masing *communication node* terdiri dari Arduino Uno, modul Bluetooth HC-06 dan nRF24L01.

Arduino Uno dalam sistem komunikasi ini berfungsi sebagai *controller* dari *communication node*. Modul Bluetooth HC-06 dalam sistem komunikasi ini berfungsi sebagai media penghubung antara perangkat android dengan Arduino Uno. Modul nRF24L01 pada sistem komunikasi ini berfungsi sebagai alat transmisi antara *communication node* satu dengan *communication node* yang lainnya. Pemilihan nRF24L01 pada sistem komunikasi ini dikarenakan modul nRF24L01 dapat melakukan transmisi jarak jauh, yaitu sejauh satu kilo meter serta modul ini tidak membutuhkan *supply* daya yang banyak sehingga dengan menggunakan nRF24L01 sistem komunikasi *mobile peer to peer* yang dibangun diharapkan dapat menjangkau area yang luas.

3.2 Arsitektur Sistem

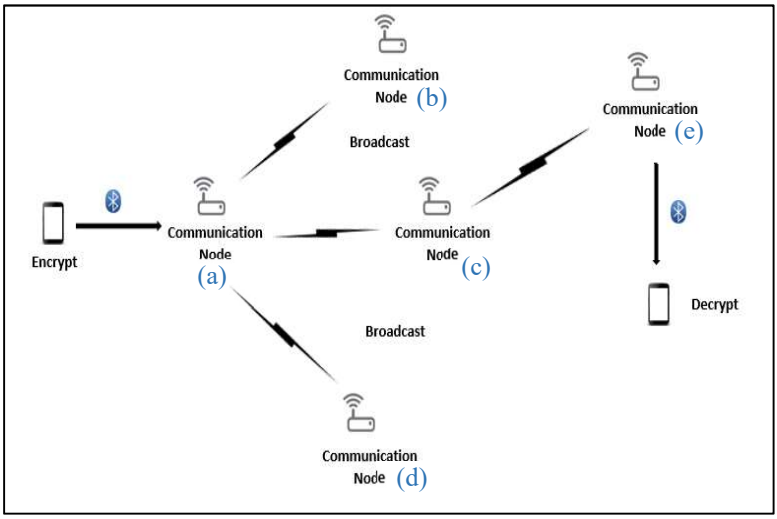
Sub bab ini akan dijelaskan mengenai arsitektur umum sistem yang akan dibangun. Sistem komunikasi *mobile peer to peer* yang dibangun terdiri dari lima buah *communication node*. Masing-masing *communication node* terdiri dari Arduino, modul nRF24L01 sebagai *controller* dari *communication node*, modul

Bluetooth HC-06 sebagai penghubung antara perangkat android dengan *communication node* dan modul nRF24L01 sebagai penghubung antara *communication node* satu dengan *communication node* yang lain. Gambar 3.1 menunjukkan rancangan *communication node* yang akan dibuat.



Gambar 3.1 Rancangan *Communcation Node*

Untuk dapat terjalin sebuah komunikasi, minimal harus ada perangkat android sebagai pengirim pesan dan penerima pesan. Masing-masing perangkat android tersebut telah terhubung dengan *communication node*. Masing-masing *communication node* akan saling mengirimkan pesan dengan *communication node* yang lainnya melalui modul nRF24L01 dengan cara suatu *communication node* akan melakukan *broadcast* pesan kepada *communication node* tetangga yang berada dalam jangkauan jarak modul nRF24L01. Skema keseluruhan sistem ditunjukkan pada gambar 3.2



Gambar 3.2 Skema Arsitetur Jaringan Sistem Keseluruhan

Setiap *communication node* memiliki *address* yang bersifat unik. Pada mulanya masing-masing pengguna harus melakukan *pairing* Bluetooth pada perangkat android dengan Bluetooth yang terpasang pada arduino. Setelah proses *pairing* berhasil, pengguna harus menentukan tujuan dari pengiriman pesan. Misalkan dalam ilustrasi yang ditunjukkan pada gambar 3.2 pengguna yang merupakan pengirim pesan adalah pengguna yang terhubung dengan *communication node* (a), sedangkan pengguna yang akan menerima pesan adalah pengguna yang terhubung dengan *communication node* (e).

Setelah menentukan tujuan pengiriman pesan, pengguna yang berperan sebagai pengirim pesan menuliskan pesan pada aplikasi pengiriman pesan dengan format [tujuan pesan][*space*][pesan] , lalu pengguna A menekan tombol "*send*" pada aplikasi pengiriman pesan. Ketika pengguna menekan tombol "*send*", aplikasi pengiriman pesan akan mengolah

masukan dari pengguna . Masukan dari pengguna akan ditambahkan ID pesan yang bersifat unik, lalu dilakukan proses enkripsi dari pesan yang telah diinputkan oleh pengguna. Pesan yang siap dikirm akan dikirim dari perangkat android ke communication node melalui modul Bluetooth, setelah *communication node* menerima pesan dari perangkat android, pesan akan ditransmisikan ke *communication node* yang lain menggunakan metode *broadcast*.

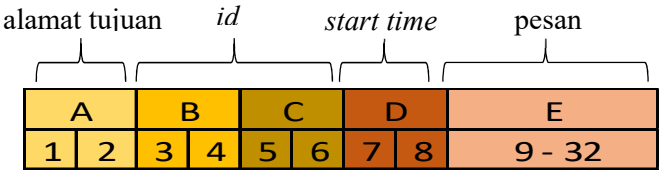
Di lain sisi *communication node* (e) yang merupakan tujuan pesan menerima *broadcast* pesan dari *communication node* (b) atau (c) maupun (d). langkah pertama yang dilakukan oleh *communication node* (e) adalah melakukan pengecekan apakah pesan tersebut pernah diterima oleh *communication node* (e), jika pesan belum pernah diterima sebelumnya, maka *communication node* (e) melakukan pengecekan lebih lanjut apakah alamat tujuan dari pesan tersebut sesuai dengan alamat pada *communication node* (e). Jika sesuai, maka *communication node* (e) akan mengirimkan pesan ke perangkat android yang terhubung dengannya melalui bluetooth. Perangkat android akan melakukan dekripsi pesan. Namun jika alamat tujuan pada pesan tidak sesuai dengan alamat *communication node* (e) , maka pesan akan disimpan dalam penyimpanan internal *communication node* (e) dan dalam periodik waktu tertentu pesan yang telah disimpan dalam *communication node* (e) akan di *broadcast* ke *communication node* yang lain sampai pesan diterima oleh tujuan pesan.

Penjelasan mekanisme komunikasi diatas merupakan mekanisme komunikasi dasar yang menerapkan *delay tolerant network* dengan *pure epidemic routing protocol*. Pada sub bab berikutnya akan dijelaskan implementasi komunikasi *peer to peer* dengan mengimplementasikan *delay tolerant network* dan optimasi dari *pure epidemic routing protocol* yang telah dijelaskan pada bab sebelumnya.

3.3 Perancangan Komunikasi *Peer to Peer* dengan Mengimplementasikan DTN *With Time-to-Live*

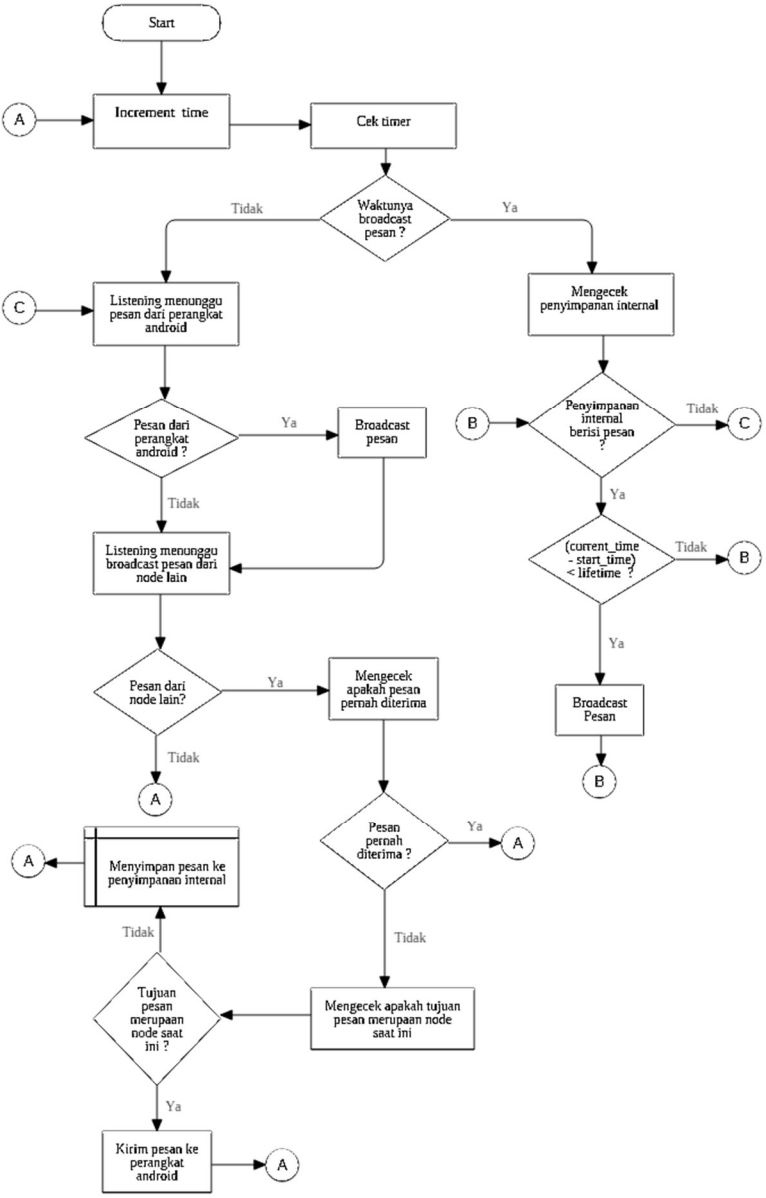
Sebagaimana yang telah dijelaskan pada bab sebelumnya, DTN *with Time to Live* merupakan sebuah sistem komunikasi yang mengimplementasikan *delay tolerant network* dan memberikan *lifetime* pada masing-masing pesan. Sebagai contoh apabila masing-masing pesan pada jaringan deberikan *lifetime* sepuluh menit, maka pesan tersebut akan terus di *broadcast* hingga menit kesepuluh. Pemberian *lifetime* pada masing-masing pesan bertujuan agar tidak terjadi kepadatan jaringan karena setiap pesan dilakukan *broadcast* secara terus menerus.

Terdapat perbedaan *header* pesan pada masing-masing tipe DTN yang telah disebutkan. Setiap pengiriman pesan menggunakan modul nRF24L01 maksimal sebanyak 32 karakter. Gambar 3.3 menunjukkan format *header* dan pesan pada DTN *with time to live.*



Gambar 3.3 Format Pesan Pada DTN *with* TTL

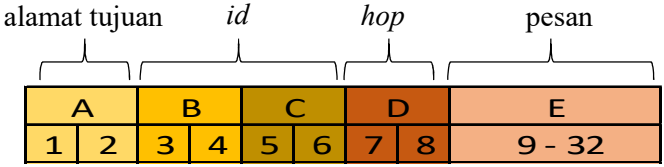
Pada blok A yang menempati karakter satu sampai dua digunakan untuk menyimpan alamat tujuan pesan. Blok B yang menempati karakter tiga sampai empat digunakan untuk menyimpan sumber pesan. Blok C yang menempati karakter lima sampai enam menyimpan nomor pesan dari masing-masing *node.* Blok B dan blok C digabungkan menjadi *id* pesan yang bersangkutan. Blok D yang menempati karakter ke lima sampai karakter ke enam menyimpan waktu pertama kali pesan di broadcast. Blok E yang menempati karakter ke sembilan sampai karakter ke 32 digunakan untuk menyimpan pesan. Diagram alir *node* dengan DTN with time to live ditunjukkan pada gambar 3.4



Gambar 3.4 Diagram Alir *Communication Node* dengan DTN *with* TTL

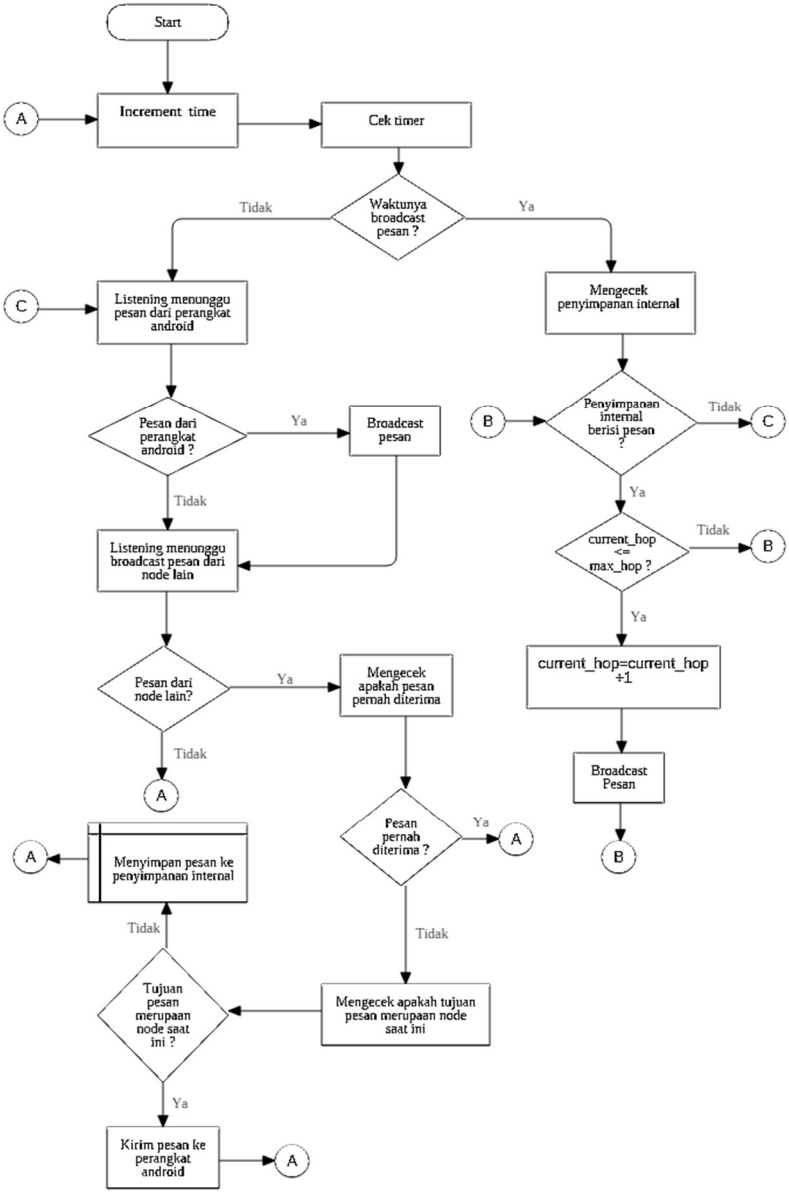
3.4 Perancangan Komunikasi *Peer to Peer* dengan Mengimplementasikan DTN *with Encounter Count*

DTN *with encounter count* merupakan sebuah sistem komunikasi yang mengimplementasikan *delay tolerant network* dan memberikan maksimal *hop* pada setiap pesan. Misalkan pesan A memiliki maksimal hop lima, pesan A akan terus di *broadcast* oleh *node* perantara apabila *hop* pesan A saat ini kurang dari sama dengan lima. Setiap sampai pada *node* perantara, *hop* pesan A akan ditambah satu. Gambar 3.5 menunjukkan format *header* dan pesan pada DTN *with encounter count*



Gambar 3.5 Format Karakter Pada DTN *with* EC

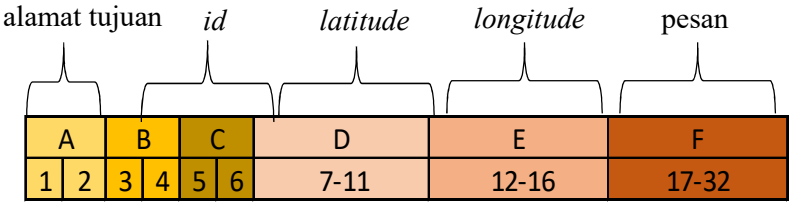
Pada blok A yang menempati karakter satu sampai dua digunakan untuk menyimpan alamat tujuan pesan. Blok B yang menempati karakter tiga sampai empat digunakan untuk menyimpan sumber pesan. Blok C yang menempati karakter lima sampai enam menyimpan nomor pesan dari masing-masing *communication node.* Blok B dan blok C digabungkan menjadi *id* pesan yang bersangkutan. Blok D yang menempati karakter ke lima sampai karakter ke enam menyimpan jumlah *hop* saat ini pada masing-masing pesan. Blok E yang menempati karakter ke sembilan sampai karakter ke tiga puluh dua digunakan untuk menyimpan pesan dari pengguna. Pada blok D akan terjadi perubahan setiap pesan yang bersangkutan sampai pada *node* perantara. Blok D menjadi indikator apakah pesan yang bersangkutan dapat di *broadcast* atau tidak. Diagram alir *communication node* dengan DTN *with encounter count* ditunjukkan pada gambar 3.6



Gambar 3.6 Diagram Alir *Communication Node* DTN *With* EC

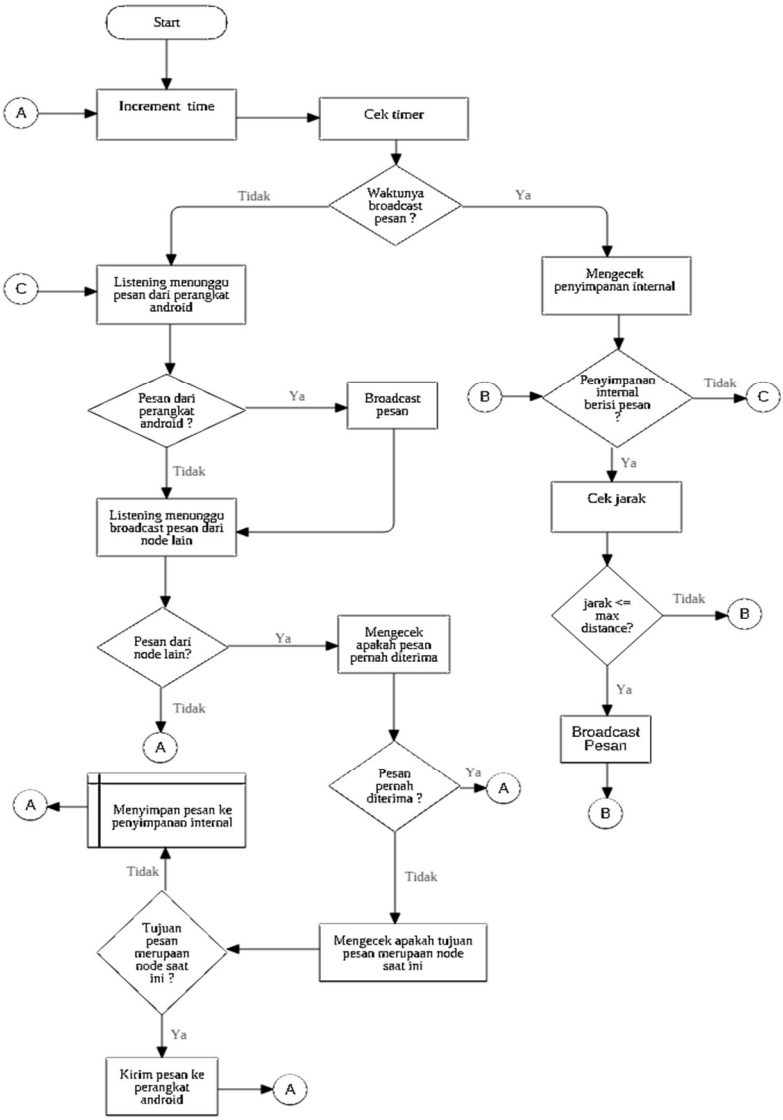
3.5 Perancangan Komunikasi *Peer to Peer* dengan Mengimplementasikan DTN *with Max Distance*

DTN *with max distance* merupakan sebuah sistem komunikasi yang mengimplementasikan *delay tolerant network* dan memberikan maksimal jarak pada masing-masing pesan. Pemberian masimal jarak pada masing masing pesan bertujuan agar tidak terjadi kepadatan jaringan karena setiap *node* melakukan *broadcast* pesan yang sama. Gambar 3.7 menunjukkan format header dan pesan pada DTN yang mengimplementasikan epidemic with max distance.



Gambar 3.7 Format Karakter Pada DTN *with* MD

Pada blok A yang menempati karakter satu sampai dua digunakan untuk menyimpan alamat tujuan pesan. Blok B yang menempati karakter tiga sampai empat digunakan untuk menyimpan sumber pesan. Blok C yang menempati karakter lima sampai enam menyimpan nomor pesan dari masing masing *communication node.* Blok B dan blok C digabungkan menjadi id pesan yang bersangkutan. Blok D merupaan posisi *suffix latitude* dari lokasi awal pesan. Blok E merupakan *suffix longitude* dari lokasi awal pesan. Alasan hanya menyertakan *suffix* dari *latitude* dan *longitude* pesan dikarenakan tiga karakter *prefix latitude* dan empat karakter *prefix longitude* memiliki kesamaan dibeberapa derah. . Blok F yang menempati karakter ke 17 sampai karakter ke 32 digunakan untuk menyimpan pesan. Diagram alir *node* dengan DTN *with max distance* ditunjukkan pada gambar 3.8.



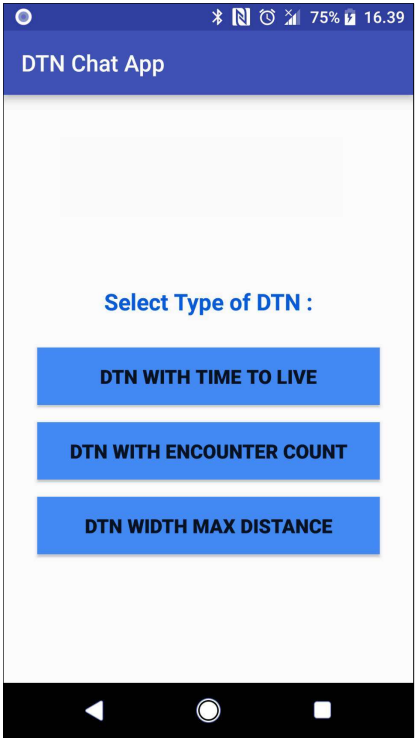
Gambar 3.8 Diagram Alir *Communication Node* dengan DTN *with* MD

3.6 Perancangan Antar Muka Pengguna

Pada sub bab ini akan dijelaskan beberapa halaman antar muka pengguna yang terdapat pada aplikasi perngiriman pesan dengan mengimplementasikan *delay tolerant network* di dalamnya

3.6.1 Halaman Menu

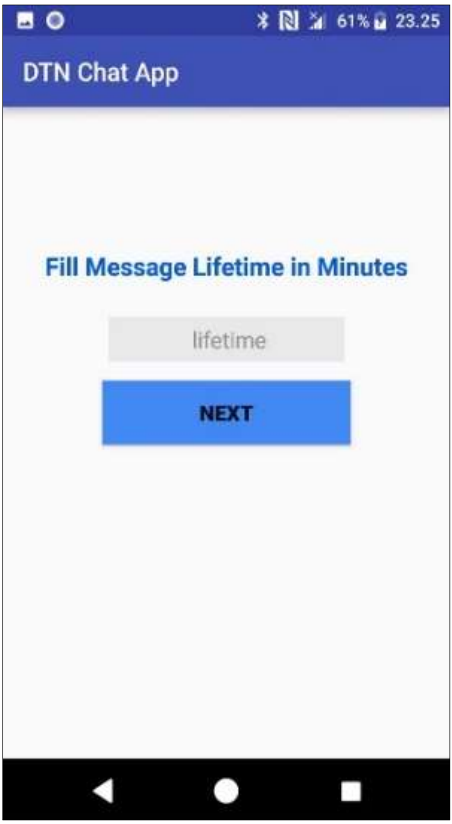
Pada halaman menu terdapat tiga tombol yang berisi pilihan tipe DTN, pengguna dapat memilih salah satu tipe DTN untuk memulai mengirim pesan. Gambar 3.9 menunjukkan halaman menu aplikasi pengiriman pesan.



Gambar 3.9 Halaman Menu

3.6.2 Halaman Input *Message Lifetime*

Pada halaman input *message lifetime,* pengguna yang telah memilih tipe DTN *with Time to Live* akan menginputkan *lifetime* dari masing-masing pesan yang akan dikirimkan. Satuan dari *lifetime* yang diinputkan pengguna adalah dalam menit Gambar 3.10 menunjukkan halaman input *message lifetime* pada apalikasi pengiriman pesan.



Gambar 3.10 Halaman Input *Message Lifetime*

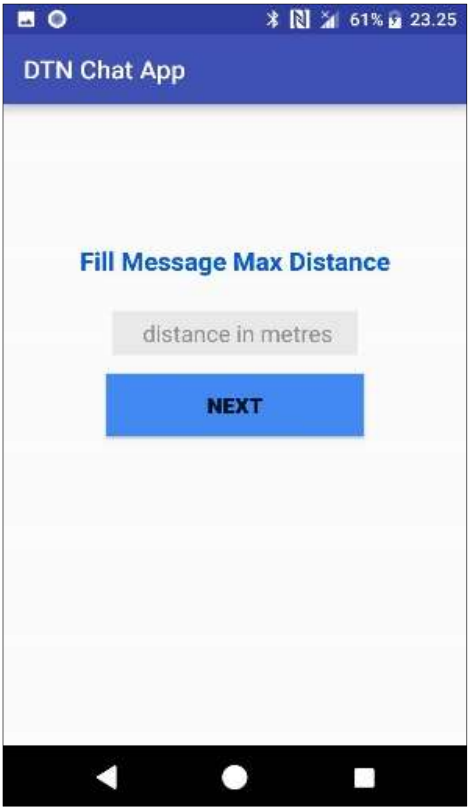
3.6.3 Halaman Input *Number of Hop*

Pada halaman input *number of hop,* pengguna yang telah memilih tipe DTN *with encounter count* akan menginputkan maksimal hop dari masing-masing pesan yang akan dikirimkan. Gambar 3.11 menunjukkan halaman input *number of hop* pada apalikasi pengiriman pesan.

Gambar 3.11 Halaman Input *Number of Hop*

3.6.4 Halaman Input *Max Distance*

Pada halaman input max distance, pengguna yang telah memilih tipe DTN *with max distance* akan menginputkan maksimal jarak pada masing-masing pesan yang dikirimkan. Satuan dari *max distance* yang diinputkan user dalam meter. Gambar 3.12 menunjukkan halaman input *max distance* pada apalikasi pengiriman pesan.



Gambar 3.12 Halaman Input *Max Distance*

3.6.5 Halaman Pilih Koneksi Bluetooth

Pada halaman pilih koneksi Bluetooth, akan muncul daftar koneksi Bluetooth yang telah berhasil di sandingkan dengan perangkat android. Untuk dapat terkoneksi dengan *communication node*, pengguna harus memilih salah satu daftar Bluetooth yang tersedia. Gambar 3.13 menunjukkan halaman pilih koneksi Bluetooth pada apalikasi pengiriman pesan.

Gambar 3.13 Halaman Pilih Koneksi Bluetooth

3.6.6 Halaman *Chat*

Pada halaman ini terdapat notifikasi di bagian atas bahwa koneksi dengan Bluetooth pada *communication node* berhasil. Untuk dapat mengirimkan pesan pengguna harus menginputkan teks pada kolom input teks lalu menekan tombol "*send*". Gambar 3.14 menunjukkan halaman chat pada apalikasi pengiriman pesan.

Gambar 3.14 Halaman *Chat*

(Halaman ini sengaja dikosongkan)

BAB IV
IMPLEMENTASI

Bab ini berisi penjelasan mengenai implementasi dari perancangan yang sudah dilakukan pada bab sebelumnya. Implementasi secara umum meliputi *pseudocode*, spesifikasi *hardware*, *schematics* dan sebagainya.

4.1 Lingkungan Implementasi

Lingkungan Implementasi merupakan lingkungan dimana sistem akan dibangun. Lingkungan implementasi dibagi menjadi Lingkungan Implementasi Perangkat keras dan Lingkungan Implementasi Perangkat Lunak

4.1.1 Lingkungan Implementasi Perangkat Keras

Pada bagian ini dijelaskan perangkat keras yang digunakan untuk membangun sistem. Lingkungan implementasi perangkat keras yang akan dibangun secara lebih rinci dijelaskan pada Tabel 4.1 dibawah ini.

Tabel 4.1 Lingkungan Implementasi Perangkat Keras

Perangkat	Detail
Perangkat Mikrokontroler	Mikrokontroler: • Atmega 328 Model: • Arduino UNO R3 Tegangan: • 5 - 12 V Memory Flash: • 32 KB

	SRAM: • 2 KB
Perangkat Wireless Transceiver	Model: • nRF24L01+ Single Chip 2.4GHz Transceiver Manufaktur: • Nordic Semiconductor Tegangan: • 1.9 - 3.6 V Frekuensi: • 2.4 GHz ISM Band Interface: • SPI Ukuran: • 20-pin 4x4 QFN Package
Perangkat Bluetooth	Model: • HC-06 Transmit power: • ≤ 4dBm, second stage Tegangan : • +3.3 VDC 50mA Frekuensi : • 2.4GHz ISM frequency band Kecepatan Transfer : • 2.1Mbps(Max)/160 kbps(Asynchronous) ; 1Mbps/1Mbps(Synchronous)
Perangkat Android	4. Model : • Sony Xperia XA1 Plus

	Sistem Operasi : • Android 8.0 Model Bluetooth: • Bluetooth 4.2 wireless technology 5. Model : • Asus Zenfone Go Sistem Operasi : • Android 5.1 Model Bluetooth: • 4.0, A2DP
Perangkat *Personal Computer*	1. Model : • DELL Inspiron 3443 Prosesor : • Intel(R) Core(TM) i7-5500 CPU @ 2.40GHz (4 CPUs), ~ 2,4GHz Memory: • 8192MB RAM 2. Model: • DELL Inspiron 3268 Prosesor: • Core - I5 7400T Memory: • 8192MB RAM

4.1.2 Lingkungan Implementasi Perangkat Lunak

Pada bagian ini dijelaskan perangkat lunak yang digunakan untuk membangun sistem komunikasi peer to peer. Tabel 4.2 menunjukkan lingkungan implementasi perangkat lunak

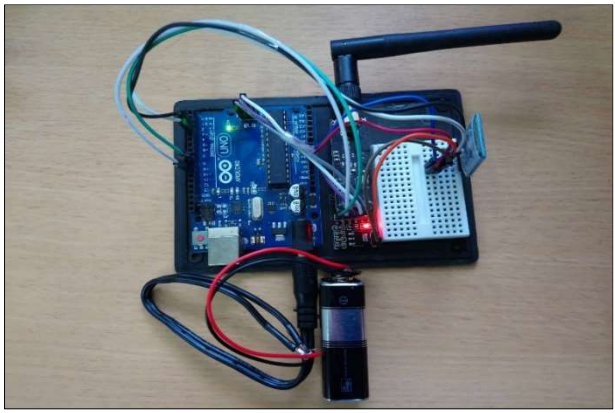
Tabel 4.2 Lingkunggan Implementasi Perangkat Lunak

Perangkat Lunak	Detail
Windows 10 Profesional	Windows 10 Profesional merupakan sistem operasi yang sebagian besar orang menggunakanya, karena kemudahan dalam pengoperasian Windows 10 Profesional
Library nRF24	*Library* yang digunakan agar Arduino dapat berkomunikasi dengan modul nRF24L01. Library ini sudah mencakup fungsi-ungsi yang dibutuhkan seperti pengiriman data, pembacaan transmisi, ACK payloads, dan pengaturan kekuatan transmisi.
Library CountUpDownTimer	*Library* yang digunakan untuk membuat penghitung waktu. Library ini digunakan karena Arduino tidak mempunyai IC untuk penghitung waktu sendiri. *Library* ini memanfaatkan pencacah pada Arduino yang menghitung berapa milidetik Arduino berjalan

Library SoftwareSerial	Library yang digunakan agar Arduino dapat berkomunikasi dengan perangkat Bluetooth HC-06
me.aflak.libraries:bluetooth:1.3.4	*Library* pada android studio IDE yang digunakan agar perangkat android dapat melakukan pairing dan berkomunikasi dengan perangkat Bluetooth lainnya, dalam penelitian ini agar perangkat android dapat berkomunikasi dengan perangkat Bluetooth yang telah terpasang pada Arduino.

4.2 Implementasi *Communication Node*

Pada bagian ini akan dijelaskan implementasi dari *communication node* yang digunakan pada sistem komunikasi *mobile peer to peer*. *Communication node* terdiri dari Arduino Uno yang dihubungkan dengan modul Bluetooth HC-06 untuk berkomunikasi dengan perangkat android dan dihubungkan dengan modul nRF24L01 untuk berkomunikaisi dengan *communication node* yang lainnya. Untuk menyuplai daya pada *communication node*, masing-masing *communication node* akan dihubungkan dengan sebuah baterai yang memiliki voltase 9 volt. Gambar 4.1 menunjukkan implementasi dari *communication node* yang akan digunakan pada sistem *komunikasi mobile peer to peer.*



Gambar 4.1 Implementasi *Communication Node*

4.3 Implementasi Kode Program Delay Tolerant Network With Time to Live

Pada bab ini akan di implementasikan *code* program *Delay Tolerant Network With Time to Live* ke masing-masing *communication node* yang sudah di rangkai seperti Gambar 4.1. Dalam implementasi *code* program akan di bagi dalam beberapa bagian dan selanjutnya akan dijelaskan pada subbab-subbab tersendiri.'

4.3.1 Variabel Global

Sub bab ini membahas variabel dan type data yang akan dibuat dalam menunjang fungsi-fungsi lain dalam program. Variabel global DTN *with Time to Live* ditunjukkan pada kode sumber 4.1.

```
pesan_bt  <- “”
set_time <- false
bc <-false
address <- 1000
```

```
myaddress <- <Node nrf address difference in
every node >
xMinute <- 0
xHour <- 0
pesan_ke <- 0
lifetime <- 0
T <- new CountUpDownTimer (DOWN)
r a d i o <- new RF24 ( 9, 10 )
Bt <- new SoftwareSerial (3 , 2)
Structure Pesan :
       msg = “”
       dst
       id_msg
       start_time
DECLARE psn[25] : ARRAY of Pesan
```

Kode Sumber 4.1 Variabel Global DTN *With Time to Live*

Sesui dengan kode sumber 4.1 , program akan mendeklarasikan dua *address*, yang pertama ialah *broadcast address* dengan nama variable *address* dan *address* masing-masing masing *communication node* dengan nama variable *myaddress.* Program mendeklarasikan variabel pesan_bt digunakan untuk menampung string yang diterima oleh *node* dari perangkat android. Program mendeklarasikan variabel set_time digunakan untuk mengatur waktu Arduino. Ketika perangkat android berhasil terhubung dengan Arduino melalui Bluetooth, perangkat android mengirimkan jam dan menit saat ini sesuai dengan format jam dan menit pada perangkat android. Arduino akan memulai sistem *clock* dengan menjalankan fungsi-fungsi yang terdapat pada library countDownTimer setelah menerima

input jam dan menit dari perangkat android. Dalam potongan pseudocode pada gambar 4.1 didefinisikan sebuah *structure* management penyimpanan pesan pada masing-masing *communication node*.

4.3.2 Fungsi Transmisi

Fungsi transmisi adalah fungsi yang berisi instruksi untuk nRF agar melakukan transmisi data ke alamat yang ditujukan. Fungsi transmisi pada DTN with Time to Live ditunjukkan pada kode sumber 4.2

```
function sentText(msg_will_send) :
   call radio.stopListening()
   panjang <- call msg_will_send.length()+1
   pesannya[panjang] <- ARRAY of CHAR
   call
   msg_will_send.toCharArray(pesannya,panjang)
   call radio.write(pesannya,panjang)
   call radio.startListening()
```

Kode Sumber 4.2 Fungsi Transmisi DTN *with Time to Live*

Sebelum melakukan pengiriman nRF diinstruksikan untuk berhenti melakukan *listening* dikarenakan nRF tidak bisa melakukan transmisi dan *listening* bersamaan. Selanjutnya nRF mengambil alamat *communication node* yang telah diatur ketika penjalankan fungsi *setup* pada Arduino dan melakukan transmisi pesan. Setelah berhasil melakukan transmisi, maka Arduino kembali pada *state Listening*

4.3.3 Fungsi *Listening* Bluetooth

Fungsi listening Bluetooth adalah fungsi yang berisi intruksi untuk bluetooth pada Arduino agar melakukan listening dan membaca apabila terdapat pesan dari perangkat android.

Fungsi listening Bluetooth pada DTN with Time to Live ditunjukkan pada kode sumber 4.3

```
function receiveBT():
       chr <- “”
       while(call bt.available())
          chr <- (char) call bt.read()
          pesan_bt <- pesan_bt+chr
          call delay(3)
       Endhile
       If set_time is false
          buf[call pesan_bt.length() + 1]:ARRAY
          of CHAR
          input[3] : ARRAY of INTEGER
          call pesan_bt.toCharArray(buf, call
          sizeof(buf))
          p ::refToChar
          *p <- buf
          Str ::refToChar
          i <- 0
          while((str <- call strtok_r(p,”/”,&p))
          is not NULL)
              input[i] <- call atoi(str)
              i <-i+1
              Endwhile
          xHour <- input[0]
          xMinute <- input[1]
          lifetime <- input[2]
          call T.StartTimer()
          set_time <-true
          pesan_bt <- “”
       else
```

```
        call sentText(pesan_bt)
        pesan_bt <- “”
```

Kode Sumber 4.3 Fungsi *Listening* Bluetooth Pada DTN *with Time to Live*

4.3.4 Fungsi *Listening* NRF

Fungsi berikut digunakan untuk mendengar dan membaca transmisi yang datang dari *communication node* yang lain. Fungsi listening NRF pada DTN *with Time to Live* ditunjukkan pada kode sumber 4.4

```
function receiveRF():
    pesan_rf[32] : ARRAY of CHAR
    is_pernah_terima <- false
    dst <- “”
    pre_msg <- “”
    msg_id <-“”
    time_start <-“”
    len <- call radio.getDinamicPayloadSize()
    call radio.read(&pesan_rf , len)
    pre_msg <- call String(pesan_rf)
    msg_id <- call pre_msg.substring(2,6);
    msg_id_int <- call msg_id.toInt()
    for each i in [1,pesan_ke]
       if  psn[i].id_msg same with msg_id_int
          is_pernah_terima <- true
    if is not is_pernah_terima
       dst <- call pre_msg.substring(0,2)
       dst_int <- call dst.toInt()
       if dst_int same with myaddress
           call bt.println(pre_msg)
       else
```

```
time_start <- call
pre_msg.substring(6,8)
pesan_ke <- pesan_ke + 1
psn[pesan_ke].start_time <- call
time_start.toInt()
psn[pesan_ke].id_msg<-msg_id_int
psn[pesan_ke].dst <- dst_int
psn[pesan_ke].msg <- call
pre_msg.substring(8,   call
pre_msg.length())
```

Kode Sumber 4.4 Fungsi *listening* NRF pada *DTN with Time to Live*

4.3.5 Fungsi *Update Time*

Fungsi *update time* adalah fungsi *time counter* / sistem *clock* pada Arduino, pada mulanya saat perangkat android terhubung dengan Arduino melalui Bluetooth, perangkat android mengirimkan jam dan menit saat ini. Setelah jam dan menit saat ini diterima oleh Arduino proses counter time dengan bantuan library countUpDownTimer dijalankan. Fungsi *update time* pada DTN with *Time to Live* ditunjukkan pada kode sumber 4.5

```
function UpdateTime:
      xMinute <- xMinute+1
      bc <- true
      if xMinute greater than 59
            xHour <- xHour+1
            xMinute <- 0
```

Kode Sumber 4.5 Fungsi *Update Time* DTN *with Time to Live*

4.3.6 Fungsi Pengecekan *Time* untuk *Broadcast* Pesan

Fungsi ini digunakan untuk menentukan waktu *broadcast* pesan, sebelum pesan di broadcast, masing-masing pesan akan dicek lifetimenya, ketika syarat *lifetime* masih memenuhi, maka pesan akan di *broadcast* ke *communication node* lain. Fungsi cek *time* pada DTN *with Time to Live* ditunjukkan pada kode sumber 4.6

```
function cekMessageTime :
   if call T.TimeHasChanged()
      if call T.ShowSecond() same with 0
       call countTime()
      if xMinute%3 same with 0 and bc is true
       for each i in [1,pesan_ke]
          new_start_time <- “”
          if (xMinute –  psn[i].start_time)
          smaller than lifetime
              if psn[i].start_time smaller than
              10
                 new_start_time <- “0”+call
                  String(    psn[i].start_time)
              else
                  new_start_time <- call String(
                  psn[i].start_time)
                  will_send <- call
                  String(psn[i].dst) + call
                  String(psn[i].id_msg)
                  +new_start_time+ psn[i].msg
                  sentText(will_send)
          delay(10)
     bc <- false
```

Kode Sumber 4.6 Fungsi Cek Time Pada DTN *with Time to Live*

4.4 Implementasi Kode Program *Delay Tolerant Network With Encounter Count*

Pada bab ini akan di implementasikan *code* program *Delay Tolerant Network with encounter count* ke masing-masing *communication node* yang sudah di rangkai seperti Gambar 4.1. Dalam implementasi *code* program akan di bagi dalam beberapa bagian dan selanjutnya akan dijelaskan pada subbab-subbab tersendiri.

4.4.1 Variabel Global

Terdapat beberapa perbedaan pada pendefinisian varibel global pada DTN *with encounter count* dibandingkan dengan DTN *with Time to Live*. Perbedaannya terletak pada structure penyimpanan pesan dan *variabel max_hop* untuk menyimpan maksimal hop pada masing-masing pesan. Variabel global pada DTN with *encounter count* ditunjukkan pada kode sumber 4.7

```
pesan_bt  <- “”
set_time <- false
bc <-false
address <- 1000
myaddress <- <Node nrf address difference in
every node >
xMinute <- 0
xHour <- 0
pesan_ke <- 0
max_hop <- 0
T <- new CountUpDownTimer (DOWN)
r a d i o <- new RF24 ( 9 , 10 )
Bt <- new SoftwareSerial (3 , 2)
Structure Pesan :
      msg = “”
      dst
```

```
id_msg
       jml_hop
       is_bc <- false
DECLARE psn[25] : ARRAY of Pesan
```

Kode Sumber 4.7 Variabel Global DTN *with Encounter Count*

4.4.2 Fungsi Transmisi

Fungsi transmisi pada DTN *with encounter count* ditunjukkan pada kode sumber 4.8

```
function sentText(msg_will_send) :
       call radio.stopListening()
       panjang <- call
       msg_will_send.length()+1
       pesannya[panjang] <- ARRAY of CHAR
       call msg_will_send.toCharArray
       (pesannya,panjang)
       call radio.write(pesannya,panjang)
       call radio.startListening()
```

Kode Sumber 4.8 Fungsi Transmisi DTN *with Encounter Count*

4.4.3 Fungsi *Listening* Bluetooth

Fungsi *listening* Bluetooth adalah fungsi yang berisi intruksi untuk bluetooth pada Arduino agar melakukan *listening* dan membaca apabila terdapat pesan dari perangkat android. Fungsi listening Bluetooth pada DTN *with encounter count* ditunjukkan pada kode sumber 4.9

```
function receiveBT():
       chr <- “”
       while(call bt.available())
```

```
chr <- (char) call bt.read()
         pesan_bt <- pesan_bt+chr
         call delay(3)
      Endhile
      If set_time is false
         buf[call pesan_bt.length() + 1] : ARRAY
         of CHAR
         input[3] : ARRAY of INTEGER
         call pesan_bt.toCharArray(buf, call
         sizeof(buf))
         p ::refToChar
         *p <- buf
         Str ::refToChar
         i <- 0
         while((str <- call strtok_r(p,"/",&p))
         is not NULL)
             input[i] <- call atoi(str)
             i <-i+1
             Endwhile
         xHour <- input[0]
         xMinute <- input[1]
         max_hop <- input[2]
         call T.StartTimer()
         set_time <-true
         pesan_bt <- ""
      else
         call sentText(pesan_bt)
         pesan_bt <- ""
```

Kode Sumber 4.9 Fungsi *Listening* Bluetooth DTN *with Encounter Count*

4.4.4 Fungsi *Listening* NRF

Fungsi berikut digunakan untuk mendengar dan membaca transmisi yang datang ke *node*. Pada fungsi *listening* NRF ini dilakukan pemecahan karakter dan disimpan dalam structure yang telah dijelaskan pada sub bab sebelumnya. Fungsi listening NRF pada DTN *with encounter count* ditunjukkan pada kode sumber 4.10

```
function receiveRF():
    pesan_rf[32] : ARRAY of CHAR
    is_pernah_terima <- false
    dst <- “”
    pre_msg <- “”
    msg_id <-“”
    time_start <-“”
    len <- call radio.getDinamicPayloadSize()
    call radio.read(&pesan_rf , len)
    pre_msg <- call String(pesan_rf)
    msg_id <- call  pre_msg.substring(2,6)
    msg_id_int <- call msg_id.toInt()
    for each i in [1,pesan_ke]
       if  psn[i].id_msg same with msg_id_int
          is_pernah_terima <- true
    if is not is_pernah_terima
       dst <- call pre_msg.substring(0,2)
       dst_int <- call dst.toInt()
       if dst_int same with myaddress
           call bt.println(pre_msg)
       else
           jml_hops <- call
           pre_msg.substring(7,8)
```

```
pesan_ke <- pesan_ke + 1
          psn[pesan_ke].jml_hop <- call
          jml_hops.toInt()
          psn[pesan_ke].id_msg <- msg_id_int
          psn[pesan_ke].dst <- dst_int
          psn[pesan_ke].msg <- call
          pre_msg.substring(8, call
          pre_msg.length())
```

Kode Sumber 4.10 Fungsi *Listening* NRF pada DTN *with Encounter Count*

4.4.5 Fungsi *Update Time*

Fungsi *update time* adalah fungsi *time counter* / *system clock* pada Arduino. *System clock* ini berfungsi untuk dapat melakukan broadcast pesan yang disimpan pada masing-masing *communication node* secara periodik. Fungsi update time pada DTN with *encounter count* ditunjukkan pada gambar 4.11

```
function countTime:
      xMinute <- xMinute+1
      bc <- true
      if xMinute greater than 59
            xHour <- xHour+1
            xMinute <- 0
```

Kode Sumber 4.11 Fungsi *Update Time* pada DTN *with Encounter Count*

4.4.6 Fungsi Cek *Time* untuk *Broadcast* Pesan

Fungsi ini digunakan untuk melakukan *broadcast* pesan yang tersimpan pada masing-masing *communication node*. Sebelum suatu pesan di *broadcast* oleh *communication node*, pad DTN with *encounter count* akan dilakukan pengecekan apakah

jumlah *hop* pada suatu pesan telah mencapai maksimal *hop* yang telah ditentukan. Jika jumlah *hop* pada pesan lebih kecil dari masimal hop yang ditentukan, maka pesan tersebut akan di broadcast. Fungsi broadcast pesan pada DTN with *encounter count* ditunjukkan pada gambar 4.12

```
function cekMessageTime :
   if call T.TimeHasChanged()
      if call T.ShowSecond() same with 0
       call countTime()
      if xMinute%3 same with 0 and bc is true
       for each i in [1,pesan_ke]
          new_start_time <- “”
              if psn[i].jml_hop smaller than
               max_hop and is notpsn[i].bc
                     psn[i].jml_hop <-
                     psn[i].jml_hop+1
                     psn[i].is_bc <-true
                     will_send <- call
                     String(psn[i].dst) + call
                     String(psn[i].id_msg)+call
                     String(psn[i].jml_hop) +
                     psn[i].msg
                     sentText(will_send)
          delay(10)
       bc <- false
```

Kode Sumber 4.12 Fungsi Cek *Time* Pada DTN *with Encounter Count*

4.5 Implementasi Kode Program *Delay Tolerant Network* with *Max Distance*

Pada bab ini akan di implementasikan *code* program *Delay Tolerant Network with max Distance* ke masing-masing *communication node* yang sudah di rangkai seperti Gambar 4.1. Dalam implementasi *code* program akan di bagi dalam beberapa bagian dan selanjutnya akan dijelaskan pada sub bab-sub bab tersendiri.

4.5.1 Variabel Global

Pada variabel global didefinisikan management structure penyimpanan pesan meliputi penyimpanan *id* pesan, penyimpanan lokasi *latitude* dan *longitude* masing-masing pesan. Variable global pada DTN with *max distance* ditunjukkan pada kode sumber 4.13

```
pesan_bt  <- “”
set_time <- false
bc <-false
address <- 1000
myaddress <- <Node nrf address difference in
every node >
xMinute <- 0
xHour <- 0
pesan_ke <- 0
lokasiku[2] : ARRAY of DOUBLE
max_distance <- 0
T <- new CountUpDownTimer (DOWN)
r a d i o <- new RF24 ( 9 , 10 )
Bt <- new SoftwareSerial (3 , 2)
Structure Pesan :
      msg = “”
```

```
       dst
       id_msg
       latitude
       longitude
       is_bc <- false
DECLARE psn[25] : ARRAY of Pesan
```

Kode Sumber 4.13 Variable Global DTN *with Max Distance*

4.5.2 Fungsi Transmisi

Fungsi transmisi digunakan untuk melakukan *broadcast* pesan dari satu *communication node* ke *communication node* yang lainnya. Fungsi transmisi pada DTN *with max distance* ditunjukkan pada kode sumber 4.14

```
function sentText(msg_will_send) :
       call radio.stopListening()
       panjang <- call
       msg_will_send.length()+1
       pesannya[panjang] <- ARRAY of CHAR
       call msg_will_send.toCharArray(
       pesannya,panjang)
       call radio.write(pesannya,panjang)
       call radio.startListening()
```

Kode Sumber 4.14 Fungsi Transmisi DTN *with Max Distance*

4.5.3 Fungsi *Listening* Bluetooth

Fungsi *listening* Bluetooth adalah fungsi yang berisi intruksi untuk bluetooth pada Arduino agar melakukan *listening* dan membaca apabila terdapat pesan dari perangkat android. Pada fungsi ini dilakukan penentuan lokasi dari *communication node.*

Communication node mendapatkan lokasinya melalui perangkat android yang terhubung dengannya. Fungsi *listening* Bluetooth pada DTN *with max distance* ditunkukkan pada kode sumber 4.15.

```
function receiveBT():
       chr <- “”
       while(call bt.available())
          chr <- (char) call bt.read()
          pesan_bt <- pesan_bt+chr
          call delay(3)
       Endhile
       If set_time is false
          buf[call pesan_bt.length() + 1] : ARRAY
          of CHAR
          input[3] : ARRAY of INTEGER
          location[2] : ARRAY of STRING
          call pesan_bt.toCharArray(buf, call
          sizeof(buf))
          p ::refToChar
          *p <- buf
          Str ::refToChar
          i <- 0
          while((str <- call strtok_r(p,”/”,&p))
          is not NULL)
              if i greater than 2
                    location[i-3] <- call
                     String(str)
              else
                   input[i] <- call atoi(str)
                    i <-i+1
          Endwhile
```

```
          xHour <- input[0]
          xMinute <- input[1]
          max_distance <- input[2]/1000
          lokasiku[0] <- call
          location[0].toDouble()
          lokasiku[1] <- call
          location[1].toDouble()
          call T.StartTimer()
          set_time <-true
          pesan_bt <- “”
       else
          call sentText(pesan_bt)
       pesan_bt <- “”
```

Kode Sumber 4.15 Fungsi *Listening* DTN *with Max Distance*

4.5.4 Fungsi *Listening* NRF

Fungsi berikut digunakan untuk mendengar dan membaca transmisi yang datang ke *communication node*. Dalam fungsi ini dilakukan pemecahan karakter setiap pesan yang sampai pada *communication node* dan disimpan pada *structure* penyimpanan pada DTN with *max distance* yang telah dijelaskan pada sub bab sebelumnya. Fungsi *listening* NRF pada DTN *with max distance* ditunjukkan pada kode sumber 4.16

```
function receiveRF():
    pesan_rf[32] : ARRAY of CHAR
    is_pernah_terima <- false
    dst <- “”
    pre_msg <- “”
    msg_id <-“”
    len <- call radio.getDinamicPayloadSize()
```

```
call radio.read(&pesan_rf , len)
pre_msg <- call String(pesan_rf)
msg_id <- call  pre_msg.substring(2,6)
msg_id_int <- call msg_id.toInt()
for each i in [1,pesan_ke]
   if  psn[i].id_msg same with msg_id_int
      is_pernah_terima <- true
if is not is_pernah_terima
   dst <- call pre_msg.substring(0,2)
   dst_int <- call dst.toInt()
   if dst_int same with myaddress
       call bt.println(pre_msg)
   else
       psn[++pesan_ke].latitude = call
       pre_msg.substring(6,11)
       psn[++pesan_ke].longitude = call
       pre_msg.substring(11,16)
       psn[pesan_ke].id_msg <- msg_id_int
       psn[pesan_ke].dst <- dst_int
       psn[pesan_ke].is_bc <- false
```

Kode Sumber 4.16 Fungsi *Listening* NRF DTN *with Max Distance*

4.5.5 Fungsi *Update Time*

Fungsi *update time* adalah fungsi *time counter* pada Arduino, pada mulanya saat perangkat android terhubung dengan Arduino melalui Bluetooth, pearangkat android mengirimkan jam dan menit saat ini. Setelah jam dan menit saat ini diterima oleh Arduino proses *counter time* dengan bantuan *library*

countUpDownTimer dijalankan. Fungsi *update time* pada DTN *with max Distance* ditunjukkan pada kode sumber 4.17

```
function countTime:
       xMinute <- xMinute+1
       bc <- true
       if xMinute greater than 59
              xHour <- xHour+1
              xMinute <- 0
```

Kode Sumber 4.17 Fungsi *Update Time* DTN *with Max Distance*

4.5.6 Fungsi Pengecekan *Time* untuk *Broadcast* Pesan

Pada fungsi ini akan dilakukan *broadcast* pesan dari satu *communication node* ke *communication node* yang lainnya. Sebelum suatu pesan di *broadcast*, akan dilakukan pengecekan jarak antara lokasi awal pesan dengan *communication node* saat ini. Apabila jarak yang di dapatkan lebih kecil daripada *max distance* yang telah ditentukan, maka pesan yang bersangkutan akan di *broadcast*. Fungsi pengecekan *time* untuk *broadcast* pesan pada DTN with max disctance ditunkukkan pada kode sumber 4.18

```
function cekMessageTime :
   if call T.TimeHasChanged()
      if call T.ShowSecond() same with 0
       call countTime()
      if xMinute%3 same with 0 and bc is true
       for each i in [1,pesan_ke]
          jarak <- call hitungJarak(
          psn[i].latitude,psn[i].longitude)
              if jarak smaller than max_distance
```

```
                will_send <- call +
                String(psn[i].dst) + call
                String(psn[i].id_msg)+
                psn[i].latitude +
                psn[i].longitude +
                psn[i].msg
                psn[i].is_bc <- true
                sentText(will_send)
        delay(10)
      bc <- false
```

Kode Sumber 4.18 Cek *Time* pada DTN *with Max Distance*

4.5.7 Fungsi Menghitung Jarak

Fungsi menghitung jarak digunakan untuk menghitung lokasi awal pesan dikirim dengan lokasi *node* pesan saat ini. Fungsi penghitungan jarak menggunakan Formula Haversine. Fungsi penghitungan jarak ditunjukkan pada kode sumber 4.19

```
function hitungJarak(lat1,lon1):
       lat2 <- lokasiku[0]
       lon2 <- lokasiu[1]
       R <- 6378
       dLat <- call deg2rad(lat2-lat1)
       dLon <- call deg2rad(lon2-lon1)
       a <- call sin(dLat/2) * call sin(dLat/2)
       + call cos(deg2rad(lat1)) * call
       cos(deg2rad(lat2)) * call sin(dLon/2) *
       call sin(dLon/2)
```

```
        c <- 2 * call atan2(call sqrt(a), call
        sqrt(1-a))
        d <- R*c
        return d
```

Kode Sumber 4.19 Fungsi Menghitung Jarak pada DTN *with Max Distance*

4.5.8 Fungsi Mengubah *Degree* ke Radian

Fungsi mengubah *degree* ke radian merupakan fungsi yang dibutuhkan oleh fungsi menghitung jarak untuk mengubah suatu bilangan bertipe *degree* ke radian. Fungsi mengubah degree ke radian ditunjukkan pada kode sumber 4.20

```
function deg2rad (deg):
        return deg * (PI/180)
```

Kode Sumber 4.20 Fungsi Mengubah *Degree* ke radian

4.6 Implementasi Kode Program Enkripsi dan Dekripsi

Metode enkripsi dan dekripsi yang digunakan dalam aplikasi pengiriman pesan ini menggunakan dua tahap enkripsi dan dekripsi, tahap pertama mengimplementasikan Metode Caesar dan setelah mendapatkan hasil dari metode pertama dilakukan teknik XOR terhadap hasil tersebut.

4.6.1 Metode Caesar

Metode Caesar yang digunakan dalam proses enkripsi dan dekripsi aplikasi pengiriman pesan ini ditunjukkan pada kode sumber 4.21

```
function Shift(msg,shift,is_encrypted):
        s <- “”
```

```
len <- call msg.length()

if is_encrypted is tue
      for x in range [0,len)
            c <- (char) call
            msg.charAt(x)+shift
            s <- s+c
else
      for x in range [0,len)
            c<- (char) call
            msg.charAt(x)-shift
return s
```

Kode Sumber 4.21 Metode Caesar

4.6.2 Teknik XOR

Teknik XOR yang digunakan dalam proses enkripsi dan dekripsi aplikasi pengiriman pesan ini ditunjukkan pada kode sumber 4.22

```
function encryptDecrypt(input, is_encrypt)
      hasil <- “”
      msg <- “”
      if is_encrypt is true
            msg <- call Shif(input,3,is_encrypt)
      else
            msg <- input
      key[3] : ARRAY of CHAR
      key <- {‘M’,’S’,’G’}
      output <- new StringBuilder();
```

```
for i in range [0, call msg.length() )
       call     output.append(char)     (     call
       msg.charAt(i) ^ key[i% call key.length()])
       if is_encrypted is false
           hasil <- Shift(call
           output.toString(),3,is_encrypted)
       else
              hasil <- call output.toString()
       return hasil
```

Kode Sumber 4.22 Teknik XOR

BAB V
UJI COBA DAN EVALUASI

Pada bab ini dibahas mengenai uji coba dan evaluasi dari segi fungsionalitas dan performa dari sistem. Uji coba fungsionalitas dan performa dibagi ke dalam beberapa skenario uji coba yang telah ditentukan. Pada setiap uji coba yang dilakukan tidak ada selisih *overhead* antara pesan yang di enkripsi maupun tidak di enkripsi.

5.1 Lingkungan Uji Coba

Dalam proses uji coba, digunakan beberapa perangkat keras dan perangkat lunak sebagai penunjang kebutuhan pengoperasian yang akan dijelaskan pada sub bab berikut.

5.1.1 Perangkat Keras

Kebutuhan perangkat keras yang digunakan dalam uji coba yaitu:

1. Empat buah Arduino uno R3.
2. Empat buah modul buetooth hc-06.
3. Empat buah NRF24L01.
4. Empat buah bateray litium 9v.
5. Empat buah *bread board*.
6. Dua buah ponsel android.

5.1.2 Perangkat Lunak

Kebutuhan perangkat lunak yang digunakan dalam uni coba yaitu:

1. Library nRF24.
2. Library CountUpDownTimer.
3. Library SoftwareSerial.
4. me.aflak.libraries:bluetooth:1.3.4

5.2 Uji Coba Fungsional

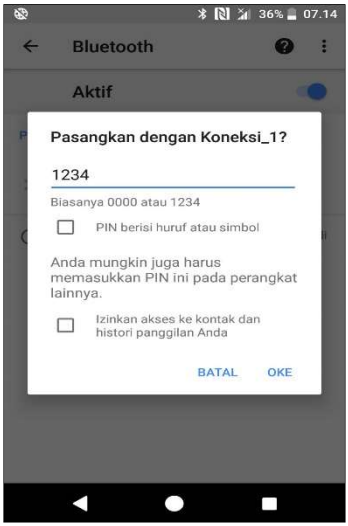
Uji coba fungsional merupakan sebuah pengujian terhadap jalannya fungsi-fungsi utama yang ada pada sistem. Uji coba fungsional meliputi:

1. Proses *pairing* antara Bluetooth di android dan Bluetooth yang telah terpasang di arduino.
2. Notifikasi menghidupkan Bluetooth pada android ketika Bluetooth dalam keadaan mati.
3. Notifikasi menghidupkan GPS pada android ketika GPS dalam keadaan mati.
4. Pengiriman pesan antara dua *communication node* tanpa enkripsi.
5. Pengiriman pesan antara dua *communication node* dengan enkripsi.
6. Pengiriman pesan lebih dari dua *communication node* dengan *delay tolerant network with time to life.*
7. Pengiriman pesan lebih dari dua *communication node* dengan *delay tolerant network with encounter count.*
8. Pengiriman pesan lebih dari dua *communication node* dengan *delay tolerant network with max distance.*

5.2.1 *Pairing* Bluetooth

Sebelum melakukan pengiriman pesan, langkah pertama yang harus dilakukan oleh pengguna adalah melakukan *pairing* Bluetooth pada android dengan Bluetooth yang terdapat pada *Communication node*. Proses *pairing* ini menggunakan fitur *pairing* yang disediakan oleh masing-masing perangkat android. Untuk melakukan *pairing* dengan Bluetooth pada *communication node* , pengguna akan diminta memasukkan password*, default password* pada modul Bluetooth pada *communication node*

adalah “1234”. Proses *pairing* dengan modul Bluetooth di *communication node* ditunjukkan pada gambar 5.1



Gambar 5.1 Proses *Pairing* Bluetooth

Pada gambar 5.1 perangkat Bluetooth pada android melakukan *pairing* dengan perangkat Bluetooth pada Arduino dengan nama “Koneksi_1”. Tabel 5.1 menunjukkan prosedur uji coba yang dilakukan pada proses pairing buetooth di android dan Bluetooth di Arduino.

Tabel 5.1 Prosedur Uji Coba *Pairing* Bluetooth

ID	UJ-01
Nama	Uji Coba Proses *Pairing* Bluetooth
Tujuan Uji Coba	Menguji proses *pairing* Bluetooth pada android dan Bluetooth pada Arduino

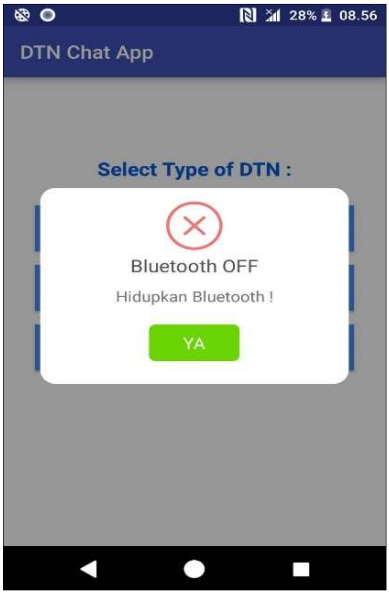
Kondisi Awal	1. Bluetooth pada Arduino tersambung dengan sumber daya sehingga dapat dikenali oleh perangkat lain. 2. Bluetooth pada android dalam keadaan menyala
Skenario	Memilih nama Bluetooth yang sesuai dari list Bluetooth yang dapat dikenali pada perangkat android.
Masukan	Password untuk malakukan pairing
Keluaran	Nama Bluetooth yang berhasil melalui proses pairing masuk dalam daftar Bluetooth yang telah berhasil dilakukan proses *pairing* pada perangkat android.
Hasil Uji Coba	BERHASIL

Proses *pairing* Bluetooth digunakan untuk memastikan bahwa Bluetooth pada perangkat android mengenali Bluetooth pada Arduino. Karena tanpa melakukan pairing, pesan yang dikirim dari perangkat android ke Arduino melalui Bluetooth tidak akan pernah sampai.

5.2.2 Menghidupkan Bluetooth Saat Pertama Kali Membuka Aplikasi

Ketika membuka aplikasi pengiriman pesan dan akan megirim pesan, terdapat kemungkinan Bluetooth dalam perangkat android dalam keadaan mati. Walaupun perangkat android sudah berhasil melakukan proses *pairing* dengan Bluetooth yang ada di *communication node* dan data *pairing* telah tersimpan di perangkat android. Maka, ketika dalam keadaan

tersebut, aplikasi pengiriman pesan akan mengintruksikan pengguna untuk menyalakan Bluetooth yang ada di perangkat android. Intruksi untuk menyalakan Bluetooth kepada pengguna dari aplikasi pengiriman pesan ditunjukkan pada gambar 5.2.



Gambar 5.2 Intruksi Menyalakan Bluetooth

Pada gambar 5.2 aplikasi pengirim pesan memberikan intruksi kepada pengguna bahwa Bluetooth pada perangkat android dalam keadaan mati. Ketika pengguna menekan tombol "YA", maka aplikasi pengirim pesan menghidupkan Bluetooth pada perangkat android. Tabel 5.2 menunjukkan prosedur uji coba yang dilakukan pada saat membuka aplikasi pengirim pesan ketika Bluetooth pada perangkat android dalam keadaan mati.

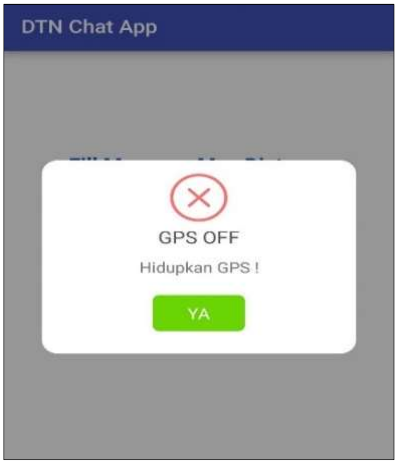
Tabel 5.2 Prosedur Uji Coba Membuka Aplikasi Saat Bluetooth Perangkat Dalam Keadaan Mati

ID	UJ-02
Nama	Uji coba proses menyalakan Bluetooth pada perangkat android.
Tujuan Uji Coba	Menguji respon aplikasi pengirim pesan ketika hendak mengirim pesan dan dalam keadaan Bluetooth pada perangkat android mati.
Kondisi Awal	Bluetooth pada perangkat android dalam keadaan mati
Skenario	Membuka aplikasi pengiriman pesan
Masukan	Menekan tombol "YA" untuk meyalakan Bluetooth pada perangkat android.
Keluaran	Bluetooth pada perangkat android meyala.
Hasil Uji Coba	BERHASIL

5.2.3 Menghidupkan *Global Positioning System* (GPS)

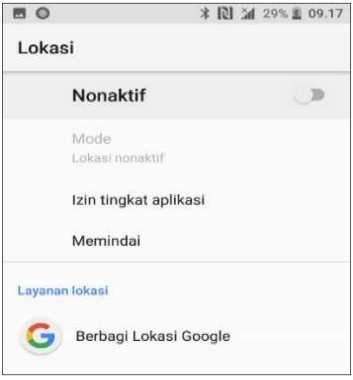
Ketika pengguna memilih DTN *with max distance* pada halaman menu aplikasi pengiriman pesan, maka GPS pada perangkat android harus dalam keadaanya menyala. Namun terdapat kemungkinan ketika pengguna memilih DTN *with max distance*, GPS pada perangkat android dalam keadaan mati. Ketika dalam keadaan tersebut, aplikasi pengiriman pesan akan memberikan notifikasi kepada pengguna bahwa GPS dalam keadaan mati dan mengintruksikan penggun auntuk menyalakan GPS pada perangkat android. Tampilan notifikasi dari aplikasi pengiriman pesan bahwa GPS dalam keadaan mati dan

mengintruksikan pengguna untuk menghidupkan GPS ditunjukkan pada gambar 5.3



Gambar 5.3 Intruksi Untuk Menyalakan GPS

Pada gambar 5.3 aplikasi pengiriman pesan memberikan notifikasi kepada pengguna bahwa GPS pada perangkat android dalam keadaaan mati. Ketika user menekan tombol "YA", maka aplikasi pengiriman pesan akan membuka halaman pengaturan untuk menyalakan GPS yang ditunjukan pada gambar 5.4



Gambar 5.4 Pengaturan GPS Pada Perangkat Android

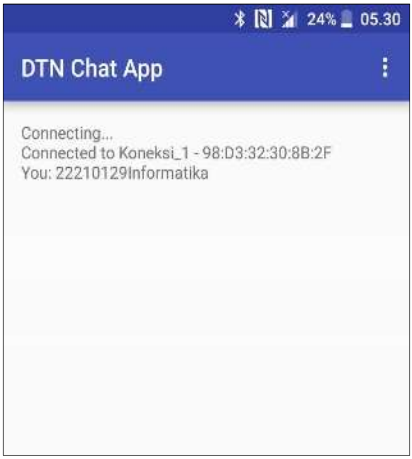
Tabel 5.3 menunjukkan prosedur uji coba yang dilakukan pada saat memilih DTN berdasarkan waktu dan GPS pada perangkat android dalam keadaan mati.

Tabel 5.3 Prosedur Uji Coba Memilih DTN *with Max Distance* Ketika GPS Perangkat Off

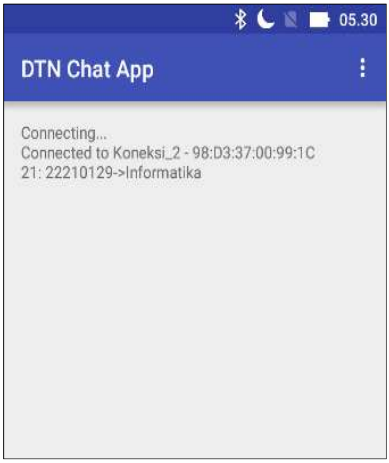
ID	UJ-03
Nama	Uji coba proses menyalakan GPS pada perangkat android
Tujuan Uji Coba	Menguji respon aplikasi pengirim pesan ketika pengguna memilih DTN berdasarkan jarak dan GPS pada perangkat android mati.
Kondisi Awal	GPS pada perangkat android dalam keadaan mati
Skenario	Menekan tombol DTN *with Max Distance*
Masukan	Menekan tombol "YA" untuk membuka pengaturan menghidupkan GPS
Keluaran	Muncul pengaturan untuk menyalakan GPS pada perangkat android
Hasil Uji Coba	BERHASIL

5.2.4 Pengiriman Pesan Tanpa Enkripsi

Sebelum mengimplementasikan algoritma *simple encryption* pada aplikasi pengiriman pesan, perlu dipastikan terlebih dahulu bahwa fungsionalitas pengiriman pesan dari *communication node* pengirim ke *communication node* penerima berhasil. Uji coba pengiriman pesan tanpa enkripsi ditunjukkan pada gambar 5.5 dan 5.6



Gambar 5.5 Halaman *Chat* Pengirim Tanpa Enkripsi



Gambar 5.6 Halaman *Chat* Penerima Tanpa Enkripsi

Pada gambar 5.5 string "22210129Informatika" merupakan string yang dikirimkan. Pengirim mengirimkan pesan dengan id pesan 2101 pada menit 29 dan diterima oleh penerima pada menit yang sama. Pengirim mengirimkan pesan "Informatika" dan diterima oleh penerima berupa string "Informatika". Tabel 5.4 menunjukkan prosedur uji coba pengiriman pesan tanpa melalui proses enkripsi.

Tabel 5.4 Prosedur Uji Coba Pengiriman Pesan Tanpa Melalui Proses Enkripsi

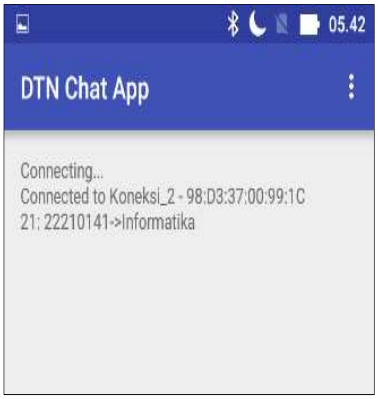
ID	UJ-04
Nama	Uji coba pengiriman pesan tanpa melalui proses enkripsi
Tujuan Uji Coba	Mengevaluasi kemampuan dasar sistem komunikasi dalam mengirimkan pesan
Kondisi Awal	Perangkat android A sebagai pengirim pesan telah terkoneksi

	dengan *communication node* A dan perangkat android B sebagai penerima pesan telah terkoneki dengan *communication node* B
Skenario	Perangkat android A mengirimkan pesan tanpa melalui proses enkripsi ke perangkat android B
Masukan	Perangkat android A mengirimkan pesan "Informatika" kepada perangkat android B
Keluaran	Perangkat android B menerima pesan dan menampilkan pada halaman *chat* berupa text "Informatika".
Hasil Uji Coba	BERHASIL

5.2.5 Pengiriman Pesan dengan Proses Enkripsi

Pada uji coba pengiriman pesan dengan proses enkripsi, pengirim pesan akan menginputkan teks yang akan dikirimkan ke penerima pesan. Ketika pengirim pesan menekan tombol "send", maka aplikasi pengiriman pesan akan melakukan enkripsi teks yang telah diinputkan pengirim. Setelah proses enkripsi selesai, pesan akan dikirimkam ke penerima pesan. Disisi lain, ketika penerima pesan menerima pesan, langkah pertama yang harus dilakukan adalah melakukan dekripsi pesan. Setelah berhasil melakukan dekripsi pesan, maka aplikasi akan menampilkan pesan yang telah di dekripsi pada halaman chat penerima di aplikasi pengiriman pesan . Uji coba pengiriman pesan dengan enkripsi ditunjukkan pada gambar 5.7 dan 5.8

Gambar 5.8 Pengirim Pesan dengan Enkripsi



Gambar 5.7 Penerima Pesan dengan Enkripsi

Untuk mempermudah pembuktian bahwa pesan dari pengirim telah dienkripsi, maka string siap kirim dari sisi pengirim ditampilkan pada halaman *chat* pengirim pesan. Gambar 5.8 pengirim pesan mengirimkan pesan dengan id 2101 pada menit 41. Pengirim mengirimkan pesan berupa teks "Informatika", kemudian pesan di enkrip oleh aplikasi pengiriman pesan menjadi "*.?&7)$+#7" . Teks yang telah di enkrip kemudian dikirimkan ke penerima pesan. Disisi lain, penerima pesan setelah menerima pesan melakuan dekripsi pesan lalu menampilkan pesan seperti pada gambar 5.7. Tabel 5.5 menunjukkan prosedur uji coba pengiriman pesan dengan melalui proses enkripsi.

Tabel 5.5 Prosedur Uji Coba Pengiriman Pesan dengan Melalui Proses Enkripsi

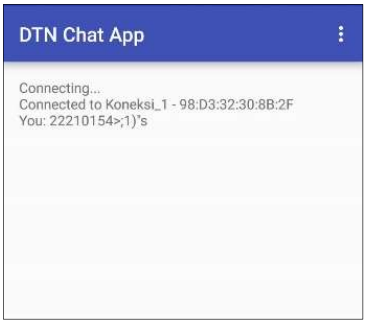
ID	UJ-05
Nama	Uji coba pengiriman pesan dengan melalui proses enkripsi

Tujuan Uji Coba	Mengevaluasi proses pengiriman pesan pada aplikasi pengiriman pesan
Kondisi Awal	Perangkat android A sebagai pengirim pesan telah terkoneksi dengan *communication node* A dan perangkat android B sebagai penerima pesan telah terkoneki dengan *communication node* B
Skenario	Perangkat android A mengirimkan pesan dengan melalui proses enkripsi ke perangkat android B
Masukan	Perangkat android A mengirimkan pesan "Informatia" yang telah dienkripsi kepada perangkat android B
Keluaran	Perangkat android B menerima pesan dan melakukan dekripsi pesan lalu menampilkan pada halaman *chat* berupa text "Informatika".
Hasil Uji Coba	BERHASIL

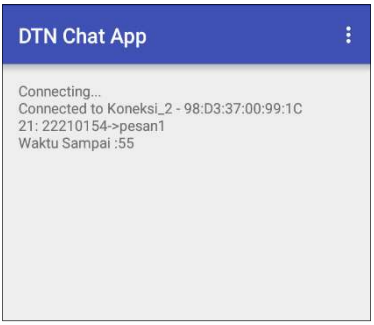
5.2.6 Pengiriman Pesan Lebih Dari Dua *Communication Node* DTN *with* TTL

Dalam uji coba ini terdapat tiga buah *communication node. Node* A sebagai pengirim pesan, *node* B sebagai penerima pesan dan *node* C sebagai *node* perantara. Antar *node* terdapat jarak 20 meter. Dikirimkan satu pesan. Pesan tersebut diberi

lifetime 10 menit dan node C akan melakukan *broadcast* pesan yang diterimanya setiap dua menit sekali. Untuk menghindari pengiriman langsung dari *node* A ke *node* B, maka ketika *node* A mengirim pesan yang ditujukan ke *node* B, *node* B dimatikan lalu dihidupkan kembali untuk menerima *broadcast* pesan dari *node* C. Gambar 5.9 merupakan hasil uji coba dari *node* A dan gambar 5.10 merupakan hasil uji coba *node* B



Gambar 5.9 *Node* Pengirim Dalam Uji Coba DTN *with* TTL dengan 3 *Communication node*



Gambar 5.10 *Node* Pengirim Dalam Uji Coba DTN *with* TTL dengan 3 *Communication Node*

Pesan yang dikirimkan memiliki id 2101 dan dikirim pada menit ke 54. Dari sisi penerima, pesan dengan id 2101 diterima pada menit ke 55. Adanya *delay* satu menit antara pengirim dan penerima terjadi karena *node* C harus menunggu waktu periodik *broadcast* nya untuk melakukan *broadcast* pesan. Tabel 5.6 menunjukkan prosedur uji coba DTN with TTL dengan menggunakan tiga *communication node.*

Tabel 5.6 Prosedur Uji Coba DTN *with* TTL dengan menggunakan tiga *communication node*

ID	UJ-06
Nama	Uji coba pengiriman pesan menggunakan tiga

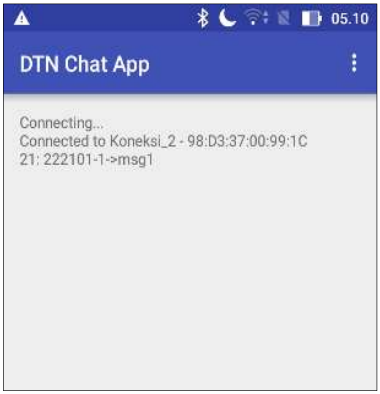
	communication node dengan DTN *with* TTL
Tujuan Uji Coba	Mengevaluasi pengiriman pesan dengan lebih dari dua *communication node* menggunakan DTN *with* TTL
Kondisi Awal	1. Perangkat android A sebagai pengirim pesan telah terkoneksi dengan *communication node* A dan perangkat android B sebagai penerima pesan telah terkoneki dengan *communication node* B. 2. User memilih *DTN with Time To Live* pada aplikasi pengiriman pesan. 3. User menginputkan *lifetime* pesan 10 menit.
Skenario	Perangkat android A mengirimkan pesan dengan melalui proses enkripsi ke perangkat android B
Masukan	Perangkat android A mengirimkan pesan "pesan 1" hingga "pesan 5" yang telah dienkripsi kepada perangkat android B.
Keluaran	Perangkat android B menerima pesan dan melakukan dekripsi pesan lalu menampilkan pada halaman *chat* berupa daftar

	pesan masuk yang ditunjukkan pada gambar 5.10
Hasil Uji Coba	BERHASIL

5.2.7 Pengiriman Pesan Lebih Dari Dua *Communication Node* DTN *with Encounter Count*

Dalam uji coba ini terdapat tiga buag *communication node. Node* A sebagai pengirim pesan, *node* B sebagai penerima pesan dan node C sebagai *node* perantara. Antar *node* terdapat jarak 20 meter. Masing-masing pesan diberikan maksimal hop sebanyak lima. Gambar 5.12 merupakan hasil uji coba dari *node* A dan gambar 5.11 merupakan hasil uji coba *node* B

Gambar 5.12 *Node* Pengirim



Gambar 5.11 *Node* penerima

Pesan yang dikirimkan memiliki id 2101, 2102 dan 2103. Dua karakter setelah id merupakan total *hop* saat ini. String "-0" berarti pesan belum mencapai *node* lain. String "-1" pada *node* penerima berarti pesan telah mencapai *node* lain sebelum sampai di *node* tujuan. Tabel 5.7 menunjukkan prosedur uji coba DTN

with *Encounter Count* dengan menggunakan tiga *communication node.*

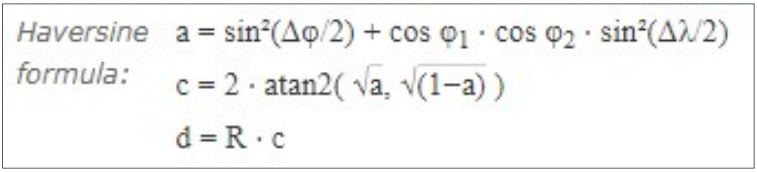
Tabel 5.7 Prosedur Uji Coba DTN *with Encounter Count* dengan Menggunakan Tiga *Communication Node*

ID	UJ-07
Nama	Uji coba pengiriman pesan menggunakan tiga *communication node* dengan DTN *with Encounter Count*
Tujuan Uji Coba	Mengevaluasi pengiriman pesan dengan lebih dari dua *communication node* menggunakan DTN with *Encounter Count*
Kondisi Awal	1. Perangkat android A sebagai pengirim pesan telah terkoneksi dengan *communication node* A dan perangkat android B sebagai penerima pesan telah terkoneki dengan *communication node* B. 2. User memilih DTN with *Encounter Count* pada aplikasi pengiriman pesan. 3. User menginputkan jumlah hop 5.
Skenario	Perangkat android A mengirimkan pesan dengan melalui proses enkripsi ke perangkat android B
Masukan	Perangkat android A mengirimkan pesan “msg 1”

	hingga “msg 3” yang telah dienkripsi kepada perangkat android B.
Keluaran	Perangkat android B menerima pesan dan melakukan dekripsi pesan lalu menampilkan pada halaman *chat* berupa daftar pesan masuk yang ditunjukkan pada gambar 5.12
Hasil Uji Coba	BERHASIL

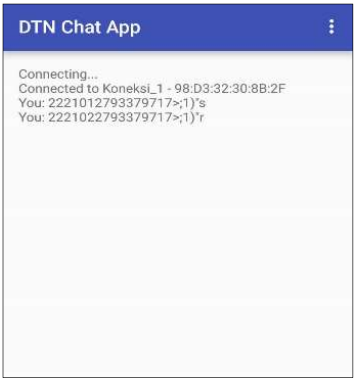
5.2.8 Pengiriman Pesan Lebih Dari *Dua Communication Node* DTN *with Max Distance*

Dalam uji coba ini terdapat tiga buah *communication node. Node* A sebagai pengirim pesan, *node* B sebagai penerima pesan dan node C sebagai *node* perantara. Antar *node* terdapat jarak 20 meter. Masing-masing pesan yang dikirim dari *node* A ke *node* B diberikan *max distance* sebesar 500 meter. Untuk menghindari pengiriman pesan secara langsung dari *node* A ke *node* B, ketika *node* A melakukan *broadcast* pesan yang ditujukan pada *node* B, maka node B dimatikan lalu di hidupkan kembali untuk menerima *broadcast* pesan dari node C. Untuk menghitung jarak antara lokasi awal pesan dengan lokasi *node* saat ini menggunakan Formula Haversine yang ditunjukkan pada gambar 5.13

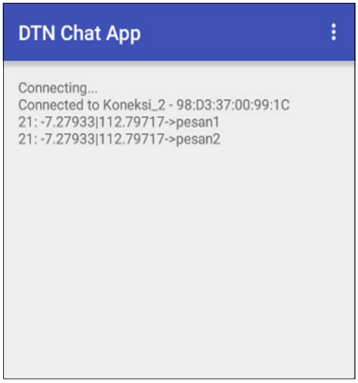


Gambar 5.13 Formula Haversine

φ adalah *latitude*, λ adalah *longitude*, R adalah radius bumi (radius = 6,371km). Gambar 5.14 merupakan hasil uji coba dari *node* A dan gambar 5.15 merupakan hasil uji coba *node* B



Gambar 5.14 *Node* pengirim dalam uji coba DTN *with Max Distance* dengan 3 *Communication node*



Gambar 5.15 *Node* penerima dalam uji coba *DTN with Max Distance* dengan 3 *Communication Node*

Tabel 5.8 menunjukkan prosedur uji coba DTN with *Encounter Count* dengan menggunakan tiga *communication node.*

Tabel 5.8 Prosedur Uji Coba DTN *with Max Distance* dengan Menggunakan Tiga *Communication Node*

ID	UJ-08
Nama	Uji coba pengiriman pesan menggunakan tiga *communication node* dengan DTN *with Max Distance*
Tujuan Uji Coba	Mengevaluasi pengiriman pesan dengan lebih dari dua

	communication node menggunakan DTN *with Max Distance*
Kondisi Awal	1. Perangkat android A sebagai pengirim pesan telah terkoneksi dengan *communication node* A dan perangkat android B sebagai penerima pesan telah terkoneki dengan *communication node* B. 2. User memilih DTN with Max Distance pada aplikasi pengiriman pesan. 3. User menginputkan *Max Distance* 500 m.
Skenario	Perangkat android A mengirimkan pesan dengan melalui proses enkripsi ke perangkat android B
Masukan	Perangkat android A mengirimkan pesan "pesan1" dan "pesan2" yang telah dienkripsi kepada perangkat android B.
Keluaran	Perangkat android B menerima pesan dan melakukan dekripsi pesan lalu menampilkan pada halaman *chat* berupa daftar pesan masuk yang ditunjukkan pada gambar 5.14
Hasil Uji Coba	BERHASIL

5.3 Uji Coba Performa

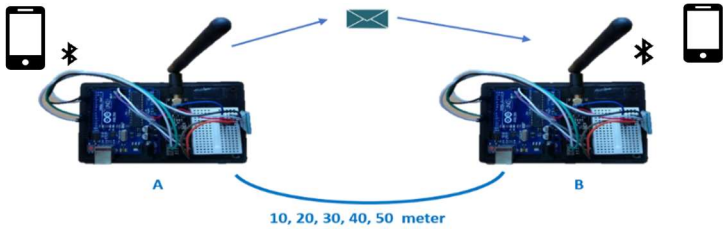
Uji coba performa sistem dilakukan untuk menguji kehandalan sistem. Adapun uji coba performa yang dilakukan meliputi:

1. Pengaruh jarak terhadap keberhasilan pengiriman pesan
2. Pengaruh pemberian *delay* pada dua pesan yang berurutan terhadap keberhasilan pengiriman pesan
3. Pengaruh jumlah data terhadap keberhasilan pengiriman pesan
4. Pengaruh ukuran data terhadap keberhasilan pengiriman pesan.
5. Pengaruh jumlah *communication node* terhadap keberhasilan pengiriman pesan
6. Pengaruh pergerakan *communication node* terhadap keberhasilan pengiriman pesan

Uji coba dilakukan di di sepanjang jalan depan plasa baru informatika ITS sampai depan tempat perkir UPMB ITS.

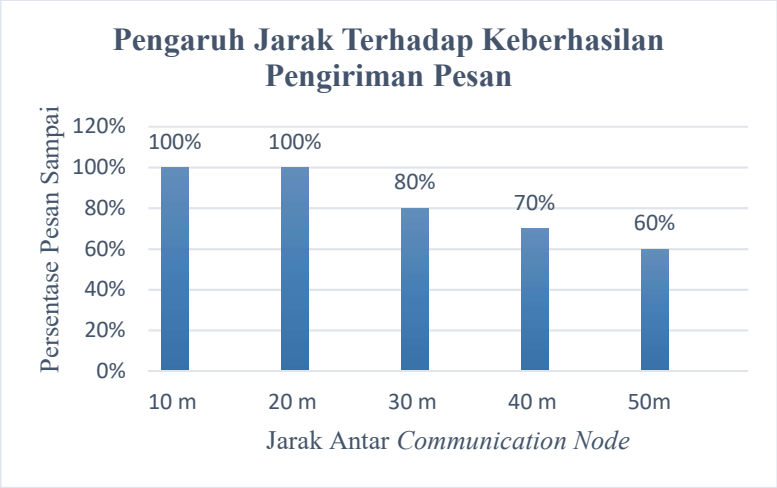
5.3.1 Pengaruh Jarak Terhadap Keberhasilan Pengiriman Pesan

Uji coba ini dilakukan untuk mengetahui keefektifan jarak antar *communication node* dalam pengiriman pesan . Uji coba ini dilakukan dengan mengirimkan pesan kepada *communication node* A ke *communication node* B sebanyak sepuluh pesan dan pada variasi jarak 10 meter, 20 meter, 30 meter, 40 meter dan 50 meter. Gambar 5.16 skenario uji coba pengaruh jarak terhadap keberhasilan pengiriman pesan.



Gambar 5.16 Skenario Uji Coba Pengaruh Jarak Terhadap Keberhasilan Pengiriman Pesan

Hasil Uji coba pengaruh jarak terhadap keberhasilan pengiriman pesan ditunjukkan pada gambar 5.17

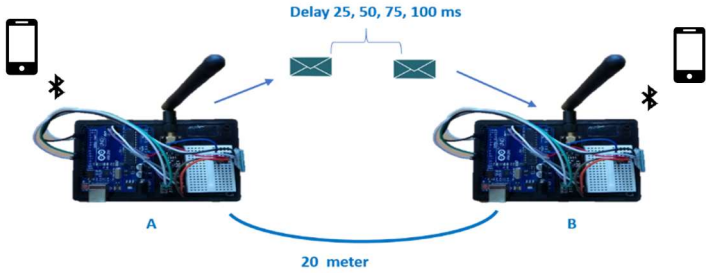


Gambar 5.17 Grafik Hasil Uji Coba Pengaruh Jarak Terhadap Keberhasilan Pengiriman Pesan

5.3.2 Pengaruh Pemberian *Delay* Antara Dua Pesan yang Berurutan Terhadap Keberhasilan Pengiriman Pesan

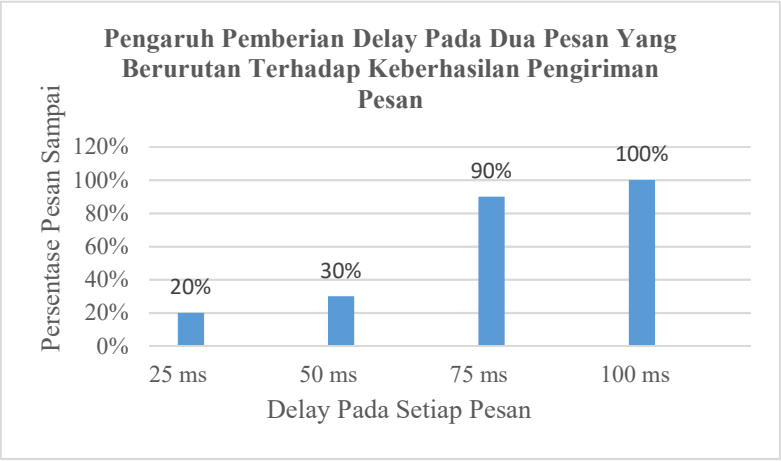
Uji coba pengaruh pemberian *delay* antara dua pesan yang berurutan terhadap keberhasilan pengiriman pesan dilakukan untuk mengevaluasi waktu minimal yang dibutuhkan *communication node* untuk menangani satu pesan yang ditujukan kepadanya. Uji coba ini dilakukan dengan mengirimkan 10 pesan dari *communication node A* ke *communication node* B dengan memvariasikan *delay,* yaitu 25ms, 50ms, 75ms dan 100 ms antara dua pesan yang berurutan dan dilakukan sebanyak empat kali uji coba. Antara *communication node* A dan *communication node* B dipisahkan dengan jarak 20 meter. Pemilihan jarak 20 meter untuk memisahkan *communication node* A dan B karena

berdasarkan hasil uji coba pengaruh jarak terhadap pengiriman pesan, jarak efektif antar *node* dalam pengiriman pesan adalah 20 meter. Gambar 5.18 menunjukkan skenario uji coba pengaruh pemberian *delay* antara dua pesan yang berurutan.



Gambar 5.18 Skenario Uji Coba Pengaruh *Delay* Antara Dua Pesan Yang Berurutan

Hasil uji coba pemberian *delay* antara dua pesan yang berurutan terhadap keberhasilan pengiriman pesan ditunjukkan pada gambar 5.19.

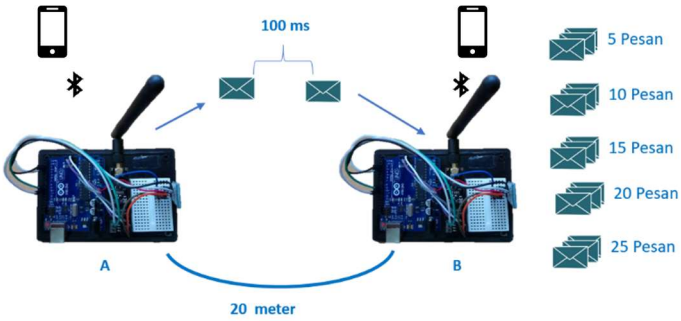


Gambar 5.19 Grafik Hasil Uji Coba Pengaruh Pemberian *Delay* Pada Pesan Terhadap Keberhasilan Pengiriman Pesan

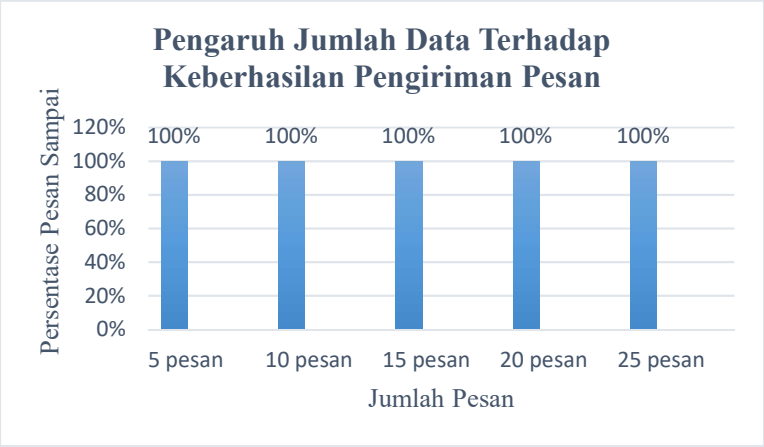
5.3.3 Pengaruh Jumlah Data Terhadap Keberhasilan Pengiriman Pesan

Uji coba pengaruh jumlah data terhadap keberhasilan pengiriman pesan dilakukan untuk mengevaluasi kemampuan *communication node* menangani pesan yang ditujukan kepadanya dalam jumlah banyak dan dalam waktu yang hampir bersamaan. Uji coba ini dilakukan dengan mengirimkan pesan dari *communication node* A ke *communication node* B dalam variasi jumlah pesan sebanyak 5 pesan, 10 pesan, 15 pesan, 20 pesan dan 25 pesan dalam waktu yang hampir bersamaan. Antara *communication node* A dan *communication node* B dipisahkan dengan jarak 20 meter dan antar pesan satu dengan pesan yang lain diberikan waktu delay 100 *millisecond*.

Pemilihan jarak 20 meter untuk memisahkan *communication node* A dan B karena berdasarkan hasil uji coba pengaruh jarak terhadap keberhasilan pengiriman pesan, jarak efektif antar *communication node* adalah 20 meter. Pemilihan *delay* sebesar 100 *ms* sesuai dengan hasil uji coba pemberian *delay* antar dua pesan, *delay* yang efektif antara dua pesan yang berurutan adalah 100 *ms*. Gambar 5.20 menunjukkan skenario dan gambar 5.21 menunjukkan hasil uji coba pengaruh jumlah data terhadap keberhasilan pengiriman pesan.



Gambar 5.20 Skenario Uji Coba Pengaruh Jumlah Data Terhadap Keberhasilan Pengiriman Pesan



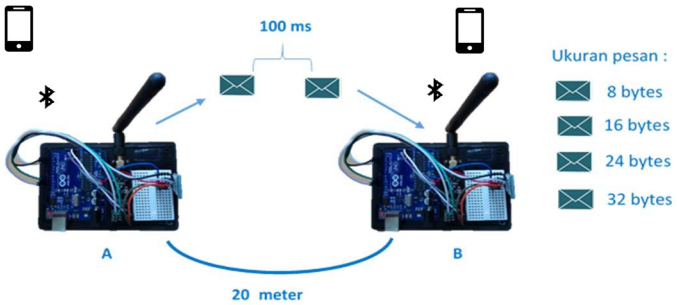
Gambar 5.21 Grafik Hasil Uji Coba Pengaruh Jumlah Data Terhadap Keberhasilan Pengiriman Pesan

5.3.4 Pengaruh Ukuran Data Terhadap Keberhasilan Pengiriman Pesan

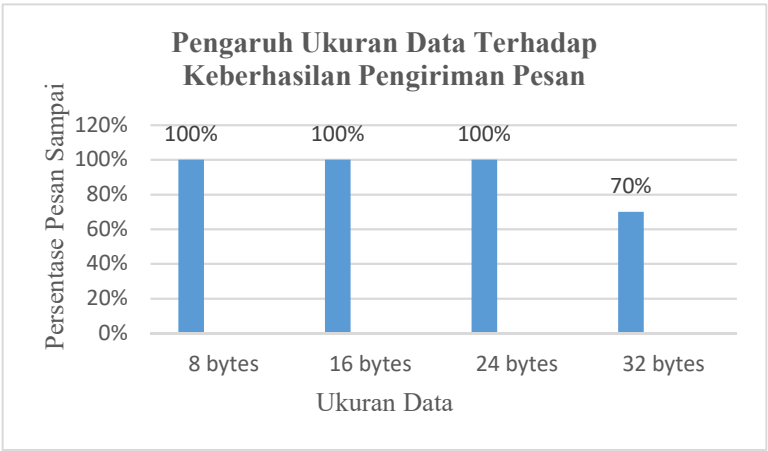
Uji coba pengaruh ukuran data terhadap keberhasilan pengiriman pesan dilakukan untuk mengevaluasi kemampuan *communication node* dalam menangani pesan dengan ukuran yang besar. Uji coba ini dilakukan dengan mengirimkan pesan dari *communication node* A ke *communication node* B dan memvariasikan ukuran data, yaitu 8 bytes, 16 bytes, 24 bytes dan 32 bytes dalam waktu yang hampir bersamaan. Antara *communication node* A dan *communication node* B dipisahkan dengan jarak 20 meter dan antar pesan satu dengan pesan yang lain terdapat waktu delay 100 *millisecond*.

Pemilihan jarak 20 meter untuk memisahkan *communication node* A dan B karena berdasarkan hasil uji coba pengaruh jarak terhadap pengiriman pesan, jarak efektif antar *communication node* adalah 20 meter. Pemilihan *delay* sebesar 100 *millisecond* sesuai dengan hasil uji coba pemberian *delay*

antar dua pesan, *delay* yang efektif antara dua pesan yang berurutan adalah 100 *millisecond.* Gambar 5.22 menunjukkan skenario dan gambar 5.23 menunjukkan hasil uji coba pengaruh ukuran data terhadap keberhasilan pengiriman pesan.



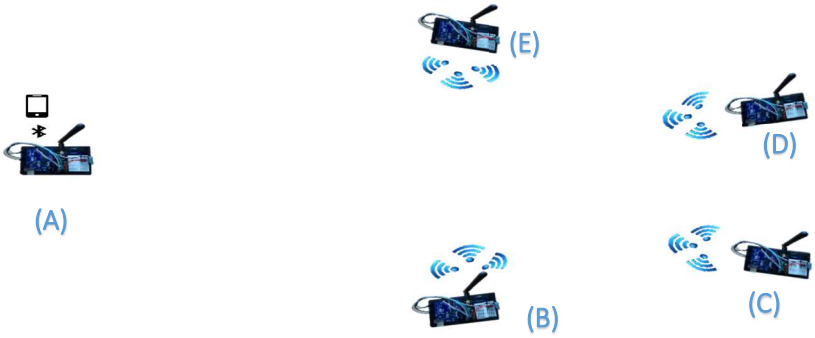
Gambar 5.22 Skenario Pengaruh Ukuran Data Terhadap Keberhasilan Pengiriman Pesan



Gambar 5.23 Grafik Hasil Uji Coba Pengaruh Ukuran Data Terhadap Keberhasilan Pengiriman Pesan

5.3.5 Pengaruh Jumlah *Communication Node* Terhadap Keberhasilan Pengiriman Pesan

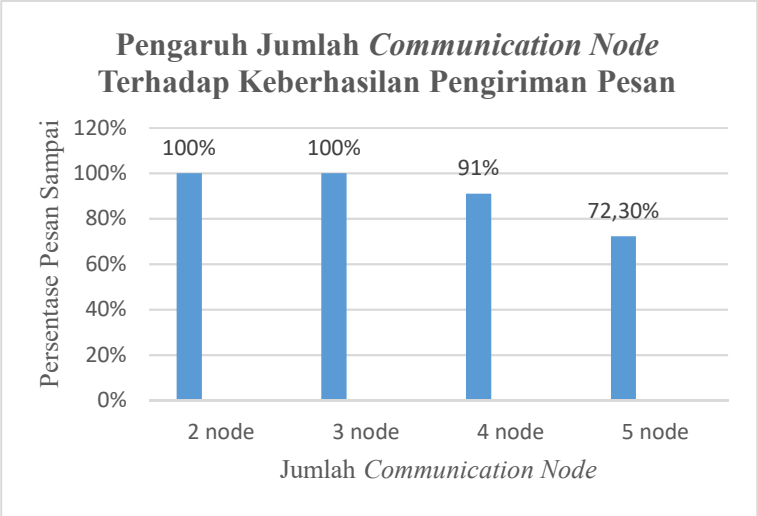
Uji coba pengaruh jumlah *communication node* terhadap keberhasilan pengiriman pesan dilakukan untuk mengevaluasi kemampuan *communication node* dalam menerima *broadcast* pesan dari beberapa *communication node* tetangganya. Uji coba ini dilakukan sebanyak empat kali dengan memvariasikan jumlah *node* pada masing-masing percobaan. Skenario uji coba pengaruh jumlah *communication node* terhadap keberhasilan pengiriman pesan ditunjukkan pada gambar 5.24



Gambar 5.24 Skenario Uji Coba Pengaruh Jumlah *Communication Node* Terhadap Keberhasilan Pengiriman Pesan

Pada percobaan pertama terdapat *node* A dan B, pada percobaan kedua terdapat *node* A,B dan C , pada percobaan ketiga terdapat *node* A,B,C serta D dan percobaan terakhir terdapat *node* A,B,C,D dan E. Setiap *node* pada masing-masing percobaan mengirimkan 100 pesan berbeda dalam waktu bersamaan. *Node* A,B,C,D dan E dapat menjangkau satu sama lain. Antara perngiriman dua pesan yang berurutan terdapat *delay* 100 ms. *Node* A akan menerima semua *broadcast* pesan dari *node* tetangga. Jumlah pesan yang diterima *node* A dicatat sebagai hasil percobaan. Gambar 5.25 menunjukkan hasil dari uji coba

pengaruh jumlah *communication node* pada keberhasilan pengiriman pesan



Gambar 5.25 Hasil Uji Coba Pengaruh Jumlah *Communication Node* Terhadap Keberhasilan Pengiriman Pesan

5.3.6 Pengaruh Pergerakan *Communication Node* Terhadap Keberhasilan Pengiriman Pesan.

Uji coba pengaruh pergerakan *communication node* terhadap keberhasilan pengiriman pesan dilakukan untuk mengevaluasi kemampuan *communication node* untuk menyimpan dan melakukan *broadcast* pesan dalam keadaan bergerak serta *delay* antara pengiriman dan penerimaan pesan. Uji coba ini dilakukan dengan meletakkan *communication node* A dan *communication node* B pada jarak yang tidak bisa menjangkau satu sama lain. Terdapat sebuah *communication node* perantara diantara *communication node* A dan *communication node* B yang bertugas menyampaikan pesan yang

di dapatkan dari *communication node* A ke *communication node* B .Gambar 5.26 menunjukkan lingkungan uji coba pengaruh pergerakan *communication node.*

Gambar 5.26 Lingkungan Uji Coba Pengaruh Pergerakan Communication Node

Terdapat tiga buah *communication node,* yaitu *communication node* A, B dan C. *Communication node* A diletakkan diujung parkir mobil mahasiswa Informatika ITS dan *communication node* B diletakkan diujung parkir motor mahasiswa Informatika ITS sebagaimana pada gambar 5.18. Perangkat android yang terhubung dengan *communication node* A berperan sebagai pengirim pesan, sedangkan perangkat android yang terhubung dengan *communication node* B berperan sebagai penerima pesan. Pesan diberikan *lifetime* selama 10 menit dan masing-masing *communication node* melakukan *broadcast* pesan

yang tersimpan setiap dua menit sekali.

Pada mulanya *communication node* C berada pada jarak kurang dari 10 meter dari*communication node* A. Sehingga ketika *communication node* A melakukan *broadcast* pesan yang ditujukan pada *communication node* B, *communication node* C juga menerima *broadcast* pesan tersebut. Setelah menerima *broadcast* pesan dari *communication node* A, seorang *user* yang sedang membawa *communication node* C berjalan meyusuri garis yang berwarna kuning. Jarak antara *communication node* A dengan communication node B menggunakan jalur berwarna kuning sebesar 155 meter dan user berjalan dengan kecepatan 5km/jam.

Ketika *communication node* C mendekati *communication node* A, maka pesan yang dibawa oleh *communication node* C sampai ke *communication node* B. Pada uji coba ini dilakukan tiga kali percobaan, masing-masing percobaan dicatat waktu ketika *communication node* A pertama kali melakukan broadcast pesan yang ditujukan ke *communication node* B dan waktu *communication node* B menerima pesan yang ditujukan kepadanya. Tabel 5.9 menunjukkan hasil percobaan dari uji coba pengaruh kecepatangerak *communication node* terhadap pengiriman pesan.

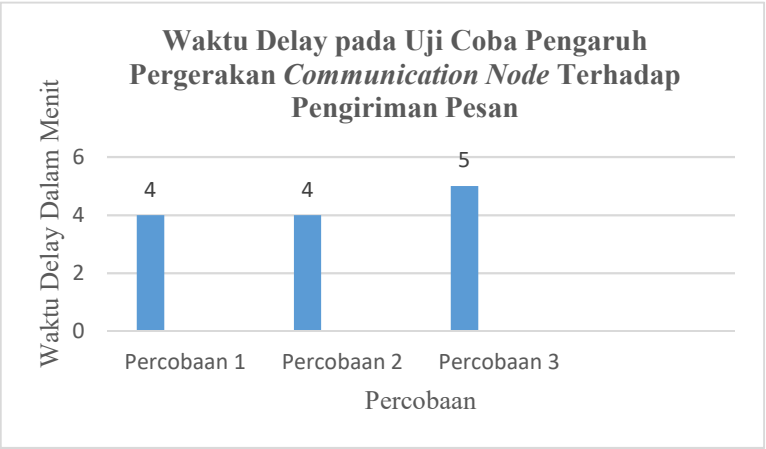
Tabel 5.9 Hasil Uji Coba Pengaruh Pergerakan *Communication Node* Terhadap Keberhasilan Pengiriman Pesan

Percobaan ke	Waktu Kirim	Waktu Sampai	Delay	Persentase Pesan Sampai
1	13.57	14.01	4 menit	100%
2	14.06	14.10	4 menit	100%
3	14.17	14.22	5 menit	100%

Pada uji coba pertama dikirimkan tiga pesan. Pada uji coba kedua dikirimkan lima pesan dan pada uji coba ketiga

dikirimkan 10 pesan. Waktu kirim menunjukkan waktu pesan dari *communication node* A dikirimkan dan waktu terima menunjukkan waktu pesan sampai pada *communication node* B.

Gambar 5.27 menunjukkan grafik waktu *delay* dari uji coba pengaruh pergerakan *communication node* terhadap pengiriman pesan.



Gambar 5.27 Grafik Hasil Uji Coba Pengaruh Pergerakan *Communication Node* Terhadap Pengiriman Pesan

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas mengenai kesimpulan yang dapat diambil dari tujuan pembuatan sistem komunikasi *mobile peer to peer* dengan mengimplementasikan konsep d*elay tolerant network* di dalamnya. Hasil uji coba yang dilakukan sebagai jawaban dari rumusan masalah yang dikemukakan sebelumnya. Selain kesimpulan, terdapat pula saran yang ditujukan untuk pengembangan sistem komunikasi *peer to peer* dengan mengimplementasikan *delay tolerant network* lebih lanjut.

6.1 Kesimpulan

Dalam proses pengerjaan Tugas Akhir mulai dari tahap analisis, desain, implementasi hingga pengujian didapatkan kesimpulan sebagai berikut:

1. *Delay Tolerant Network* dan algoritma *simple encryption* telah di terapkan pada sistem komunikasi *mobile peer to peer* menggunakan modul nRF24L01 dan telah diuji pada dua perangkat android Asus Zenfone Go dan Sony Experia dan berjalan sebagaimana mestinya.
2. Pengiriman pesan antar *communication node* efektif hingga pada jarak 20 meter. Ketika jarak antar *communication node* 30 meter, tingkat keberhasilan pengiriman pesan 80%. Ketika jarak antar *communication node* 40 meter, tingkat keberhasilan pengiriman pesan 70 %. Ketika jarak antar *communication node* 50 meter, tingkat keberhasilan pengiriman pesan 60%.
3. Setiap *communication node* mampu megolah pesan dengan enkripsi sebanyak sepuluh pesan per detik.

6.2 Saran

Berikut merupakan beberapa saran untuk pengembangan sistem komunikasi *mobile peer to peer* dengan mengimplementasikan *delay tolerant network* dimasa yang akan datang berdasarkan pada hasil perancangan, implementasi dan uji coba yang dilakukan.

1. Untuk memperkecil pesan yang sampai pada *communication node* dalam keadaan rusak, perlu diimplementasikan algoritma pengecekan *header* pesan.
2. Ketika jarak antar *communication node* lebih dari 20 meter, perlu dilakukan pengiriman pesan sebanyak dua kali setiap *communication node* melakukan *broadcast* pesan.

DAFTAR PUSTAKA

[1] Ali, S., Qadir, J., & Baig, A. (2010). Routing Protocols in Delay Tolerant Networks – A Survey. *International Conference on Emerging Technologies (ICET)* (p. 70). Islamabad: School of Electrical Engineering and Computer Sciences.

[2] Park, H., Yang, J., Park, J., Kang, S. G., & Choi, J. K. (2008). A Survey on Peer-to-Peer Overlay Network Schemes. *IEEE Internet Computing*, 986.

[3] Singh, M. M., & Mandal, J. K. (2015). Reliability Analysis of Mobile Ad Hoc Network. *International Conference on Computational Intelligence and Communication Networks*, (p. 161). West Bengal.

[4] Suharsono,Awin. 2015 . Pengertian dan Latar Belakang Delay Tolerant Network. [Daring]. Tersedia pada http://aswinsuharsono.lecture.ub.ac.id/files/2012/07/Post1-StoreAndForward.jpg

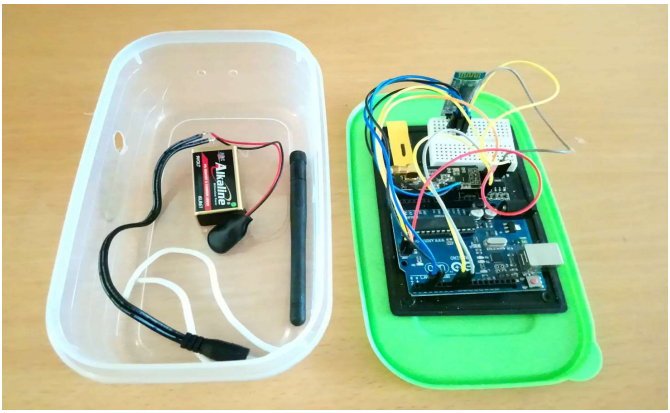
[5] Singh, M. M., & Mandal, J. K. (2015). Reliability Analysis of Mobile Ad Hoc Network. International Conference on Computational Intelligence and Communication Networks, (p. 161). West Bengal.

[6] Wikipedia, "Arduino," Desember. 2018, page Version ID: 847557047. [Daring]. Tersedia pada: https://en.wikipedia.org/w/index.php?title=Arduino&oldid=847557047. [Diakses: Page Version ID: 847557047].

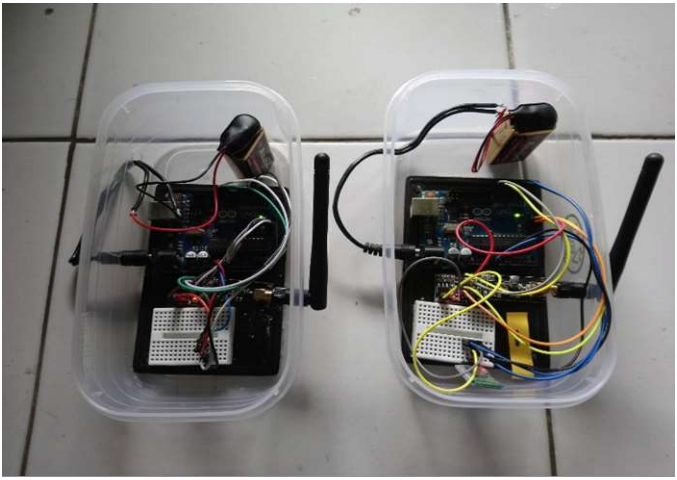
[7] Nordic Semiconductor, “nRF2401 PRODUCT SPECIFICATION.” .

[8] “HC-06 Bluetooth Module”. [Daring]. Tersedia pada https://www.sgbotic.com/index.php?dispatch=products.view&product_id=2471s

LAMPIRAN A
Dokumentasi Pembuatan *Communication Node*



Gambar A.1 Mempersiapkan *Communication Node* dan *Box*

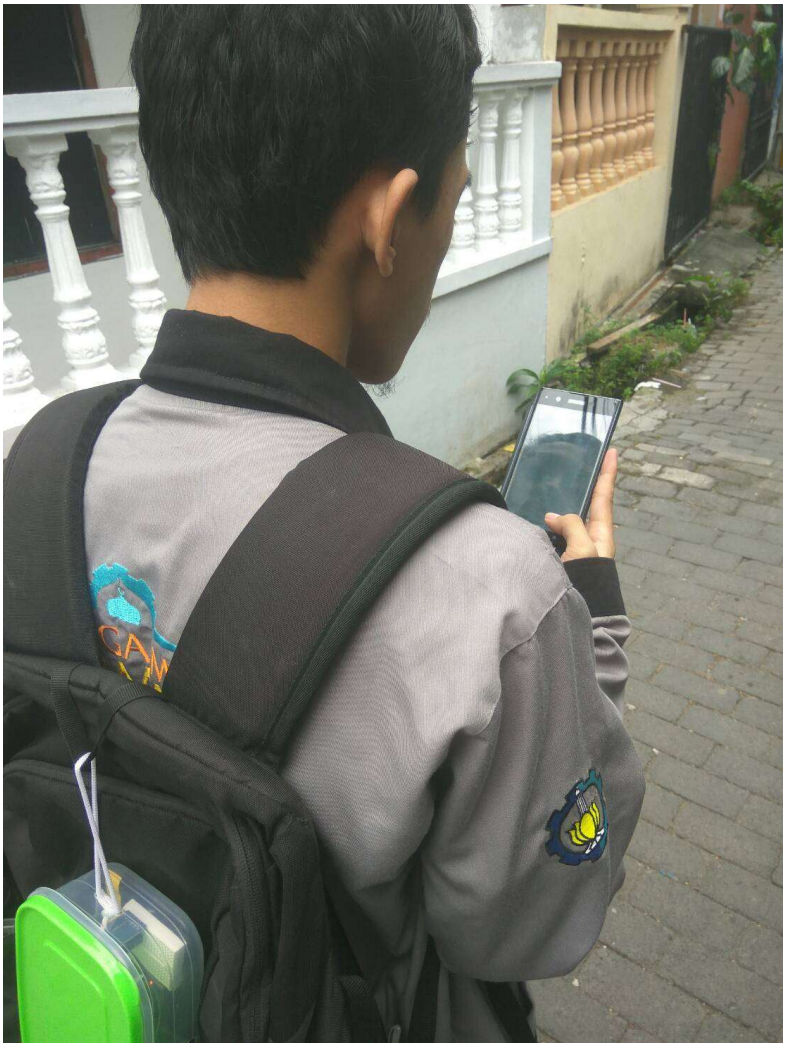


Gambar A.2 Meletakkan *Communication Node* ke dalam *Box*

Gambar A.3 Menutup Box yang Berisi *Communication Node*

Gambar A.4 Proses *Pairing*

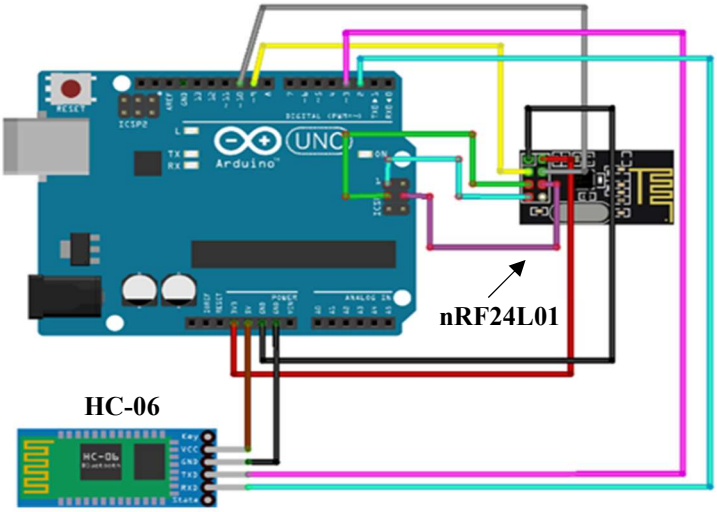
Gambar A.5 Peletakan *Communication Node* Pada Pengguna



Gambar A.6 Pengguna Berkomunikasi melalui Perangkat Android + *Communication Node*

(Halaman ini sengaja dikosongkan)

LAMPIRAN B
Wiring Communication Node



Gambar B.1 Rangkaian Arduino, HC-06 dan nRF24L01

nRF24L01		Arduino
CE	→	Digital 9
CSN	→	Digital 10
MISO	→	ICSP MISO
MOSI	→	ICSP MOSI
SCK	→	ICSP SCK
VCC	→	3,3 V
GND	→	GND

Gambar B.2 *Wiring* Arduino dan nRF24L01

HC-06		Arduino
RX	→	Digital 2
TX	→	Digital 3
VCC	→	5 V
GND	→	GND

Gambar B.3 *Wiring* Arduino dan HC-06

(Halaman ini sengaja dikosongkan)

LAMPIRAN C
Kode Sumber

```
#include <SPI.h>
#include <RF24.h>
#include <SoftwareSerial.h>
#include<CountUpDownTimer.h>

SoftwareSerial bt(3,2);
String pesan_bt = "";
bool set_time = false,bc = false;
const int address = 1000;
const int myaddress = 21;
int xMinute = 0, xHour = 0;
int pesan_ke = 0,lifetime;
boolean id_pesan_masuk[26][30];
CountUpDownTimer T(UP, HIGH);
RF24 radio(7,8);

typedef struct{
  String msg = "";
  int dst;
  int id_msg;
  int start_time;
}Pesan;
Pesan psn[20];
void setup() {
  Serial.begin(9600);
  radio.begin();
  radio.setRetries(15,15);
```

```
  Serial.begin(9600);
  radio.begin();
  radio.setRetries(15,15);
  radio.setPALevel(RF24_PA_LOW);
  radio.enableDynamicPayloads();
  radio.openReadingPipe(0, address);
  radio.openWritingPipe(address);
  radio.startListening();
  bt.begin(9600);
  xMinute = 0, xHour = 0;
  for(int i=21;i<26;i++){
      for(int j=0;j<30;j++){
        id_pesan_masuk[i][j]=false;
      }
    }
}
void countTime() {
  ++xMinute;
  bc=true;
  if (xMinute > 59) {
    ++xHour;
    xMinute = 0;
  }
}
void sentText(String msg_will_send) {
   radio.stopListening();
   int panjang = msg_will_send.length()+1;
   char pesannya[panjang];
   msg_will_send.toCharArray(pesannya,panjang);
```

```
   radio.write(&pesannya,strlen(pesannya) );
   bt.println(msg_will_send);
   radio.startListening();
}

void receiveRF(){
  char pesan_rf[32] = "";
  String dst = "",pre_msg = "",msg_id =
"",time_start = "";
  int len = radio.getDynamicPayloadSize();
  radio.read(&pesan_rf, len);
  pre_msg = String(pesan_rf);
  String msg_id_src = pre_msg.substring(2,4);
  String msg_id_count = pre_msg.substring(4,6);
  int id_src = msg_id_src.toInt();
  int id_count = msg_id_count.toInt();
  Serial.println(pre_msg);
  if(!id_pesan_masuk[id_src][id_count]){
    id_pesan_masuk[id_src][id_count] = true;
     dst = pre_msg.substring(0,2);
     int dst_int = dst.toInt();
     if(dst_int ==myaddress ){
          bt.println(pre_msg);
     }else{
        msg_id = pre_msg.substring(2,6);
        time_start = pre_msg.substring(6,8);
        psn[++pesan_ke].start_time =
         time_start.toInt();
        psn[pesan_ke].id_msg = msg_id.toInt();
        psn[pesan_ke].dst = dst_int;
```

```
        psn[pesan_ke].msg =
         pre_msg.substring(8,pre_msg.length());
      }
      Serial.print("RF : ");
      Serial.println(pre_msg);
  }
}

void receiveBT(){
  char chr;
  while(bt.available()){
     chr = (char)bt.read();
     pesan_bt+=chr;
     delay(3);
  }
  Serial.println(pesan_bt);
  if(!set_time){

       char buf[pesan_bt.length()+1];
       int input[3];
       pesan_bt.toCharArray(buf, sizeof(buf));
       char *p = buf;
       char *str;
       int i=0;
       while ((str = strtok_r(p, "/", &p)) != NULL)
        {
          input[i] = atoi(str);
          i++;
       }
      xHour = input[0];
      xMinute = input[1];
```

```
      lifetime = input[2];
      T.StartTimer();
      set_time = true;
      pesan_bt="";

  }else{
     Serial.println("kirim pesan");
     sentText(pesan_bt);
     pesan_bt = "";
  }
}

void cekMessageTime(){
  if(T.TimeHasChanged()){
      if (T.ShowSeconds()==0){
        countTime();
      }
      if((xMinute%2)==0 && bc){
        Serial.println("cek isi buffer");
        for(int i=1;i<=pesan_ke;i++){
          String new_start_time="";
          if((xMinute - psn[i].start_time) <=
            lifetime ){ // lifetimenya 5 menit
            if(psn[i].start_time < 10){
              new_start_time =
                "0"+String(psn[i].start_time);
            }else{
              new_start_time =
                String(psn[i].start_time);
            }
```

```
            String will_send =
             String(psn[i].dst)+String(
            psn[i].id_msg)+
            new_start_time+psn[i].msg;
            sentText(will_send);
            Serial.println("broadcast");
            Serial.println(will_send);
          }
          delay(10);
        }
        bc=false;
      }

      Serial.print(xHour);
      Serial.print(":");
      Serial.println(xMinute);
  }
}

void loop() {
  T.Timer();
  cekMessageTime();
  if (radio.available()) {
    receiveRF();
  }
  if(bt.available()) {
      receiveBT();
  }
}
```

Kode Sumber C.1 DTN *With Time to Live*

```
#include <SPI.h>
#include <RF24.h>
#include <SoftwareSerial.h>
#include<CountUpDownTimer.h>

SoftwareSerial bt(3,2);
String pesan_bt = "";
char chr;
bool set_time = false,bc = false;
const int address = 1000;
const int myaddress = 21;
int xMinute = 0, xHour = 0;
int pesan_ke = 0,max_hop;
CountUpDownTimer T(UP, HIGH);
RF24 radio(7,8);
typedef struct{
  String msg = "";
  int dst;
  int id_msg;
  int jml_hop;
  boolean is_bc = false;
}Pesan;
Pesan psn[25];
void setup() {
  Serial.begin(9600);
  radio.begin();
  radio.setRetries(15,15);
```

```
  radio.setPALevel(RF24_PA_LOW);
  radio.enableDynamicPayloads();
  radio.openReadingPipe(0, address);
  radio.openWritingPipe(address);
  radio.startListening();
  bt.begin(9600);
  xMinute = 0, xHour = 0;
}
void countTime() {
  ++xMinute;
  bc=true;
  if (xMinute > 59) {
    ++xHour;
    xMinute = 0;
  }
}
void sentText(String msg_will_send) {
   radio.stopListening();
   int panjang = msg_will_send.length()+1;
   char pesannya[panjang];
   msg_will_send.toCharArray(pesannya,panjang);
   radio.write(&pesannya,strlen(pesannya) );
   radio.startListening();
}
void Broadcast(){
  Serial.println("Broadcast");
}
void receiveRF(){
  char pesan_rf[32] = "";
```

```
  boolean is_pernah_terima = false;
  String dst = "",pre_msg = "",msg_id = "",jml_hops
= "";
  int len = radio.getDynamicPayloadSize();
  radio.read(&pesan_rf, len);
  pre_msg = String(pesan_rf);
  diterima ?
  msg_id = pre_msg.substring(2,6);
  int msg_id_int = msg_id.toInt();
  for(int i=1;i<=pesan_ke;i++){
    if(psn[i].id_msg == msg_id_int) {
      is_pernah_terima = true;
    }
  }
  if(!is_pernah_terima){
     dst = pre_msg.substring(0,2);
     int dst_int = dst.toInt();
     if(dst_int ==myaddress ){
          bt.println(pre_msg);
     }else{
        jml_hops = pre_msg.substring(7,8);
        psn[++pesan_ke].jml_hop = jml_hops.toInt();
        psn[pesan_ke].dst = dst_int;
        psn[pesan_ke].id_msg = msg_id_int;
        psn[pesan_ke].msg = pre_msg.substring(
       8,pre_msg.length());
      }
      Serial.print("RF : ");
      Serial.println(pre_msg);
```

```
  }
}
void receiveBT(){
  while(bt.available()){
     chr = (char)bt.read();
     pesan_bt+=chr;
     delay(3);
  }
  if(!set_time){
       char buf[pesan_bt.length()+1];
       int input[3];
       pesan_bt.toCharArray(buf, sizeof(buf));
       char *p = buf;
       char *str;
       int i=0;
       while ((str = strtok_r(p, "/", &p)) != NULL)
{
         input[i] = atoi(str);
         i++;
       }
      xHour = input[0];
      xMinute = input[1];
      max_hop = input[2];
      T.StartTimer();
      set_time = true;
      pesan_bt="";
  }else{
     sentText(pesan_bt);
```

```
      pesan_bt = "";
  }
}
void cekMessageTime(){
  if(T.TimeHasChanged()){
      if (T.ShowSeconds()==0){
        countTime();
      }
      if((xMinute%3)==0 && bc){
        Serial.println("TES 1");
        for(int i=1;i<=pesan_ke;i++){
          Serial.println("Tes 2");
          if( psn[i].jml_hop <= max_hop &&
            !psn[i].is_bc ){
            psn[i].jml_hop+=1;
            psn[i].is_bc = true;
            String will_send = String(psn[i].dst)+
           String(psn[i].id_msg)+"-
           "+String(psn[i].jml_hop)+
           psn[i].msg;
            sentText(will_send);
            Serial.print("broadcast:");
            Serial.println(will_send);
          }
          delay(10);
        }
        bc=false;
      }
      Serial.print(xHour);
```

```
      Serial.print(":");
      Serial.println(xMinute);
  }
}
void loop() {
  T.Timer();
  cekMessageTime();
  if (radio.available()) {
    receiveRF();
  }
  if(bt.available()) {
      receiveBT();
  }
}
```

Kode Sumber C.2 DTN *With Encounter Count*

```
#include <SPI.h>
#include <RF24.h>
#include <SoftwareSerial.h>
#include<CountUpDownTimer.h>

SoftwareSerial bt(3,2);
String pesan_bt = "";
char chr;
bool set_time = false,bc = false;
const int address = 1000;
const int myaddress = 21;
int xMinute = 0, xHour = 0;
int pesan_ke = 0;
```

```
double lokasiku[2],max_distance;
CountUpDownTimer T(UP, HIGH);
RF24 radio(7,8);
typedef struct{
  String msg = "";
  int dst;
  int id_msg;
  String latitude,longitude;
  boolean is_bc;
}Pesan;
Pesan psn[25];
void setup() {
  Serial.begin(9600);
  radio.begin();
  radio.setRetries(15,15);
  radio.setPALevel(RF24_PA_LOW);
  radio.enableDynamicPayloads();
  radio.openReadingPipe(0, address);
  radio.openWritingPipe(address);
  radio.startListening();
  bt.begin(9600);
  xMinute = 0, xHour = 0;
}
void countTime() {
  ++xMinute;
  bc=true;
  if (xMinute > 59) {
    ++xHour;
    xMinute = 0;
```

```
  }
}
void sendText(String msg_will_send) {
   radio.stopListening();
   int panjang = msg_will_send.length()+1;
   char pesannya[panjang];
   msg_will_send.toCharArray(pesannya,panjang);
   radio.write(&pesannya,strlen(pesannya) );
   radio.startListening();
}
void receiveRF(){
  char pesan_rf[32];
  boolean is_pernah_terima = false;
  String dst = "",pre_msg = "",msg_id =
"",time_start = "";
  int len = radio.getDynamicPayloadSize();
  radio.read(&pesan_rf, len);
  String pre = String(pesan_rf);
  pre_msg = pre.substring(0,len);
  msg_id = pre_msg.substring(2,6);
  int msg_id_int = msg_id.toInt();
  for(int i=1;i<=pesan_ke;i++){
    if(psn[i].id_msg == msg_id_int) {
      is_pernah_terima = true;
    }
  }
  if(!is_pernah_terima){
     dst = pre_msg.substring(0,2);
     int dst_int = dst.toInt();
```

```
      if(dst_int == myaddress ){
           bt.println(pre_msg);
           Serial.print("terima : ");
           Serial.println(pre_msg);
      }else{
         psn[++pesan_ke].latitude =
        pre_msg.substring(6,11);
         psn[pesan_ke].longitude =
        pre_msg.substring(11,16);
         psn[pesan_ke].msg =
        pre_msg.substring(16,len);
         psn[pesan_ke].id_msg = msg_id_int;
         psn[pesan_ke].dst = dst_int;
         psn[pesan_ke].is_bc = false;
       }
  }
}
void receiveBT(){
  while(bt.available()){
     chr = (char)bt.read();
     pesan_bt+=chr;
     delay(3);
  }
  if(!set_time){
       char buf[pesan_bt.length()+1];
       int input[3];
       String location[2];
       pesan_bt.toCharArray(buf, sizeof(buf));
       char *p = buf;
```

```
        char *str;
        int i=0;
        while ((str = strtok_r(p, "/", &p)) != NULL)
       {
          if(i>2){
           location[i-3] = String(str);
          }else{
           input[i] = atoi(str);
          }
          i++;
        }
       xHour = input[0];
       xMinute = input[1];
       max_distance = input[2];
       lokasiku[0] = location[0].toDouble();
       lokasiku[1] = location[1].toDouble();
       T.StartTimer();
       set_time = true;
       pesan_bt="";
  }else{
     sendText(pesan_bt);
     pesan_bt = "";
  }
}
double hitungJarak(String lat1,String lon1){
  String latitude_pesan = "-7."+lat1;
  String longitude_pesan = "112."+lon1;
  double lat_pesan = latitude_pesan.toDouble();
  double long_pesan = longitude_pesan.toDouble();
```

```
  double lat2 = lokasiku[0];
  double lon2 = lokasiku[1];
  double R = 6371; // Radius of the earth in km
  double dLat = deg2rad(lat2-lat_pesan);
  double dLon = deg2rad(lon2-long_pesan);
  double a =
    sin(dLat/2) * sin(dLat/2) +
    cos(deg2rad(lat_pesan)) * cos(deg2rad(lat2)) *
    sin(dLon/2) * sin(dLon/2);
  double c = 2 * atan2(sqrt(a),sqrt(1-a));
  double d = R * c * 1000; // Distance in m
  return d;
}

double deg2rad(double deg) {
  return deg * (PI/180);
}
void cekMessageTime(){
  if(T.TimeHasChanged()){
      if (T.ShowSeconds()==0){
        countTime();
      }
      if((xMinute%2)==0 && bc){
        Serial.println("TES 1");
        for(int i=1;i<=pesan_ke;i++){
          Serial.println("Tes 2");
          double jarak =
hitungJarak(psn[i].latitude,psn[i].longitude);
          if(jarak <= max_distance &&
```

```
           !psn[i].is_bc){
            String will_send = String(psn[i].dst)+
             String(psn[i].id_msg)+
             psn[i].latitude+
             psn[i].longitude+psn[i].msg;
             psn[i].is_bc = true;
             sendText(will_send);
          }
          delay(10);
        }
        bc=false;
      }
  }
}
void loop() {
  T.Timer();
  cekMessageTime();
  if (radio.available()) {
    receiveRF();
  }
  if(bt.available()) {
      receiveBT();
  }
}
```

Kode Sumber C.3 DTN *With Max Distance*

BIODATA PENULIS

Dely Teja Mukti, biasa dipanggil Dely, lahir pada tanggal 22 Maret 1997 di Kota Kediri, Jawa Timur. Penulis merupakan seorang mahasiswa yang sedang menempuh pendidikan S1 di Departemen Informatika Institut Teknologi Sepuluh Nopember. Penulis merupakan putra dari pasangan Bapak M. Ali dan Ibu Sulistyowati. Selama menempuh pendidikan di Departemen Informatika, Penulis juga aktif dalam oraganisasi kemahasiswaan, Diantaranya ialah Staf Departemen Teknologi Himpunan Mahasiswa Teknik Computer-Informatika, staf NPC Schematics 2016 pada tahun kedua, dan ketua Departemen Syiar Keluarga Muslim Informatika ITS pada tahun ketiga.