IMPLEMENTASI TIME SYNCHRONIZATION PADA WSN UNTUK METODE TDMA MENGGUNAKAN ALGORITMA TPSN

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Ardy Novian Erwanda
NIM: 125150301111019

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2016

PENGESAHAN

IMPLEMENTASI TIME SYNCHRONIZATION PADA WSN UNTUK METODE TDMA MENGGUNAKAN ALGORITMA TPSN

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Ardy Novian Erwanda
NIM: 125150301111019

Skripsi ini telah diuji dan dinyatakan lulus pada
28 Juli 2016
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Sabriansyah Rizqika Akbar, S.T, M.Eng	Aswin Suharsono, S.T, M.T.
NIP: 19820809 201212 1 004	NIK: 201102 840919 1 001

Mengetahui
Ketua Jurusan Teknik Informatika

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

PERNYATAAN ORISINALITAS

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

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

Malang, 28 Juli 2016

Ardy Novian Erwanda

NIM: 125150301111019

KATA PENGANTAR

Puji syukur kehadirat Allah SWT yang telah melimpahkan rahmat, taufik dan hidayah-Nya sehingga laporan skripsi yang berjudul "Implementasi Time Synchronization pada WSN untuk metode TDMA menggunakan algoritma TPSN" ini dapat terselesaikan.

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

1. Allah SWT atas kehendak dan nikmat-Nya laporan skripsi ini telah selesai dengan baik
2. Ibu Retno Mustikowati, Ibunda tercinta yang selalu men-*support* penulis melalui doa-doa yang dipanjatkan
3. Bapak Sabriansyah Rizqika Akbar, S.T, M.Eng selaku dosen pembimbing pertama dan Kaprodi Teknik Komputer yang tak henti-hentinya memberikan *support* dan bimbingannya kepada penulis untuk segera menyelesaikan skripsi ini.
4. Bapak Aswin Suharsono, S.T, M.T. selaku dosen pembimbing kedua yang selalu membantu ananda dalam menuntaskan laporan skripsi ini.
5. Ulfa Kurniawati yang senantiasa mendukung dan selalu membantu selama proses pembuatan skripsi.
6. Bapak Kambali, ayahanda penulis, dan Tiffany, adik penulis yang senantiasa mendoakan keberhasilan penyusunan skripsi ini.
7. Bapak Adharul Muttaqin,S.T., M.T selaku Mantan Kaprodi Teknik Komputer serta segenap Bapak/Ibu Dosen, Staf Administrasi dan Perpustakaan Program Studi Teknik Komputer Universitas Brawijaya.
8. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D selaku Ketua Program Studi Informatika
9. Teman-teman tercinta di program studi Teknik Komputer 2012 yang selalu memberikan semangatnya sehingga ananda dapat menyelesaikan skripsi ini.
10. Dan orang-orang yang selalu mensupprot dan mendokan penulis yang tidak dapat penulis ucapakan satu persatu. Terimakasih atas semua doa dan support baik materil maupun non meteril.

Penulis mengucapkan banyak terima kasih dan penulis sadar akan adanya beberapa kekurangan pada laporan ini. Jadi penulis berharap adanya penyempurnaan dari pihak-pihak terkait. Semoga laporan ini dapat memberikan manfaat dan referensi untuk melakukan penelitan dalam penyempurnaan sistem.

Malang, 28 Juli 2016

Penulis

ardy.erwanda@gmail.com

ABSTRAK

WSN telah menjadi peralatan pengiriman data yang populer saat ini, dikarenakan sifatnya yang lebih mudah untuk diimplementasikan dan praktis serta tidak banyak memakan biaya jika dibandingkan jaringan kabel biasa. Namun, WSN juga memiliki kekurangan, yaitu permasalahan interferensi pada pengiriman data. Salah satu metode untuk menyelesaikannya adalah menggunakan metode TDMA. TDMA bekerja dengan membuat banyak *node* untuk dapat mengirimkan data secara bergantian, demi menghindari adanya interferensi pada pengiriman data. Untuk mengaplikasikan metode TDMA, dibutuhkan metode sinkronisasi waktu untuk menyamakan waktu lokal setiap *node* dalam WSN. Dalam jaringan berkabel, proses sinkronisasi waktu tidak menemui kendala yang signifikan, namun dalam WSN, akan mendapati beberapa kendala seperti *delay* propagasi yang cukup besar. Dari masalah yang ada, maka diterapkan salah satu metode sinkronisasi waktu dalam WSN yaitu TPSN, supaya metode TDMA bisa dijalankan dengan baik. Perancangan sistem akan menggunakan Arduino Nano, NRF24L01, dan laptop. Skenario pengujian berdasarkan pada waktu yang dibutuhkan pada masing-masing tahapan implementasi. Terdapat 3 tahapan yang diamati pada pengujian, yaitu *discovery*, sinkronisasi, dan TDMA. Dari ketiga tahapan tersebut, didapatkan hasil yang menunjukkan bahwa pembentukan hierarki sampai level 2 membutuhkan waktu kurang dari 10 detik. Dengan rata-rata level 1 kurang dari 2 detik, dan level 2 kurang dari 9 detik. Algoritma TPSN yang diimplementasikan sudah berhasil membuat *node* WSN tersinkronisasi dan pengiriman data berjalan sesuai *time slot* yang telah dibuat. Apabila terjadi perubahan struktur hierarki tidak akan mempengaruhi keberhasilan algoritma TPSN dan pengiriman data pada pengujian TDMA, selama persediaan time slot masih ada. Keakuratan waktu penerimaan data dengan jadwal TDMA memiliki selisih maksimal 90 milidetik.

Kata Kunci— *Time Synchronization*, TPSN, TDMA, *node* WSN, interferensi, *time slot*

ABSTRACT

Wireless Sensor Node (WSN) has become a popular data transmission device nowadays, due to it's characteristics such as easy to implement, flexible, and cheap, rather than another common wired network. However, WSN still have some issues that need to be solved, such as interference in data transmission if there are many WSN nodes send data at the same time and using the same frequency channel, the data collision will occur. One of many solutions to solve this issue is Time Division Multiple Acces (TDMA) method. TDMA works by making those WSN nodes to send their own data sequentiallly, so the nodes will send the data at a different time according to the slot time they have got before. In order to apply the TDMA method, the local clock of the entire WSN nodes must be synchronized. There are significant issues in wired network time synchronization, nevertheless, the issue will appear in WSN. because the propagation delay in WSN is too long, even the small area. Due to this issue, the aim of this research is to apply one of the best time synchronization algorithm called Time-sync Protocol in Sensor Network (TPSN) to support the TDMA method. The hardware implementation used Arduino Nano, NRF24L01, and a laptop. The system was tested, and The results showed that the TPSN algorithm could help the TDMA method made the WSN nodes' time to be synchronized, and sent data at the appropriate time slot. Discovery Phase for level-2-hierarchy construction needs 10 seconds to be completed. The time accuracy of data transmission is 90 milliseconds from TDMA schedule (maximum).

Keyword: Time Synchronization, TPSN, TDMA, WSN node, interference, time slot

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar belakang

Wireless sensor network (WSN) telah menjadi perhatian hampir di seluruh dunia dalam beberapa tahun terakhir ini. Dimana banyak sekali penelitian yang dilakukan terkait dengan perkembangan WSN ini dari berbagai aspek, dikarenakan banyaknya keterbatasan pada jaringan nirkabel jika dibandingkan dengan jaringan kabel. Salah satu bentuk pengembangan teknologi ini adalah dari sisi protokol akses, misalnya *Time Division Multiple Access* atau TDMA yang membutuhkan metode sinkronisasi waktu atau disebut juga dengan *Time Synchronization*.

Dengan protokol TDMA, memungkinkan bagi modul perangkat WSN untuk meningkatkan jumlah data yang bisa ditransmisikan secara nirkabel dalam spektrum frekuensi yang sama (Sohraby & Minoli & Znati, 2007). TDMA akan membuat banyak *node* untuk dapat mengrimkan data secara bergantian, demi menghindari adanya tubrukan atau interferensi data. Untuk bisa menggunakan TDMA, terlebih dahulu perlu digunakan metode *Time Synchronization* guna menyamakan *clock* waktu dari semua perangkat WSN yang diletakkan secara terpisah satu sama lain.

Time Synchronization adalah kunci dari berbagai aplikasi dan sistem operasi yang digunakan pada sistem komputasi terdistribusi. Beberapa protokol telah dikembangkan dan digunakan untuk *Time Synchronization* jaringan kabel maupun nirkabel. Dengan banyak keterbatasan, jaringan nirkabel –atau dalam hal ini WSN- membutuhkan lebih banyak perhatian dan pengembangan daripada jaringan kabel tradisional. Berdasarkan berbagai aplikasi yang diterapkan, presisi waktu WSN yang ditingkatkan hanya beberapa milidetik, dapat meningkatkan performa aplikasi tersebut secara signifikan (Simon et al, 2004)

Namun, beberapa modul perangkat WSN seperti NRF24L01, tidak memiliki fitur otomatis dalam hal *Time Synchronization* tersebut. Untuk itu, perlu dilakukan pembuatan secara manual kode program yang bisa mengatur mekanisme *Time Synchronization* dalam TDMA WSN seperti dijelaskan di atas. Beberapa algoritma berhasil ditemukan dan diuji coba pada peralatan WSN.

Berbagai penelitian tentang *Time Synchronization* WSN telah banyak dipublikasikan. Diantaranya adalah penelitian berjudul *Post Facto Synchronization* yang dilakukan oleh Elson dan Estrin, pada pendekatan tersebut, setiap *clock node* secara alami tidak tersinkronkan, sebuah *node beacon* secara periodik melakukan *broadcast* pesan kepada *node-node* sensor yang berada dalam jangkauannya. Ketika sebuah even terdeteksi, tiap *node* melakukan *record* terhadap waktu even tersebut, membandingkannya dengan waktu lokal masing-masing. Sebuah algoritma bernama *Reference Broadcast Synchronization* atau RBS, menjadi hasil bagi penelitian ini (Elson, 2002). Algoritma tersebut dijalankan dengan model topologi jaringan *star*, dimana *node beacon* (kemudian dinamakan *node root*) adalah pusat dari topologi ini. Dalam satuan waktu tertentu, *node beacon* akan melakukan pengiriman waktu even secara periodik. Kemudian setelah semua *node* menerima, untuk mengurangi jumlah *delay* –yang bisa mengurangi tingkat

presisinya– , semua *node* penerima akan saling mencocokkan waktu even satu sama lain. Algoritma akan ini memiliki kelemahan signifikan, jika dilakukan pada jaringan WSN skala besar, karena *node beacon* tidak akan bisa menjangkau semua *node*. Beberapa penelitian terkait yang dilakukan oleh Yoon, et al (2010) menggunakan algoritmanya bernama Tiny-Sync, mencoba memperbaiki sistem perhitungan waktu RBS lebih akurat. Namun, tidak membahas mengenai pengembangan dalam skala yang lebih besar.

Berdasarkan pemaparan di atas, penulis berminat untuk melanjutkan penelitian dengan menggunakan algoritma TPSN yang menggunakan topologi *tree*, sehingga bisa diterapkan dalam skala yang lebih besar. Penelitian dilakukan dengan modul WSN NRF24L01 yang akan diprogram untuk implementasi algoritma tersebut.

1.2 Rumusan masalah

Berdasarkan latar belakang yang telah dikemukakan di atas, maka rumusan masalah dapat disusun sebagai berikut:

1. Bagaimana mengimplementasikan metode sinkronisasi waktu pada modul Arduino dan NRF24L01 dengan memanfaatkan metode TPSN?
2. Bagaimana kinerja metode TPSN dalam melakukan sinkronisasi waktu ditinjau dari nilai akurasi waktu?

1.3 Tujuan

Tujuan yang ingin dicapai dari penelitian ini adalah:

- Tujuan Umum
  Menambahkan fitur pada NRF24L01 dengan mekanisme protokol *schedulling*, untuk mengatur pengiriman data *node-node* WSN yang menggunakan kanal frekuensi yang sama
- Tujuan khusus
  1. Mengimplementasikan algoritma TPSN untuk mendukung metode TDMA sehingga tidak terjadi interferensi pada pengiriman data.
  2. Menguji tingkat keberhasilan pengiriman data WSN dengan penjadwalan yang diatur oleh metode TDMA.

1.4 Manfaat

Manfaat yang ingin dicapai dari penelitian ini adalah:

1. Bagi penulis
   a. Dapat menambah pengetahuan dan wawasan, serta dapat mengaplikasikan teori yang diperoleh selama perkuliahan.
   b. Dapat mendapatkan pengalaman lebih dalam mengkaji teknologi yang dapat digunakan sebagai awal mula perancangan modul WSN yang *framework*nya perlu dibuat secara manual.
2. Bagi pengguna
   a. Dapat menggunakan hasil penelitian ini untuk dikembangkan maupun ditambahkan dengan framework lain demi pengembangan Tiny-OS pada modul WSN NRF24L01.

b. Menjadi user yang ambil andil dalam pemberian saran dan kritik yang membangun.

1.5 Batasan masalah

Berdasarkan latar belakang dan rumusan masalah yang telah dikemukakan sebelumnya, maka penelitian ini memiliki batasan-batasan masalah sebagai berikut:

- Penelitian dilakukan hanya menggunakan *radio frequency* NRF24L01, sebagai modul *wireless node*
- Mikrokontroler yang digunakan pada tiap *node* adalah Arduino Nano.
- Pengujian dilakukan dalam ruangan dengan kondisi setiap *node* dapat tersambung dengan minimal satu *node* lain secara langsung tanpa penghalang.
- Percobaan dilakukan dengan *node* berjumlah empat buah, dengan 1 buah *node* sebagai *root* yang akan digunakan sebagai acuan waktu bagi *node* lain.
- *Node root* ditentukan langsung oleh penulis dan memiliki kode program yang berbeda dengan *node* lain.
- Penelitian dilakukan dengan penitikberatan pada algoritma *time synchronization*, tanpa fokus mendalam pada metode TDMA tertentu yang digunakan.

1.6 Sistematika pembahasan

Penyusunan proposal skripsi ini menggunakan kerangka pembahasan yang tersusun sebagai berikut:

BAB 1 Pendahuluan

Menjelaskan tentang latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah, dan sistematika pembahasan dari "Penerapan *Time Synchronization* untuk TDMA pada modul WSN menggunakan algoritma TPSN ".

BAB 2 Landasan Kepustakaan

Menguraikan tentang referensi dan dasar teori yang mendasari tentang algoritma TPSN untuk mengatur waktu penjadwalan TDMA pada NRF24L01.

BAB 3 Metode Penelitian

Membahas tentang langkah kerja yang dilakukan dalam penulisan diantaranya studi literatur, analisis kebutuhan sistem, implementasi dan pengujian dari perancangan modul *Wireless Sensor Network.*

BAB 4 Rekayasa Persyaratan

Membahas tentang semua kebutuhan atau *requirements* system, serta kebutuhan fungsional dan non fungsional system.

BAB 5 Perancangan dan Implementasi

Membahas tentang perancangan perngkat keras dan lunak yang digunakan dalam penelitian, serta cara pengimplementasiannya.

BAB 6 Pengujian dan Analisis

Membahas tentang pengujian dan analisis dari bab sebelumnya. Pengujian ini dilakakukan dengan menggunakan parameterwaktu proses dan keberhasilan pengiriman data.

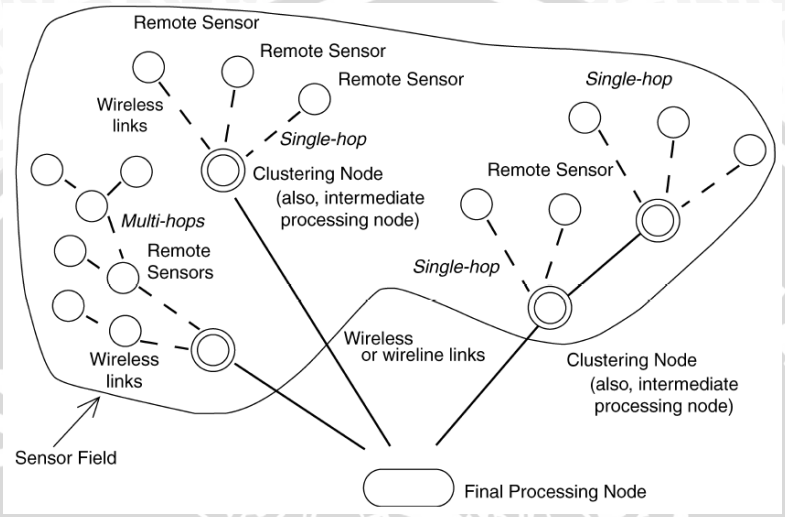
BAB 7 Kesimpulan dan Saran

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

BAB 2 LANDASAN KEPUSTAKAAN

Bab 2 ini akan menguraikan tentang referensi dan dasar teori yang mendasari tentang algoritma TPSN untuk mengatur waktu penjadwalan TDMA pada NRF24L01 beserta kebutuhan perangkat yang digunakan.

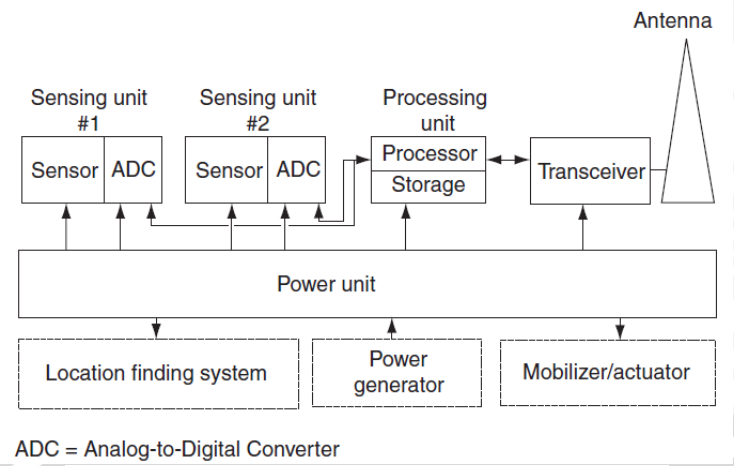
2.1 *Wireless Sensor Network*



Gambar 2.1 Ilustrasi *Wireless Sensor Network*
Sumber: Sohraby, et al. (2007)

WSN singkatan dari *Wireless sensor network* (jaringan sensor nirkabel) adalah suatu jaringan nirkabel yang terdiri dari kumpulan *node* sensor yang tersebar di suatu area tertentu (*sensor field*); serta sebuah *base station,* yang merupakan komponen penerima dan pengumpul semua informasi dari semua *node* untuk kemudian diolah menjadi informasi yang berguna bagi banyak keperluan. Tiap *node* sensor memiliki kemampuan untuk mengumpulkan data lingkungan menggunakan sensor yang sesuai, mengolahnya menjadi data digital, dan mengirimkannya pada *node* sensor lainnya menggunakan protokol tertentu.

Protokol maupun struktur topologi dalam WSN bisa dalam berbagai jenis, tergantung dari keperluan dan struktur maupun kondisi lapangan dimana WSN tersebut diterapkan. Protokol yang digunakan akan mempengaruhi model pengiriman anatar *node.* Seperti pada gambar 2.1, dimana *node* WSN yang tersebar dalam suatu *field,* menerapkan pemodelan pembagian *cluster;* setiap *cluster* akan bertanggung jawab atas anggota *cluster*-nya. Hal ini berguna untuk mengatasi kasus pengiriman seperti *multi-hops*, yang membuat pengiriman antar *node* A dan B dilewatkan pada sebuah *node* lain yang terletak diantara keduanya, karena permasalahan jarak. Sehingga data akan diteruskan pada banyak *node* sampai bisa menjangkau *node* tujuan. Sistem *milti-hops* ini juga banyak diterapkan untuk berbagai keperluan, diantaranya untuk topologi *tree* yang menerapkan sistem hierarki dan pe-*level*-an.



Gambar 2.2 Ilustrasi *Node* WSN secara umum
Sumber: Sohraby, et al. (2007)

Komponen suatu *node* pada WSN ini meliputi sensor, ADC untuk *converter* pada sensor, modul *wireless* atau *transceiver* yang dapat dilengkapi antena penguat, sumber tegangan, *processing unit*, *location finding system,* serta memori seperti pada Gambar 2.2. Seluruh komponen tertanam dalam sebuah *node board*, kemudian berkomunikasi dengan beberapa *node* lainnya secara nirkabel, membentuk suatu jaringan yang memiliki fungsi sebagai sistem monitoring yang mampu bekerjasama mengumpulkan data dari lapangan berupa karakteristik dari sensor. Dalam hal ini, karena WSN dapat digunakan untuk berbagai aplikasi maka penggunaan sensor dapat dipilih sesuai kebutuhan sistem.

Sistem WSN ini lebih jauh efisien dibandingkan dengan penggunaan kabel. Sistem ini memiliki fungsi untuk berbagai jenis aplikasi dimana WSN mampu memenuhi kebutuhan teknologi dalam berbagai bidang ilmu, seperti halnya pada bidang biologi, pertanian, perikanan dan lain sebagainya. (Kazem S, 2007). WSN mampu diterapkan pada lokasi-lokasi yang jaringan berkabel tidak mampu untuk menjangkaunya. Hal ini juga yang menyebabkan WSN menjadi lebih murah karena ketiadaan infrastruktur kabel.

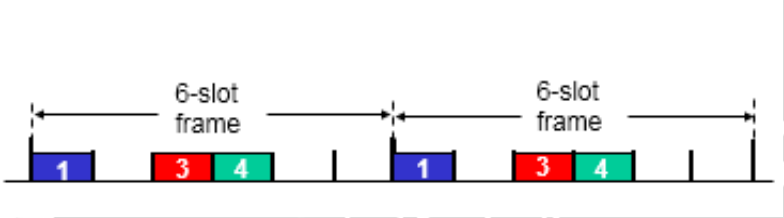
2.2 *Time Division Multiple Access*

Protokol untuk *Link* yang memiliki banyak akses (*Multiple access links*) adalah protokol pada *link layer* yang digunakan untuk mengatur mekanisme pengaksesan jalur ketika terdapat pengiriman beberapa *source node* menuju *destination*-nya masing-masing melalui sebuah media yang sama atau kanal frekuensi yang sama, untuk mencegah adanya *collision* ataupun interferensi dari pengiriman-pengiriman tersebut. Protokol *Multi Access* memiliki tiga mekanisme utama, yaitu *channel partitioning, random access,* dan *taking turns.*

Channel Partitioning adalah mekanisme Protokol *Multi Access* dengan algoritma terdistribusi yang mengatur bagaimana *channel* atau *link* yang digunakan secara bersama-sama untuk dibagi menjadi beberapa "*pieces*" yang selanjutnya akan dialokasikan ke tiap-tiap *node.* Terdapat berbagai metode pada protokol *channel partitioning*, diantaranya, *Frequency Division Multiple Access (FDMA), Time Division Multiple Access (TDMA),* dan *Code Division Multiple Access*

Random Access adalah mekanisme pengiriman *multi access* yang membuat pengiriman tiap *node* pada sembarang waktu, dan akan melakukan pengiriman ulang jika terjadi *collision*. Untuk bisa menerapkan mekanisme *random access,* maka nrf24l01 harus memiliki *library* tambahan yang mengatur *collision detection.* Contoh dari protokol *random access* adalah CSMA, CSMA/CD, dan CSMA/CA.

Taking Turn merupakan mekanisme *multiple access* yang mengatur pengiriman tiap *node* berdasarkan *polling* atau *token* yang dikirimkan oleh sebuah *master node* sebagai "undangan"*,* kemudian *node* penerima baru akan mengirimkan datanya sehingga tidak terjadi pengiriman bersama-sama. Mekanisme ini memiliki beberapa kelemahan seperti *polling/tokens overhead, latency* pada pengiriman, dan *single point of failure.* (Kurose, et al, 2012)



Gambar 2.3 Ilustrasi TDMA
Sumber: (Kurose, et al, 2012)

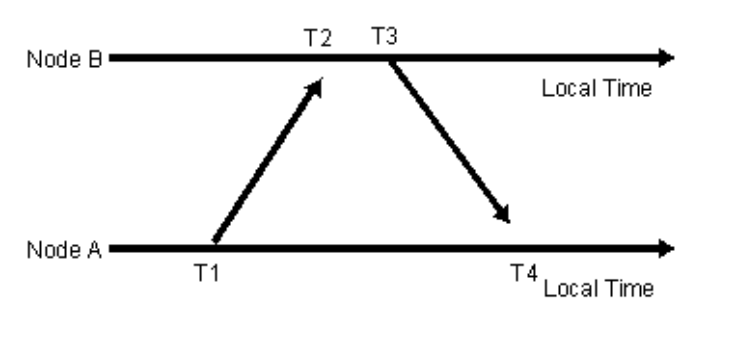
Time Division Multiple Access atau TDMA adalah teknologi untuk peralatan pengiriman data digital menggunakan teknik multipleksi pembagian waktu. TDMA bekerja dengan membagi frekuensi radio ke dalam slot waktu yang dapat mendukung pemakaian aliran kanal data secara bersama-sama. Transmisi dalam bentuk urutan *frame,* dimana tiap frame dibagi menjadi beberapa slot waktu, dan tiap slot waktu bersifat *dedicated* untuk sebuah transmitter tertentu. (Stallings, 2005). Setiap pengirim akan memiliki jadwal pengiriman yang berbeda dengan jadwal pengirim yang lain, dan semua pengirim memiliki waktu yang tersinkronisasi. Seperti ditunjukkan pada Gambar 2.3, terdapat 6 *time slot* pada sebuah *frame,* dimana beberapa slot digunakan oleh pengirim 1, 3, dan 4; sedangkan slot 2,5, dan 6 sedang dalam kondisi *idle.* Pengiriman kembali dilakukan pada *frame* yang lainnya.

2.3 *Time-sync Protocol for Sensor Network*

Time-Sync Protocol for Sensor Networks TPSN adalah sebuah algoritma sinkronisasi waktu pada WSN, yang menggunakan pemodelan topologi *tree*. Dimana terdapat sebuah *node* sebagai *root* yang berada di pangkal *tree*, sinkronisasi waktu dimulai dari *root* tersebut. Dengan menggunakan aliran data yang disebut *timestamp* yang dikirim dari *root* ke hierarki dibawahnya, waktu disamakan melalui *timestamp* tersebut. TPSN memiliki keunggulan dibandingkan beberapa algoritma lain seperti skala yang lebih besar daripada algoritma RBS (Alson et al, 2001).

Terdapat dua fase pada algoritma TPSN, fase pertama adalah fase *discovery*, fase ini digunakan saat pertama kali seluruh *node* membentuk hierarki, dimulai dari *node root*. Berlanjut ke *node* level 1, level 2, dan seterusnya. Seluruh

node akan mengingat posisinya sebagai penghuni dari level masing-masing, serta mengenali alamat *parent*-nya. Sistem ini dilakukan dengan pertama kali aliran data paket *discovery* dari *node root* diteruskan sampai ke level paling ujung, sehingga semua *node* masuk ke daftar hierarki *tree.* Fase kedua adalah fase sinkronisasi, pada fase ini, semua *node* akan memulai sinkronisasi waktu. Semua diawali dari *node* dengan level paling kecil terlebih dahulu. *Node level* 1 akan mengirim paket permintaan sinkronisasi atau *request* sinkronisasi pada waktu T1 ke *node root* (level 0)*,* root akan menerima pada waktu T2, akan membalas dengan *acknowledgment packet* atau paket balasan sinkronisasi pada waktu T3, yang berisi nomor hierarki, T1, T2, dan T3. *Node level* 1 akan melakukan perhitungan dari aliran skenario ini dengan melakukan pencocokan waktu. Hal serupa akan dilakukan oleh *node level* 2 kepada *node level* 1, kemudian berulang seterusnya sampai *node level* i yang paling ujung melakukan hal yang sama kepada *node level* i-1. Berikut skenario pada gambar 2.4.

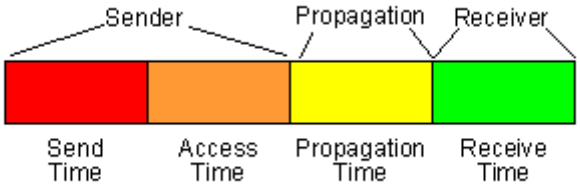


Gambar 2.4 Mekanisme TPSN
Sumber: Roche (2006)

Setelah menerima balasan atas paket *request* sinkronisasi yang dikirimkan, sebuah *node* akan segera melakukan pencocokan waktu dengan menghitung selisih dari *timestamp* yang diperoleh dari *parent*-nya dengan waktu local yang dimiliki. Selisih *timestamp* akan digunakan sebagai penambah atau pengurang pada waktu lokal, tergantung apakah *timestamp* yang diperoleh lebih awal atau lebih akhir dari waktu lokal. Kemudian untuk menghindari perbedaan waktu atau *timestamp* yang disertakan pada paket balasan sinkronisasi akibat dari *delay* yang ada, maka akan dilakukan penambahan *delay* setelah pencocokan waktu lokal.

Terdapat berbagai *delay* selama proses pengiriman berlangsung, yaitu *delay* pengiriman (*send time*), *delay* akses (*access time*), *delay* propagasi (propagation time), dan *delay* penerimaan (*receive time*) seperti ditunjukkan pada Gambar 2.5. *Delay* pengiriman adalah waktu konstruksi data untuk transmisi pada jaringan, *delay* akses adalah waktu dari MAC *layer* untuk mengakses jaringan, *delay* propagasi adalah waktu yang dibutuhkan untuk transmisi melalui medium pengiriman secara fisik, dan *delay* penerimaan adalah waktu yang dibutuhkan *node* penerima untuk memproses data dan melakukan transfer kepada *host.* Dalam komunikasi data pada TPSN, hanya *delay* propagasi yang memiliki angka cukup besar untuk diperhitungkan, sedangkan *delay-delay* lainnya memiliki angka kecil sehingga bisa diabaikan.

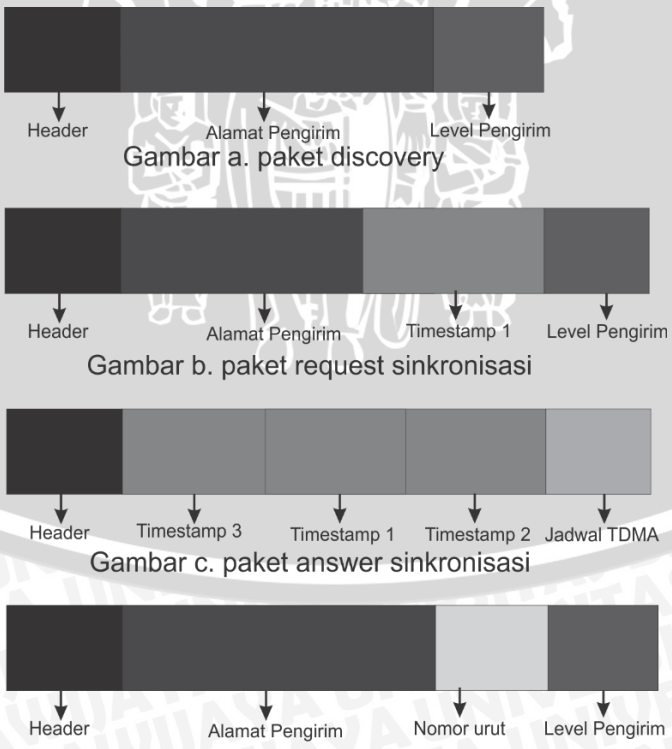
Untuk menghitung *delay* propagasi pada TPSN, seperti ditunjukkan pada persamaan 2.1. Dengan menggunakan skenario pada Gambar 2.4, keseluruhan perhitungan *delay* dilakukan pada *node* A (pe-*request*), untuk kemudian ditambahkan pada *timestamp* yang diterima.



Gambar 2.5 Macam-macam *delay*
Sumber: Roche (2006)

$$d = \frac{(T2-T1)+(T4-T3)}{2} \qquad (2.1)$$

Untuk komunikasi antar *node* pada masing-masing tahapan atau level, digunakan beberapa paket berbeda, tiap-tiap paket dikenali melalui "header" yang merupakan data array urutan pertama dari paket tersebut. Paket *Discovery* berawalan karakter "B"; paket Sinkronisasi berawalan karakter "R" untuk request dan "S" untuk balasan; dan paket TDMA berawalan "T". Sehingga sebuah *node* penerima akan melakukan pegolahan berdasarkan paket yang sesuai. Secara keseluruhan, keempat paket ditunjukkan oleh gambar 2.6 a – d.



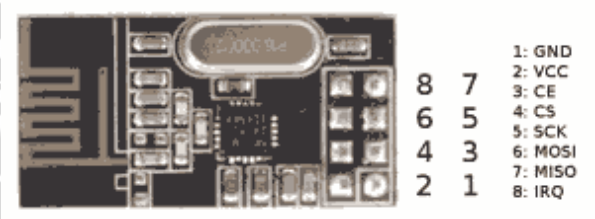
Gambar 2.6 Paket-paket pengiriman

2.4 NRF24L01

NRF24L01 adalah sebuah modul komunikasi jarak jauh yang memanfaatkan pita gelombang rf 2.4GHz ISM (*Industrial Scientific and Medical*). Modul ini menggunakan antarmuka SPI untuk berkomunikasi. Tegangan dari modul ini adalah 3,3 Volt DC. Fitur yang dimiliki modul wireless nRF24L01 adalah sebagai berikut:

- Kecepatan operasi maksimum hingga 2Mbps, modulasi GFSK yang efisiensi, kemampuan Anti-gangguan, Sangat cocok untuk aplikasi kontrol industri.
- *Built-in hardware* dan *link layer*
- 125 opsional saluran kerja
- Saluran waktu *switching* yang sangat singkat, dapat digunakan untuk frekuensi hopping
- Kompatibel dengan seri nRF24XX
- I / O dapat menerima tingkat 5v input
- Pita frekuensi ISM terbuka global, 0dBm ditransmisikan tegangan, bebas lisensi maksimum untuk menggunakan
- Transmisi jarak hingga 100 meter di luar ruangan terbuka

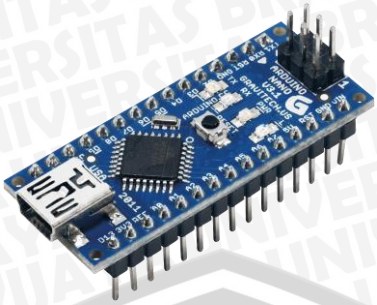
Penampilan modul NRF24L01, seperti ditunjukkan gambar 2.7



Gambar 2.7 Modul *wireless* NRF24L01
Sumber: Chantrell (2013)

2.5 Arduino Nano

Arduino Nano merupakan board mikrokontroler milik Arduino untuk digunakan dalam pemrograman pada penelitian ini. Arduino Nano cocok untuk digunakan sebagai *node* pada modul WSN, karena memiliki beberapa keunggulan, Nano didesain untuk pengguna yang membutuhkan fleksibilitas, keterjangkauan harga, dan ukuran yang kecil (Arduino, 2015). Dalam ukurannya yang kecil, tersedia pin 5 Volt dan 3,3 Volt sehingga bisa digunakan untuk menghidupkan peralatan-peralatan yang membutuhkan tegangan pada kedua nilai tegangan tersebut. Arduino Nano juga dapat bekerja dengan kabel USB Mini-B yang mudah didapat sehingga mempermudah dalam penggunaannya. USB Mini-B tersebut bisa digunakan sebagai suplai daya dan juga sebagai penghubung Arduino Nano dengan PC untuk berkomunikasi. FTDI *chip* yang berperan sebagai penghubung perangkat keras Arduino Nano – PC sudah disertakan langsung secara *embedded* pada *board* Arduino, sehingga tidak memerlukan perangkat keras FTDI. Tampilan Arduino Nano seperti ditunjukkan gambar 2.8.



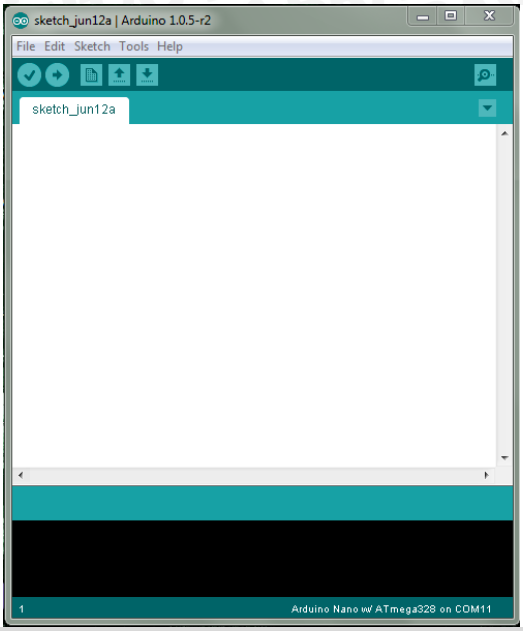
Gambar 2.8 Arduino Nano
Sumber: Arduino (2016)

Berikut fitur lengkap Arduino Nano**:**

- ATmega328
- FTDI on board
- USB connection off board
- Supports auto-reset
- 5V regulator
- Max 150mA output
- Dimensi 45 mm x 18 mm
- Berat 5 gram
- DC *input* 5V sampai 12V
- On board Power and Status LEDs
- Analog Pins: 8
- Digital I/Os: 14

2.5.1 Arduino IDE

Arduino IDE adalah Software resmi Arduino yang digunakan untuk pemrograman semua Mikrokontroler Arduino. Dalam penelitian ini, penulis menggunakan Arduino IDE untuk membuat program dalam bahasa C dan *debugging*-nya, meng-*upload* program ke board Arduino, menambahkan *library*, dan mengamati aktivitas Arduino Nano menggunakan Serial Monitor. Tampilan *interface* Arduino IDE seperti ditujukkan gambar 2.9.

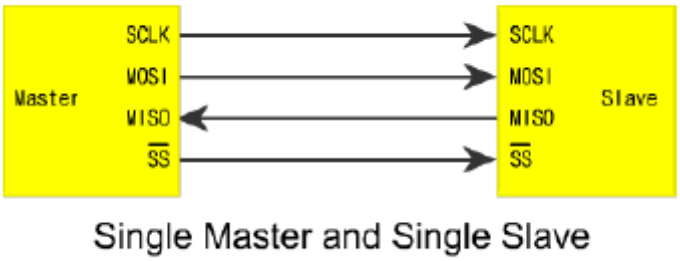


Gambar 2.9 Arduino IDE

2.6 *Serial Peripheral Interface*

Serial Peripheral Interface (SPI) adalah koneksi bus serial sinkronus yang didesain oleh Motorola. Bus SPI merupakan *interface* komunikasi serial dengan 4 kabel *full-duplex*, yang dirancang dengan sistem *Master/Slave.* SPI digunakan untuk komunikasi serial antara mikrokontroler host dengan *peripheral* lain (Mathivanan, 2007). Beberapa *peripheral* yang mendukung SPI diantaranya *converter* ADC/DAC, memori EEPROM, *Real Time Clocks* (RTC), sensor, dan kontroler LCD maupun USB. SPI mengizinkan koneksi langsung dari beberapa *peripheral* tersebut untuk tersambung langsung dengan mikrokontroler. *Transfer rate* dari SPI sebesar 1 Mbps atau 2 Mbps, tergantung dari konfigurasi *Master/Slave* yang digunakan. 4 jalur yang digunakan SPI adalah SCK, MOSI, MISO, dan CS.

Sambungan SPI pada sebuah *Master* dan sebuah *Slave* seperti ditunjukkan pada Gambar 2.10. Untuk memulai komunikasi, bus *master* mengkonfigurasikan *clock* melalui SCK, menggunakan frekuensi yang didukung oleh *slave peripheral.* Kemudian master memilih *slave* dengan logika level 0 pada *select line* (CS). Setiap siklus *clock* SPI, sebuah aliran data *full-duplex* terjadi. *Master* mengirim sebuah bit melalui MOSI dan *slave* menerimanya. Begitu pula sebaliknya, *Slave* mengirimkan bit melaui MISO, dan *Master* menerimanya. Sistem ini akan digunakan untuk pengiriman data pada sambungan antara Arduino Nano dan NRF24L01.



Gambar 2.10 Ilustrasi Sambungan Master dan Slave SPI

Sumber: Emertxe (2014)

2.7 Mirf *Library*

Mirf *Library* adalah modul yang digunakan untuk keperluan pemrograman NRF24L01 pada Arduino. Modul Mirf menyediakan beberapa fungsi *controlling* umum NRF yang bisa langsung dipakai meliputi konfigurasi dan operasional, yang dapat bekerja pada ATMEGA168 dan ATMEGA328 (Arduino, 2013). Ketentuan Mirf ditunjukkan pada Tabel 2.1. Fungsi default Mirf untuk NRF24L01 dan Arduino Nano seperti ditunjukkan pada Tabe 2.2.

Tabel 2.1 Ketentuan Default Mirf

Property	Fungsi	Angka Default
cePin	Kontroler RX/TX	8
csnPin	*Chip Select Not*	7
channel	Kanal frekuensi	0 – 127
payload	Ukuran paket	Default 16, maksimum 32

Tabel 2.2 Fungsi Default Mirf

Fungsi	Deskripsi
void init(void)	Inisialisasi modul, konfigurasi pin, modul SPI
void setRADDR(byte *addr)	Pengaturan alamat penerimaan
void setTADDR(byte *addr)	Pengaturan alamat tujuan pengiriman
void config(void)	Set *channel* dan lebar *payload.* Reset RX mode dan *flush* fifo
bool dataReady(void)	Cek apakah ada data diterima
void getData(byte *data)	Ambil data array data sepanjang ukuran *payload*
void send(byte *data)	Kirim ‘data’
bool isSending(void)	Cek apakah sedang kondisi mengirim
bool rxFifoEmpty(void)	Cek apakah rx fifo kosong
bool txFifoEmpty(void)	Cek apakah tx fifo kosong

Tabel 2.2 (Lanjutan)

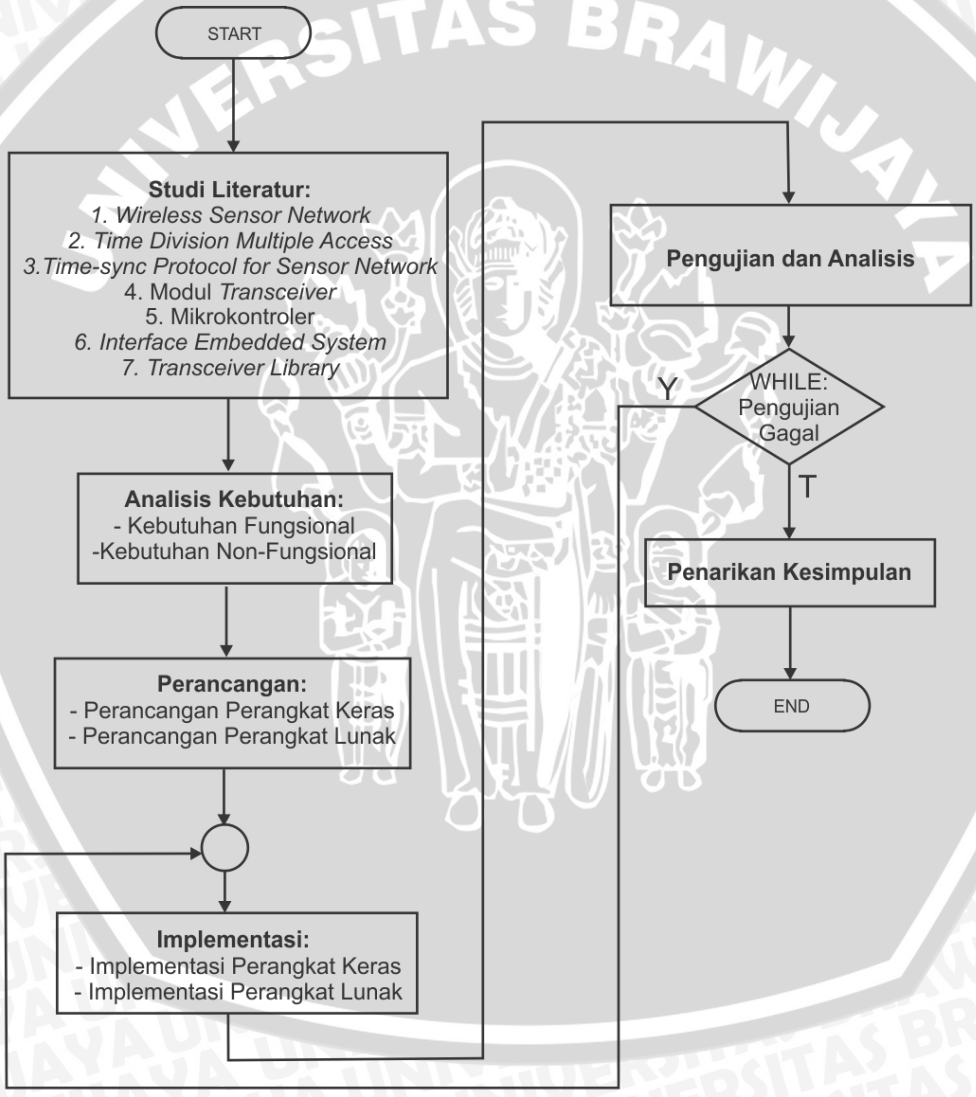
byte getStatus(void)	Mengembalikan nilai status register
void powerUpRx(void)	*Power up* chip dan set ke mode penerimaan
Void powerUpTx(void)	*Power up* chip dan set ke mode pengiriman

BAB 3 METODOLOGI

Bab 3 ini akan membahas tentang langkah kerja yang dilakukan dalam penulisan; diantaranya studi literatur, analisis kebutuhan sistem, implementasi, dan pengujian dari perancangan modul *Wireless Sensor Network.*

3.1 Metodologi Penelitian

Melalui tahap – tahap kegiatan yang tertuang dalam kerangka berpikir yang meliputi metode pengumpulan data dan metode pengolahan data. Kerangka berpikir pada penelitian ini dapat dilihat pada gambar 3.1 berupa diagram alir yang menggambarkan keseluruhan mekanisme sistem secara garis besar.



Gambar 3.1 Kerangka berfikir

3.1.1 Studi Literatur

Studi Literatur digunakan untuk menambah studi pustaka dan pengetahuan yang dilakukan untuk mengerjakan penulisan laporan dan penelitian. Studi literatur dilaksanakan dengan cara pengumpulan teori dan pustaka yang berkaitan dengan penelitian. Teori dan pustaka yang dipelajari meliputi:

1. *Wireless Sensor Network* (WSN)

   Studi literatur pada WSN dilakukan dengan pengkajian beberapa buku literatur seputar WSN guna memahami secara mendalam karakteristik dari WSN, batasan WSN, dan permasalahan dalam WSN dewasa ini. Pengkajian juga dilakukan pada *paper* implementasi WSN maupun survey mengenai WSN untuk mengamati perbandingan metode-metode yang dapat diterapkan pada WSN.
2. Metode *Time Division Multipe Access* (TDMA)

   Kajian seputar TDMA akan dilakukan menggunakan berbagai sumber literatur yang berkaitan dengan jaringan komputer pada umumnya, kemudian akan dikembangkan dengan mencari literatur panduan penerapan prinsip-prinsip TDMA dari sudut pandang pada *level embedded* WSN.
3. Algoritma *Time-sync Protocol for Sensor Network* (TPSN)

   Kajian pustaka pada TPSN akan difokuskan pada jurnal survey yang telah membandingkan beberapa algoritma sinkronisasi waktu, guna mengamati keunggulan dan kelemahan masing-masing algoritma terhadap Algoritma TPSN. kajian juga dilakukan untuk mengetahui teknik pengimplementasian Algoritma TPSN dengan benar.
4. Modul *Transceiver*

   Kajian mengenai modul *transceiver* akan difokuskan pada beberapa penelitian terkait yang telah dilakukan dan membandingkannya dengan pedoman manual modul *transceiver* yang digunakan.
5. Mikrokontroler

   Studi terkait dengan mikrokontroler berguna untuk mengetahui spesifikasi sistem yang digunakan, data dapat diperoleh dari *datasheet* mikrokontroler.
6. *Interface* pada *Embedded System*

   Interface pada *Embedded System* yang akan diterapkan pada hubungan antar perangkat keras berguna untuk keperluan *debugging* atau pencarian *error* yang dapat terjadi pada perangkat keras. Studi dilakukan pada literatur *Embedded System* dan pada hasil empiris yang dilakukan melalui pengujian.
7. *Transceiver Library* pada Mikrokontroler

   Pada bagian *library,* akan dilakukan studi mengenai teknik penggunaan seluruh komponen fungsi pada *library* untuk mengoptimalkan kinerja dari perangkat keras yang digunakan. Stufi juga dilakukan melalui percobaan empiris.

3.1.2 Analisis Kebutuhan

Dari studi pustaka yang telah dilakukan, maka penulis kemudian akan melakukan analisis mengenai kebutuhan fungsional dari sistem ini.

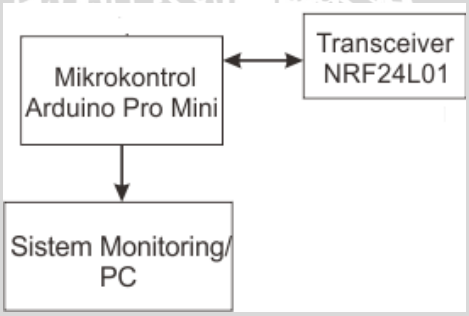
- Kebutuhan fungsional
  a. Sistem memiliki mekanisme waktu yang terstruktur secara keseluruhan
  b. Sistem dapat melakukan pengiriman data secara terjadwal sesuai mekanisme TDMA yang diterapkan
- Kebutuhan non-fungsional

  Untuk mengetahui spesifikasi kebutuhan non-fungsional sistem, dilakukan analisis yang melibatkan perangkat keras yang digunakan.
  a. 4 buah mikrokontroller Arduino Nano sebagai unit pemroses data
  b. 4 buah modul NRF24L01 sebagai perangkat *wireless*
  c. 1 buah komputer sebagai monitoring keberhasilan pengiriman data

3.1.3 Perancangan

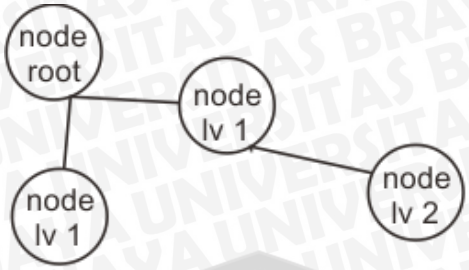
Perancangan sistem adalah tahap mulai merancang suatu sistem yang mampu memenuhi semua kebutuhan fungsional dalam penelitian ini. Perancangan ini terdiri dari perancangan perangkat keras, yaitu tentang pembuatan skematik dan pemasangan secara *embedded node* WSN, meliputi mikrokontroler Arduino dan modul NRF24L01. Kemudian pemasangan perangkat lunak Arduino beserta kebutuhan *driver.* Dan peracangan *Flowchart* pemrograman TPSN sistem.

Perancangan perangkat keras untuk tiap *node* seperti ditunjukkan pada gambar 3.2, terdapat mikrokontroller yang akan bertindak sebagai pemroses data, pada mikrokontroller ini pula akan diisikan program untuk algoritma TPSN dan TDMA. Mikrokontroler ini menggunakan sumber tegangan melalui USB yang tersambung dengan laptop, yang sekaligus menjadi media pengiriman data serial Arduino, kemudian tersambung ke *transceiver* NRF24L01 guna aliran data *wireless*. Disambungkan pula *output led* sebagai indikator.



Gambar 3.2 Diagram blok *node* WSN

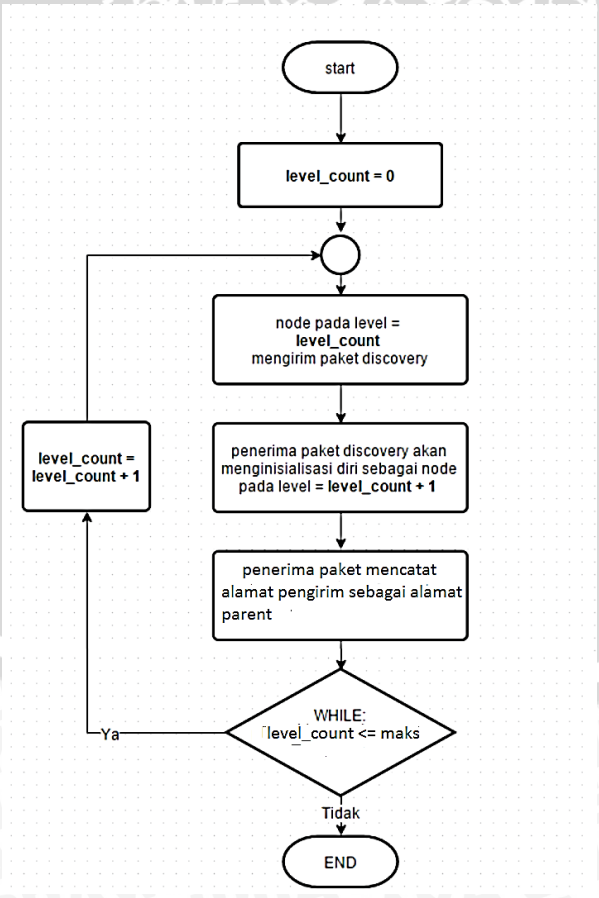
Pada gambar 3.3, ditunjukkan sebuah topologi *tree* yang menjadi dasar dari mekanisme algoritma TPSN, terdapat sebuah *node root* di pangkalnya, yang kemudian tersambung ke *node-node* lain membentuk sebuah topologi, yang di salah satu ujungnya akan disambungkan ke sistem monitoring.



Gambar 3.3 Topologi *tree* dalam TPSN

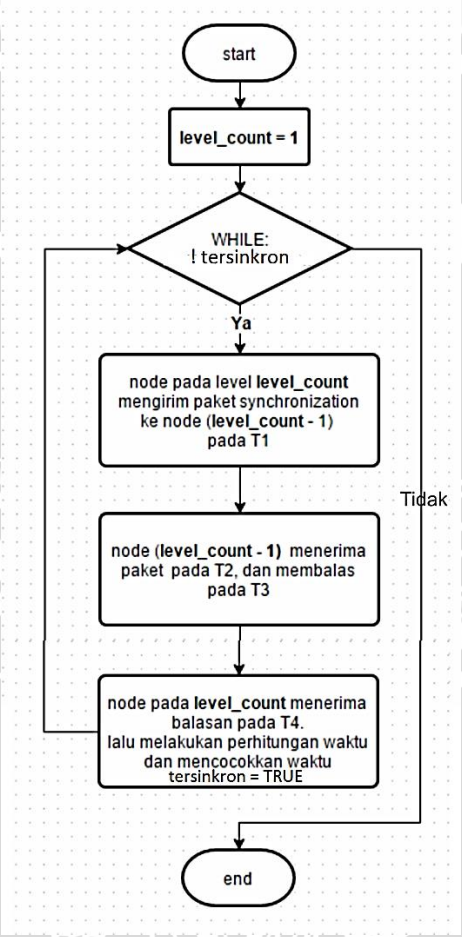
Perancangan *flowchart* untuk pemrograman algoritma TPSN digambarkan seperti pada gambar 3.5 dan 3.6. TPSN bekerja melalui dua fase. Fase pertama adalah fase *discovery*, yaitu fase untuk *node root* menentukan struktur topologi *tree* yang akan dibentuk, dan menentukan *node-node* yang mengisi tiap *level* topologi. Sedangkan fase kedua adalah fase *synchronization*, yaitu fase untuk mencocokkan waktu semua *node* dengan *node root*.

Fase pertama ditunjukkan oleh gambar 3.4, fase ini diawali ketika *root* (sebagai *level* 0) mengirim paket *discovery* kepada *node* terdekatnya. *Node* yang bisa menerima akan menjadi anggota *tree* pada level 1, kemudian *node level* 1 akan mengirim paket yang serupa untuk mencari anggota *level* berikutnya, sampai semua *node* tergabung dalam topologi.



Gambar 3.4 *Flowchart* TPSN fase *discovery*

Fase kedua ditunjukkan pada gambar 3.5, pada fase ini, akan dilakukan sinkronisasi waktu semua *node* untuk disamakan dengan waktu yang dimiliki *node Root*. Sama seperti fase pertama, sinkronisasi juga dimulai dari *level* paling kecil. Sebuah *node* akan mengirim paket *synchronization* dengan waktu T1, diterima oleh *node* sebelumnya. *Node* sebelumnya tersebut akan membalas dengan rincian T2, waktu penerimaan paket; dan T3, waktu pembalasan. *Node* pengirim awal akan menerima balasan pada T4, kemudian akan melakukan perhitungan waktu, dan mencocokkan waktu.



Gambar 3.5 *Flowchart* TPSN fase *Synchronization*

3.1.4 Implementasi

Implementasi dilakukan berdasarkan perancangan yang telah dibuat. Implementasi sistem ini memiliki beberapa tahapan, yaitu:

a. Implementasi perangkat keras dan konfigurasinya. Pada tahap ini akan diaplikasikan komponen-komponen *hardware* yang sesuai dengan rancangan. Konfigurasi pada masing-masing perangkat dilakukan untuk memastikan bahwa *hardware* yang ada telah terhubung dengan baik.
b. Implementasi perangkat lunak. Beberapa perangkat lunak pendukung akan diaplikasikan untuk menunjang jalannya sistem. Perangkat lunak tersebut seperti aplikasi Arduino. Arduino ini akan digunakan sebagai *compiler* untuk mengeksekusi kode program yang dibuat, dan melakukan proses *download*

ke board Arduino Nano. Selain itu, dibutuhkan *driver software* yang menghubungkan aplikasi Arduino dengan *port* keluaran PC.

Dari kombinasi implementasi di atas, diharapkan sistem dapat berjalan sesuai dengan apa yang diharapkan, yaitu:

1. Sistem memiliki mekanisme waktu yang terstruktur secara keseluruhan
2. Sistem dapat melakukan pengiriman data secara terjadwal sesuai mekanisme TDMA yang diterapkan

3.1.5 Pengujian dan Analisis

Pengujian dan analisis sistem dilakukan untuk mengetahui apakah kinerja dan performa keseluruhan sistem yang telah dirancang sesuai dengan spesifikasi kebutuhan yang melandasinya. Pengujian dilakukan dengan melakukan skenario sebagai berikut:

1. Pengecekan secara berkala pada setiap 30 detik, apakah semua *node* memiliki waktu yang sesuai dengan *node root*.
2. Pengecekan pengiriman dan penerimaan data setelah TDMA diterapkan.

Sedangkan analisis, dilakukan dengan menggunakan beberapa parameter sebagai berikut:

1. Penghitungan selisih waktu setiap *node* dengan *node root*.
2. Pencacahan data yang dikirimkan, dan penghitungan persentase kesesuaian antara data yang diterima dengan penjadwalan TDMA setiap *node.*

3.1.6 Kesimpulan

Kesimpulan terdiri dari ringkasan hasil pengujian dari sistem yang telah dilakukan, dan pembahasan mengenai pencapaian kebutuhan fungional maupun non-fungsional. Kesimpulan juga meliputi saran sebagai penutup dari penelitian untuk membahas mengenai ketentuan dan harapan tentang pengembangan penelitian berikutnya di masa mendatang.

BAB 4 REKAYASA PERSYARATAN

Dalam bab ini menjelaskan persyaratan minimal yang harus dipenuhi untuk perancangan hingga implementasi. Dengan harapan perancangan dan implementasi sistem TDMA dengan menggunakan algoritma TPSN bisa berjalan dengan baik.

4.1 Pendahuluan

Pada penelitian mengenai penerapan algortima TPSN dan skenario TDMA ini, penulis menentukan tujuan sistem, dan kegunaan dari dibuatnya sistem tersebut.

4.1.1 Tujuan

Perancangan sistem ini ditujukan untuk membantu proses pengiriman TDMA oleh banyak *node* WSN tanpa adanya interferensi yang bisa terjadi pada pengiriman tersebut, yang dapat menyebabkan tidak tersampaikannya data yang dikirim. Untuk itu, semua *node* harus mengikuti metode penjadwalan yang telah diatur, sehingga pengiriman akan dilakukan secara bergilir dari beberapa *node* tersebut yang sudah disamakan waktu lokalnya masing-masing. Secara kesuluruhan, akan ada tiga skenario dalam penelitian ini, yang terdiri dari dua fase sinkronisasi waktu dengan TPSN, ditambah sabuah skenario untuk pengujian TDMA.

4.1.2 Kegunaan

Sistem ini berguna untuk menyebarkan sebuah waktu lokal dari *node* yang bertindak sebagai *root*, kepada *node-node* lain yang akan mengikutinya sebagai acuan sesuai level hierarkinya. *Node-node* yang baru ditambahkan juga dapat bergabung dalam hierarki yang telah terbentuk setelah menerima paket *discovery* dari *node* yang akan menjadi *parent*-nya.

4.2 Kebutuhan Sistem

Kebutuhan sistem dibuat untuk mendeskripsikan kebutuhan fungsional dan kebutuhan non-fungsional sistem. Dengan demikian, dengan analisis ini, maka akan mempermudah dalam pembuatan sistem.

4.2.1 Kebutuhan Fungsional

Kebutuhan fungsional sistem antara lain:

a. Semua *node* memiliki waktu lokal yang sama
   Tahap ini diselesaikan setelah hierarki terbentuk, dan sebelum skenario TDMA bisa dilakukan, setiap *node parent* bertanggung jawab atas *node child*-nya
b. Model hierarki dapat dibuat otomatis
   Hierarki harus dibentuk ketika sebuah *node* aktif dengan menentukan level dan alamat dari *parent* yang akan dimintai sinkronisasi waktu, tanpa

harus melakukan setting terlebih dahulu secara manual pada tiap-tiap *node*

c. TDMA yang diterapkan dapat mencegah interferensi data
Jadwal pengiriman tiap *node* akan dikirimkan bersamaan dengan paket answer sinkronisasi yang berisi *time-stamp,* kemudian jadwal terebut kan digunakan untuk menjadwal skenario pengiriman TDMA.

4.2.2 Kebutuhan Non-fungsional

Kebutuhan non fungsional sistem antara lain:

a. Fase pembentukan tiap level hierarki diselesaikan tidak lebih dari 10 detik.
Semua *node* yang telah tersinkronisasi waktunya, berhak untuk menyebarkan paket *discovery* jika *clock*-nya telah tersinkronisasi dengan *parent*-nya.

b. Waktu yang dibutuhkan untuk keseluruhan sistem menyelesaikan sinkronisasi kurang dari 40 detik.
Sebelum tersinkronisasi, setiap *node* akan meminta paket sinkronisasi ke *parent*-nya dengan menunggu sebelumnya sesuai waktu tunggu acak yang dia hasilkan sendiri, demi menghindari interferensi dengan *node* lainnya.

c. Tingkat metode sinkronisasi mencapai satuan milidetik.
Mayoritas model WSN memiliki *delay* dalam satuan milidetik.

4.2.3 Kebutuhan Antarmuka Perangkat Keras

Antarmuka perangkat keras sistem ini adalah:

1. Arduino Nano
2. NRF24L01
3. Komputer/Laptop

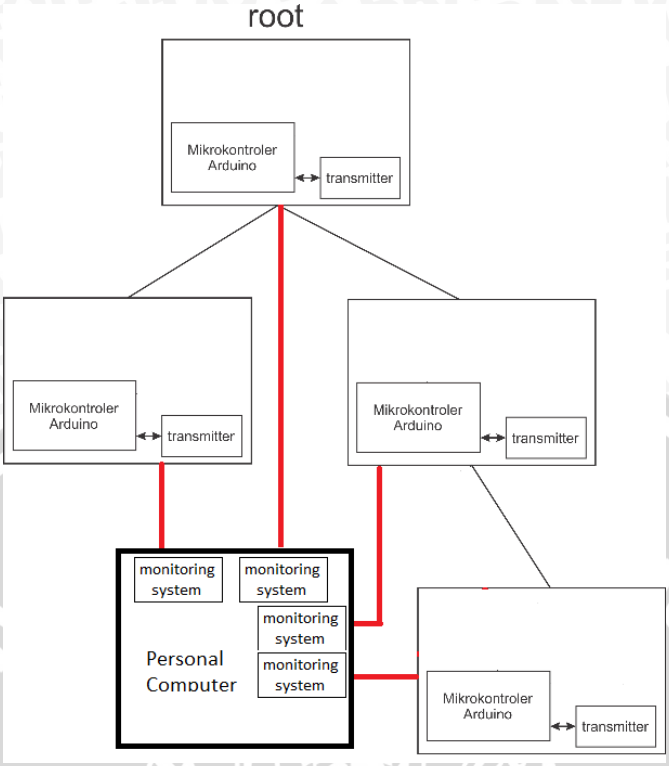
4.2.4 Kebutuhan Antarmuka Perangkat Lunak

Antarmuka perangkat lunak sistem ini adalah:

1. OS Windows 7
2. Arduino IDE
3. Library C++ (SPI, Mirf, NRF24L01, MirfHardwareSpiDriver, stdio, stdlib, dan String)
4. Driver USB

4.3 Diagram Blok

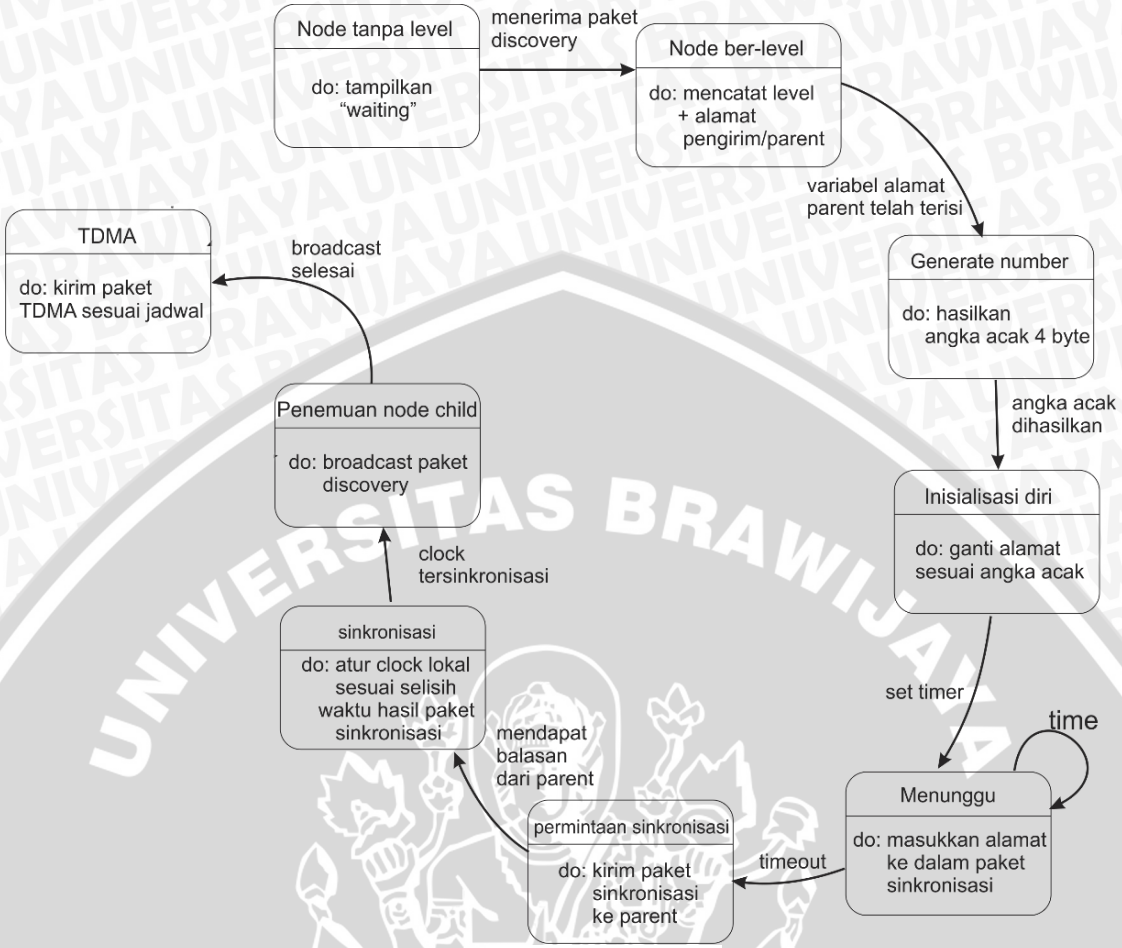
Diagram blok sistem terdiri dari model arsitektural yang menggambarkan posisi tiap-tiap hardware dalam skenario penelitian, dan model behavioral yang menggambarkan perilaku mekanisme sistem secara keseluruhan.



Gambar 4.1 Model Arsitektural Sistem

Berdasarkan gambar 4.1 di atas, dapat diterangkan sebagai berikut:

a. Tiap *node* terdiri dari satu set mikrokontroler dan transmitter
b. Mikrokontroler sebagai pengolah data
c. Garis hitam sebagai komunikasi nirkabel
d. Garis merah sebagai komunikasi kabel serial
e. Transmitter untuk modul pengiriman nirkabel
f. *Node root* adalah *node* yang tidak melakukan proses permintaan sinkronisasi kepada *node* lain, dan berada pada salah satu ujung model jaringan untuk bisa menjangkau tempat yang jauh
g. Monitoring system digunakan untuk melihat log yang dihasilkan sebuah *node*, menggunakan PC.



Gambar 4.2 Model behavioral

Dari gambar 4.2, dapat diterangkan secara lebih rinci sebagai berikut:

a. Tahapan *node* bergabung kedalam hierarki meliputi state "*node* tanpa *level*" sampai state "inisialisasi diri"
b. Fase sinkronisasi meliputi *state* "permintaan sinkronisasi" sampai "sinkronisasi"
c. Setelah memiliki *clock* yang tersinkronisasi, *node* akan melakukan *discovery* selagi melakukan *scenario* pengiriman TDMA.

BAB 5 PEMBAHASAN

Bab ini menjelaskan tentang perancangan sistem TDMA dengan Algoritma TPSN pada topologi *tree*. Dan implementasi sistem dari hasil perancangan berupa implementasi perangkat keras dan perangkat lunak.

5.1 Perancangan Sistem

Perancangan sistem ini meliputi perancangan perangkat keras dan perangkat lunak. Perancangan perangkat keras system akan membahas perangkat yang digunakan dan hubungannya. Perancangan perangkat lunak akan membahas program bahasa C++ yang dijalankan dalam IDE yang kemudian diupload pada perangkat keras Arduino.

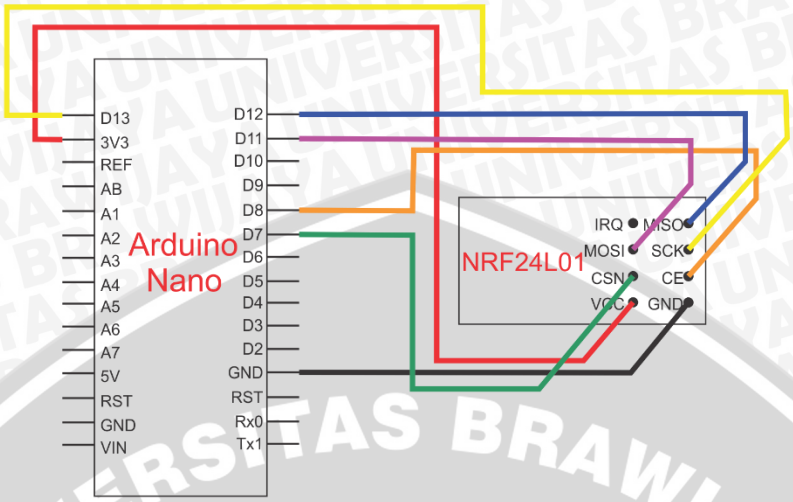
5.1.1 Perancangan Perangkat Keras

Perancangan perangkat keras ini akan menjelaskan tentang perangkat yang digunakan dalam system. Perangkat keras yang digunakan adalah Arduino Nano, NRF24L01, Laptop. Arduino Nano sebagai mikrokontroler akan tersambung dengan NRF24L01 dengan sambungan pin pada tabel 5.1. Dan gambaran sambungan antar pin Arduino dan NRF24L01 seperti ditunjukkan pada Gambar 5.1.

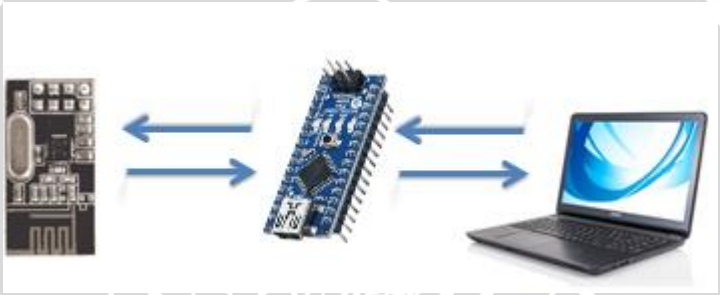
Tabel 5.1 Sambungan pin Arduino Nano dan NRF24L01

No	Pin Arduino Nano	Pin NRF24L01
1	7	CSN
2	8	CE
3	11	MOSI
4	12	MISO
5	13	SCK

Kemudian Arduino Nano akan disambungkan dengan Laptop menggunakan kabel USB untuk upload program, sumber daya tegangan, dan komunikasi data untuk monitoring. Secara keseluruhan hubungan perangkat keras ditunjukkan seperti gambar 5.2.



Gambar 5.1 Skematik Rangkaian Arduino Nano dan NRF24L01



Gambar 5.2 Hubungan perangkat keras

Komunikasi data nirkabel yang dilewatkan antar NRF24L01 akan diproses dan diteruskan oleh Arduino Nano untuk kemudian diteruskan ke PC sebagai sistem monitoring.

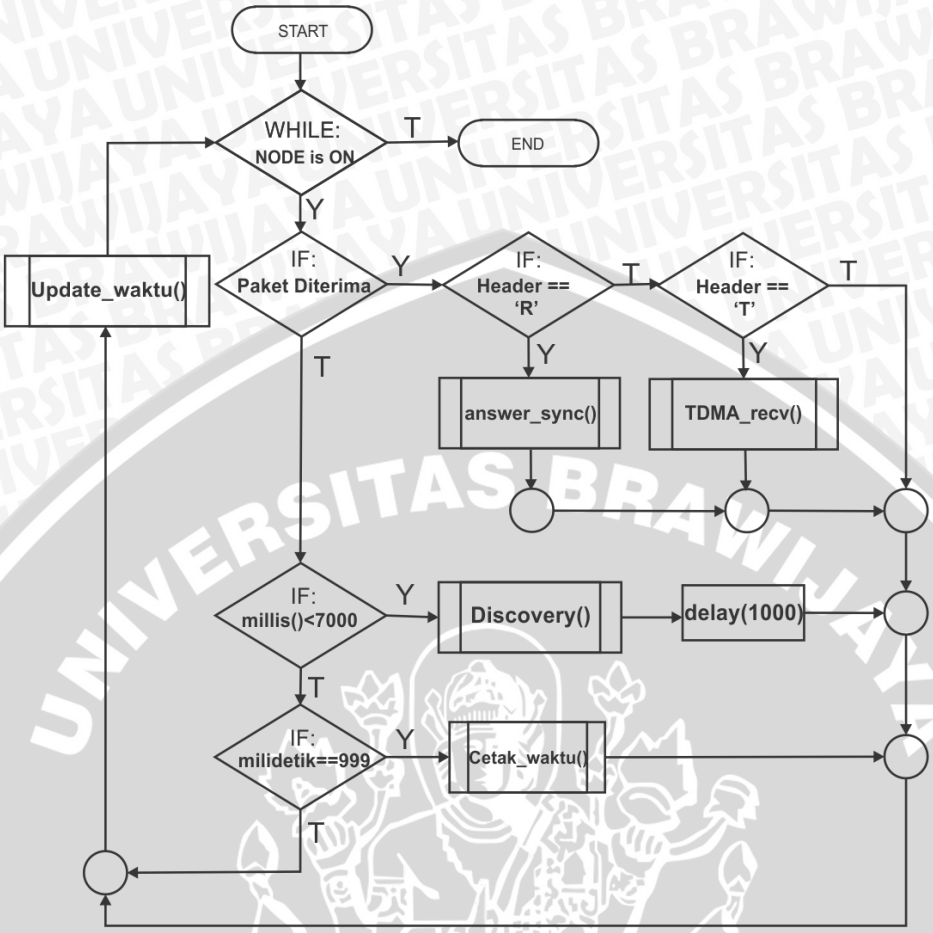
5.1.1 Perancangan Perangkat Lunak

Perancangan perangkat lunak pada system berisi kode program yang di-*compile* dan diupload menggunakan Arduino IDE dari laptop ke Arduino. Untuk menambahkan fitur NRF24L01 pada kode program, maka pada program akan ditambahkan library Mirf yang akan mengatur komunikasi data seperti pengiriman dan penerimaan data, pengalamatan, ukuran *payload,* dan kanal frekuensi yang digunakan.

Program ini terdiri dari dua jenis, program Root dan program *Node*. Program root akan membuat *node* yang menjalankannya melakukan tahapan *discovery* dan sinkronisasi secara langsung tanpa harus menunggu indikator tertentu. Sedangkan program *node* membuat *node* yang menjalankannya, harus menunggu hingga dikenali oleh root atau *parent*-nya melalui Fase *Discovery* dan telah tersinkronisasi, untuk kemudian bisa menjalankan tahapan *discovery* dan sinkronisasi lagi terhadap *child*-nya.

Program Root seperti ditunjukkan oleh diagram alir pada Gambar 5.3, rutinitas program Root dilakukan dengan pengecekan keberadaan paket yang diterima, kemudian jika *clock*-nya dalam fungsi millis() kurang dari 7000 (7 detik), maka akan melakukan *Discovery*, setelah itu fungsi *delay*() selama 1 detik berguna untuk memastikan paket *Discovery* dikirim dalam setiap 1 detik berturut-turut selamat 7 detik awal setelah *node Root* diaktifkan. Hal ini dilakukan supaya tidak terlalu "membanjiri" trafik keseluruhan *node* dengan Paket *Discovery* tersebut, selain itu, juga berguna untuk mempermudah dalam pengamatan proses ketika sebuah *node* bergabung dalam hierarki.

Setelah 7 detik, Root hanya melakukan pengecekan apakah ada paket yang diterima untuk diproses lebih lanjut, jika tidak ada, maka *Root* hanya akan mencetak waktu, *sampling* waktu diambil setiap 1 detik, maka pencetakan juga dilakukan setiap 1 detik dengan indicator milidetik = 999. Waktu perlu dilakukan *update* pada fungsi update_waktu() dengan tujuan untuk menentukan jam, menit, detik, dan milidetik dari *trigger* utama pada fungsi millis(). Jika ada paket yang diterima, maka akan diidentifikasi jenis paket tersebut melalui *header* yang merupakan array pertama pada paket tersebut. Paket yang mungkin diterima adalah *Request* Sinkronisasi dengan *header* 'R', dan paket TDMA dengan *header* 'T'. Paket *Request* Sinkronisasi akan ditangani oleh Fungsi answer_sync(), untuk dibalas dengan *timestamp*. Sedangkan Paket TDMA akan ditampilkan beserta aalamat pengirimnya.



Gambar 5.3 Diagram Alir Fungsi Utama Program *Root*

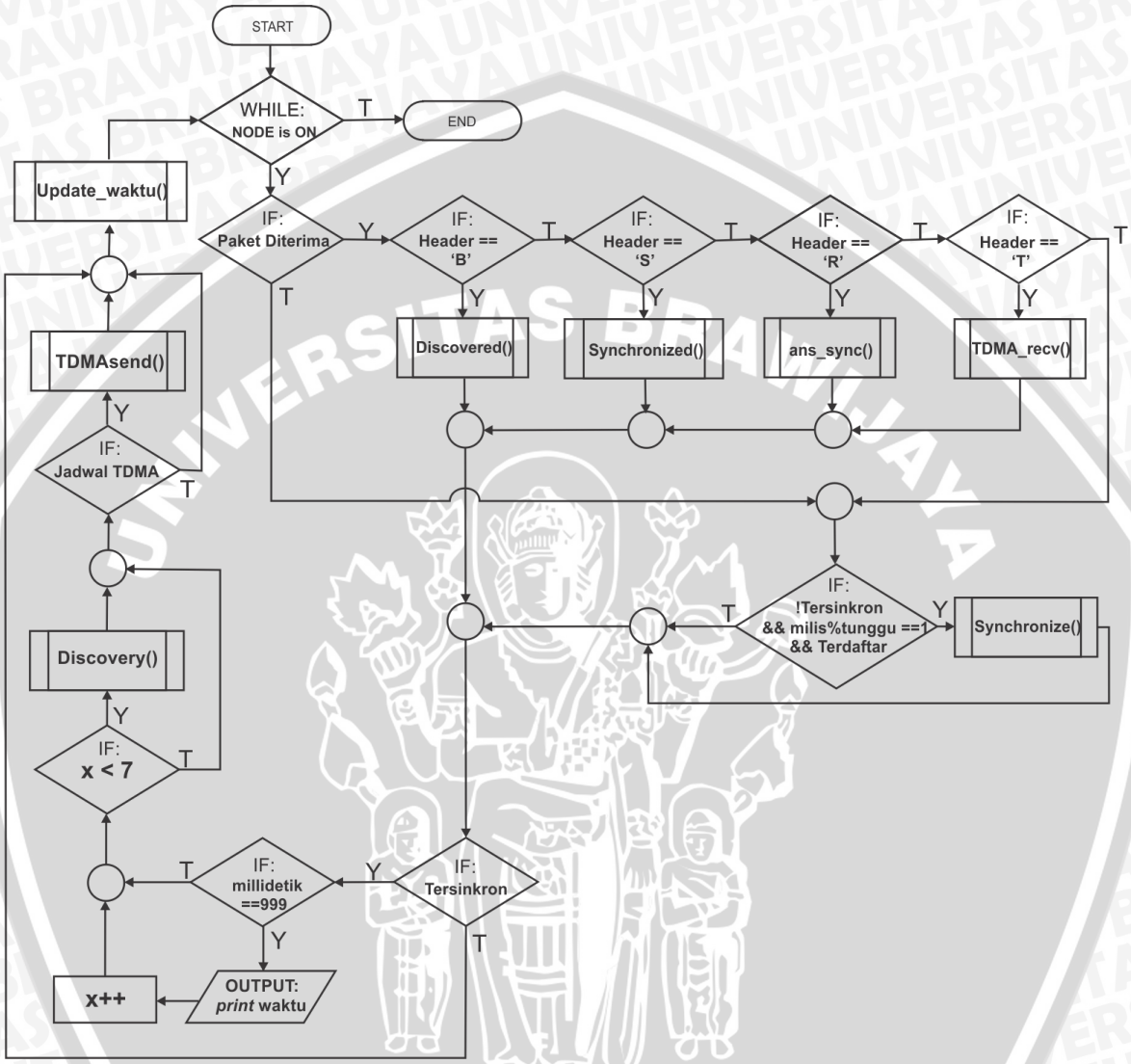
Program utama *Node* memiliki alur yang hampir sama dengan Program utama *Root,* namun memerlukan kompleksitas yang lebih banyak. Hal ini karena lebih banyak paket yang bisa diterima maupun dikirim. Selain itu, pada program utama *Node* juga terdapat perhitungan Algoritma TPSN untuk menghitung *delay* dan mencocokkan waktu.

Seperti ditunjukkan diagram alir pada Gambar 5.4, program utama *Node* juga melakukan pengecekan paket yang diterima sebagai rutinitas utama seperti pada Program *Root*. Paket yang mungkin diterima meliputi Paket *Discovery,* Paket Sinkronisasi, Paket *Request* Sinkronisasi, dan Paket TDMA.

Saat awal diaktifkan, Program tidak bisa melakukan apapun kecuali melakukan pengecekan apakah ada paket yang diterima, karena *node* tersebut sedang dalam kondisi tidak "terdaftar", dan tidak "tersinkronisasi". Setelah menerima Paket *Discovery,* maka *node* tersebut akan berada dalam kondisi terdaftar. Setelah terdaftar, maka *node* dapat mengeksekusi Fungsi synchronize() jika waktu saat itu sama dengan waktu tunggu.

Setelah melakukan sinkronisasi, maka *node* memasuki kondisi tersinkronisasi, maka akan dapat melakukan pencetakan waktu supaya penguji dapat mengamati waktu *node* tersebut, selain itu *node* juga sudah dapat

mengeksekusi Fungsi *discovery*() untuk melakukan *broadcast* Paket *Discovery* mencari *child* yang tidak terjangkau oleh *parent*-nya. Pada kondisi ini, *node* juga dapat mengeksekusi Fungsi TDMAsend() jika waktunya sama dengan *timeslot* yang telah didapatkan.

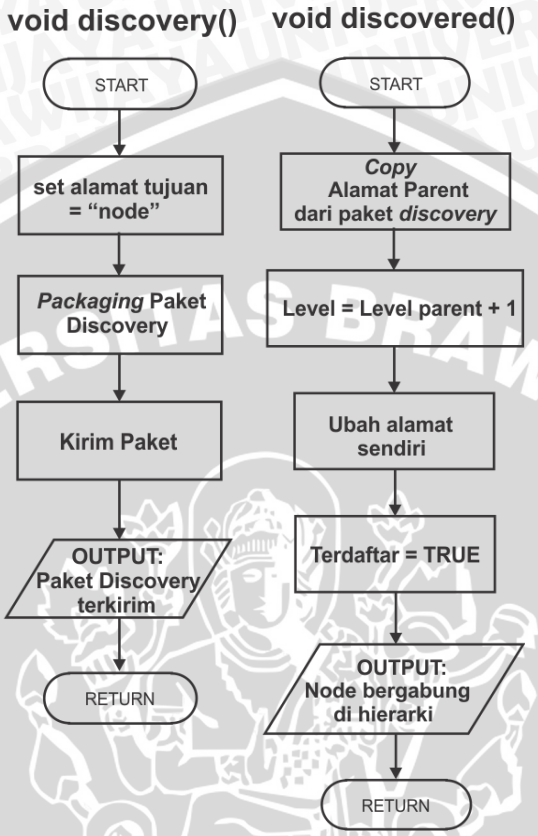


Gambar 5.4 Diagram Alir Fungsi Utama Program *Node*

Gambar 5.5 menunjukkan dua buah fungsi yang bertanggung jawab mengatasi proses *Discovery*, Gambar 5.5a, adalah Fungsi *discovery*() untuk melakukan *broadcast* paket tersebut pada sisi *node* pengirim. Tahapan pertama adalah melakukan konfigurasi alamat tujuan *default*, kemudian memasukkan nilai variabel pada paket yang hendak dikirim, setelah itu paket dikirim, kemudian melakukan pencetakan pada Serial Monitor yang menandakan bahwa ada Paket *Discovery* yang baru saja dikirim.

Diagram alir pada Gambar 5.5b, adalah fungsi untuk melakukan *handle* atas Paket *Discovery* yang telah beredar dan masuk pada sisi penerima. Langkah-langkah yang dilakukan adalah menyalin alamat calon *parent* pada paket

tersebut, menentukan *level* berdasarkan *level parent* yang ada pada paket tersebut, mengubah alamat supaya tidak lagi menerima Paket *Discovery,* mengganti status Terdaftar menjadi "TRUE", dan melakukan pencetakan pada Serial Monitor.



(a) **(b)**

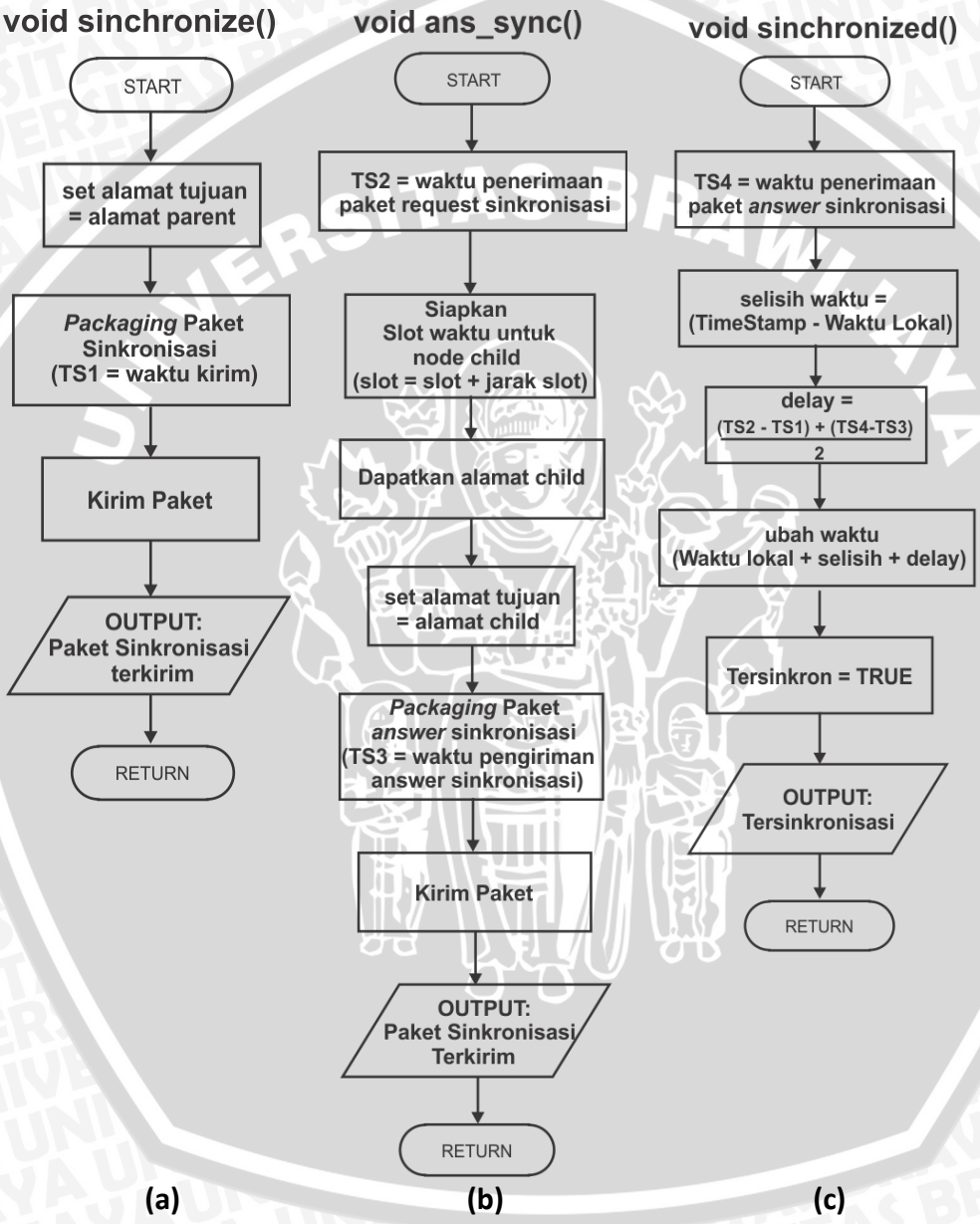
Gambar 5.5 Diagram Alir Fungsi Fase *Discovery*

Pada Gambar 5.6, ditunjukkan tiga buah fungsi yang terlibat pada Fase Sinkronisasi, yaitu synchronize() pada sisi peminta digunakan untuk melakukan *request* sinkronisasi, ans_sync() pada sisi penjawab digunakan untuk menjawab jika ada *request* sinkronisasi dengan *timestamp* dan *timeslot* yang akan digunakan pada skenario TDMA, kemudian synchronized() pada sisi peminta digunakan untuk menangani Paket Sinkronisasi balasan dari sisi penjawab.

Ketiga fungsi terdapat pada Program *Node,* dan hanya Fungsi ans_sync() yang ada pada Program *Root. Node* biasa akan melakukan permintaan sinkronisasi pada *parent*-nya, dan juga melayani permintaan sinkronisasi dari *child*-nya. Sedangkan *Node Root* hanya melayani permintaan sinkronisasi karena tidak memiliki *parent.*

Fungsi synchronize() pada Gambar 5.6a berjalan seperti demikian, pertama mengatur tujuan pengiriman pada alamat *parent* yang tadi didapatkan setelah menerima Paket *Discovery,* dilanjutkan dengan persiapan "pembungkusan" Paket *Request* sinkronisasi yang juga meliputi waktu pengiriman, setelah siap kemudian paket dikirim, dan diakhiri dengan mencetak pada Serial Monitor.

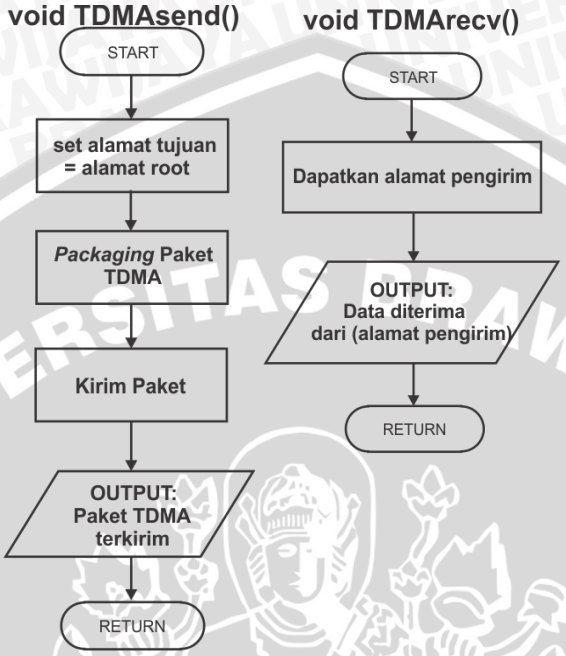
Fungsi ans_sync() pada Gambar 5.6b dieksekusi ketika paket *request* telah tiba, maka akan dicatat waktu penerimaan dan dimasukkan pada variabel TS2, kemudian menyiapkan slot waktu yang belum terpakai, mencatat alamat *child* pengirim tersebut untuk di-set sebagai alamat tujuan dari paket balasan yang akan dikirimkan, pembungkusan paket menyertakan TS1 sebagai waktu pengiriman *request,* TS2 sebagai waktu penerimaan *request,* dan TS3 sebagai waktu pengiriman paket balasan.



Gambar 5.6 Diagram Alir Fungsi Fase Sinkronisasi

Pada Gambar 5.6c, setelah paket balasan dikirimkan, maka ketika sisi peminta menerima paket balasan, akan dilanjutkan dengan megeksekusi Fungsi synchronized(), mencatat waktu penerimaan paket sebagai TS4. Kemudian menghitung selisih waktu lokal dengan waktu yang ada pada paket balasan,

menghitung *delay* dari keempat variabel TS1, TS2, TS3, dan TS4. Setelah itu mengubah waktu dengan menambahkan selisih dan *delay* hasi perhitungan yang teah dilakukan. Setelah selesai mengubah waktu, *node* akan mengubah status menjadi Tersinnkronisasi. Eksekusi fungsi diakhiri dengan mencetak pada Serial Monitor.



(a) **(b)**

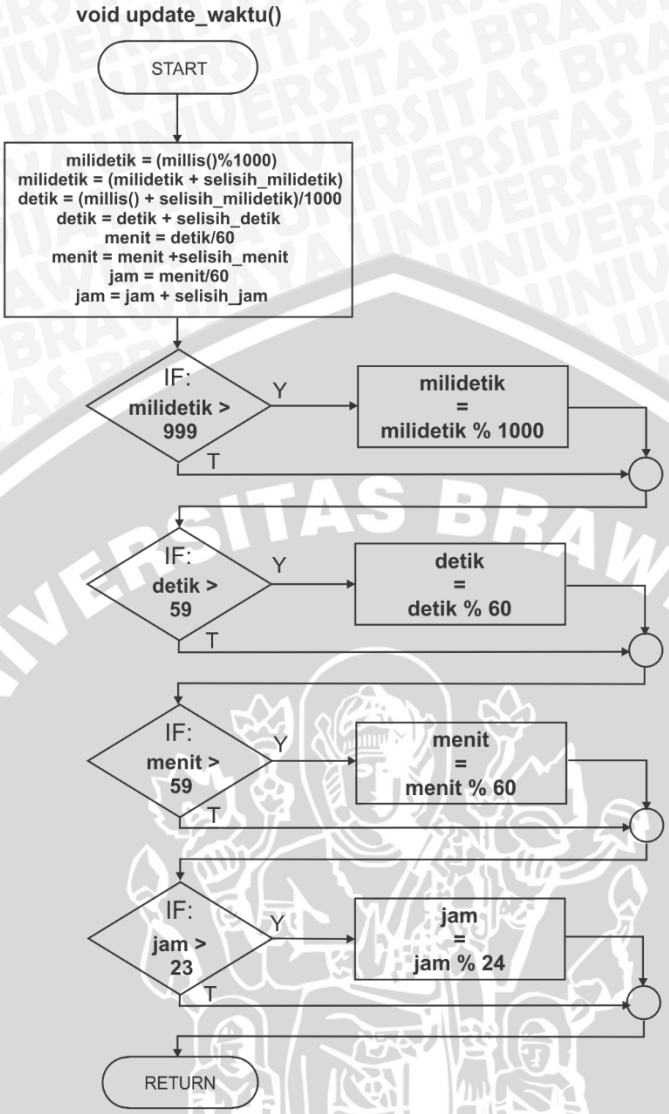
Gambar 5.7 Diagram Alir Fungsi Fase TDMA

Gambar 5.7 menunjukkan diagram alir fungsi yang menangani skenario pengiriman TDMA, kedua fungsi bisa dieksekusi ketika waktu diatas 1 menit. Fungsi TDMAsend() pada Gambar 5.7a, dan TDMArecv() pada 5.7b, adalah dua fungsi pengirim dan penerima paket TDMA.

Langkah-langkah yang dijalankan Fungsi TDMAsend() adalah melakukan konfigurasi alamat tujuan pengiriman, membungkus paket, mengirim paket, setelah itu mencetak pada Serial Monitor.

Pada sisi penerima, akan dicatat alamat pengirim untuk kemudian dicetak pada Serial Monitor sebagai pengirima TDMA. Peneliti dapat memantau keberhasilan sinkronisasi waktu dari skenario TDMA yang dilakukan tersebut.

Pada gambar 5.8, ditunjukkan sebuah fungsi update_waktu() yang berguna untuk melakukan pemisahan fungsi milis() menjadi jam, menit, detik, dan milidetik. Selain itu juga terdapat baris program yang mengatur pencocokan waktu setelah terjadi *update.* Yaitu variabel selisih_jam, selisih_menit, selisih_detik, dan selisih_milidetik yang selalu ditambahkan setiap fungsi dipanggil untuk mencocokkan waktu.



Gambar 5.8 Diagram Alir Fungsi *Update* Waktu

5.2 Implementasi Sistem

Pada tahapan implementasi sistem menjelaskan tentang spesifikasi perangkat keras dan spesifikasi perangkat lunak, batasan-batasan implementasi, serta implementasi perangkat keras dan algoritma TPSN pada TDMA.

5.2.1 Spesifikasi Perangkat Keras

Implementasi sistem pada spesifikasi perangkat keras yang digunakan antara lain laptop, Arduino Nuno, dan NRF24L01.

1. Laptop

   Laptop disini berfungsi sebagai media IDE Arduino pembuatan kode program, dan sebagai monitoring. Berikut spesifikasi laptop yang digunakan pada table 5.2.

Tabel 5.2 Spesifikasi Laptop

Lenovo G480	
Processor	Intel(R) Core(TM) i3-3120M CPU @ 2.50GHz (4CPUs)
Memory	2048MB RAM
Display	Intel(R) HD Graphic 4000

2. Arduino Nano
   Arduino Nano sebagai mikrokontroler untuk memproses data NRF24L01. Berikut spesifikasinya pada table 5.3

Tabel 5.3 Spesifikasi Arduino Nano

Arduino Nano	
Mikrokontroler	ATmega 328
Tegangan operasional	5 Volt
Pin Digital I/O	14 Pin (termasuk 6 support PWM)
Arus DC	40 mA
Flash Memory	32 KB
SRAM	2 KB
EEPROM	1 KB
Kecepatan clock	16 MHz
Dimensi	0,73 x 1.70 inchi

3. NRF24L01
   NRF24L01 sebagai modul nirkabel untuk berkomunikasi antar *node*. Berikut spesifikasinya pada table 5.4

Tabel 5.4 Spesifikasi NRF24L01

NRF24L01	
Kecepatan operasi	2 Mbps(maks), modulasi GFSK
Wireless rate	1 atau 2 Mbps
SPI rate	0 – 8 Mbps
Support Kristal osilator	60 ppm, 16 Mhz
Tegangan operasional	1,9 – 3,6 Volt
Tegangan I/O	5 Volt
Saluran frekuensi	125 total frekuensi operasional
Radius transmisi	100 m (maks)
Support Kristal osilator	60 ppm, 16 Mhz
Tegangan operasional	1,9 – 3,6 Volt
Tegangan I/O	5 Volt
Saluran frekuensi	125 total frekuensi operasional
Radius transmisi	100 m (maks)

5.2.2 Spesifikasi Perangkat Lunak

Spesifikasi perangkat lunak pada system berisi perangkat lunak yang akan digunakan pada pembuatan system.

Tabel 5.5 Spesifikasi perangkat lunak

Lenovo G480	
Operating system	Windows 7 Ultimate 64-bit
Programming tool	Arduino IDE

5.2.3 Batasan implementasi

Dalam melakukan perancangan system diperlukan adanya batasan implementasi agar system terimplementasi dengan semestinya. Batasan implementasi pada system ini antara lain:

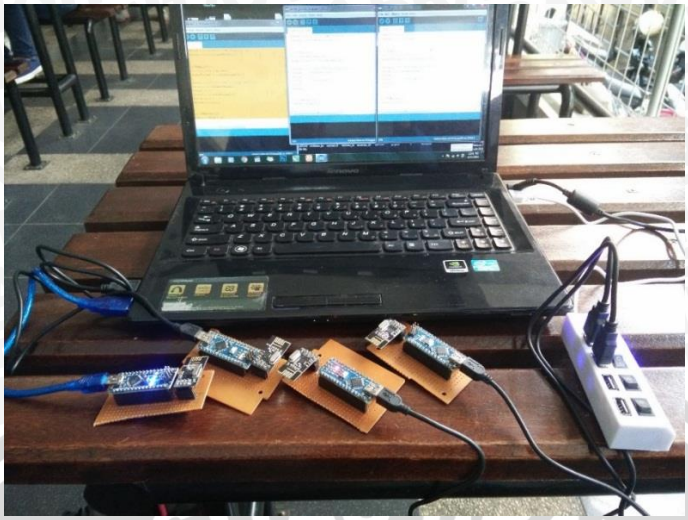
1. *Node* WSN yang digunakan sebanyak 4 buah, dengan posisi *Node Root* sebanyak 1 buah.
2. Sistem *Monitoring* calon *Node* level 2 diaktifkan ketika *Node Root* berhenti melakukan broadcast paket *Discovery,* supaya calon *Node* level 2 tidak masuk ke level 1.
3. *Monitoring* pengujian fungsional sistem dilakukan melalui *Serial Monitor* yang ada pada Arduino IDE.

5.2.4 Implementasi perangkat

Implementasi perangkat ini berisi tahapan pengoperasian perangkat-perangkat yang diperlukan ketika melakukan implementasi. Pada implementasi ini terdapat dua pembahasan yaitu implementasi perangkat keras dan implementasi perangkat lunak WSN.

5.2.4.1 Implementasi perangkat keras WSN

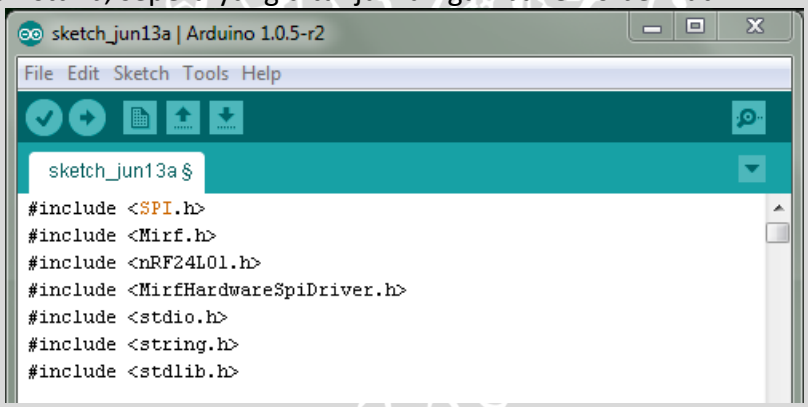
Implementasi ini berisi tahapan pengoperasian perangkat yang akan digunakan. Seperti pada sub bab sebelumnya yaitu sub bab perancangan. Implementasi perangkat keras diawali dengan melakukan sambungan Arduino Nano sebagai mikrokontroler. Arduino Nano akan tersambung dengan modul *wireless* NRF24L01 melalui sambungan pin. Kemudian Arduino Nano akan disambungkan dengan Laptop menggunakan kabel USB untuk upload program. Berikut pada gambar 5.9 adalah implementasi perangkat keras WSN.



Gambar 5.9 Sambungan keseluruhan sistem

5.2.4.2 Implementasi perangkat lunak ke WSN

Implementasi perangkat lunak ke WSN dilakukan melalui Arduino IDE. Kode program di-*compile* dan diupload menggunakan dari laptop ke Arduino. Untuk dapat dijalankan program membutuhkan beberapa *library* untuk mempermudah dalam konfigurasi dan pengunaan perangkat keras. Dalam melakukan sambungan NRF24L01 dibutuhkan beberapa *library* yang dimasukkan kedalam *source code*-nya, diantaranya, SPI, Mirf, nRF24L01, stdio, string, dan stdlib, seperti yang ditunjukkan gambar 5.10 berikut.



Gambar 5.10 *Library* yang digunakan

BAB 6 PENGUJIAN

Bab ini membahas tentang proses pengujian dan analisis terhadap sistem setelah proses perancangan dan implementasi telah dilakukan. Pengujian ini mengunakan 4 *node* WSN. Pengujian dilakukan untuk mengetahui fungsionalitas sistem, waktu operasional yang dibutuhkan untuk setiap fase, serta fleksibilitas dari sistem yang telah dibuat. Kemudian akan dilakukan analisis untuk mendapatkan kesimpulan dari hasil pengujian yang telah diperoleh.

6.1 Pengujian Sambungan *Node* WSN ke Laptop (PC)

Pengujian *node* WSN ke laptop digunakan untuk mengetahui *node* tersambung dengan benar dan dapat berfungsi sebagaimana mestinya. Komponen yang diperlukan dalam melakukan pengujian ini adalah Arduino Nano yang telah terpasang pada PCB dan tersambung dengan NRF24L01, kemudian disambungkan ke laptop menggunakan kabel USB serial.

6.1.1 Prosedur pengujian sambungan *Node* WSN ke Laptop (PC)

Untuk pengujian sambungan *Node* WSN ke Laptop, dilakukan prosedur yang dipaparkan pada tabel 6.1.

Tabel 6.1 Prosedur Pengujian *Node* WSN ke Laptop

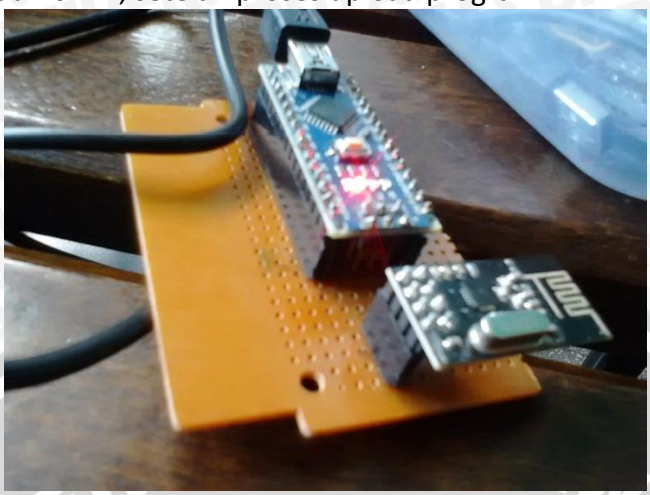
Kasus Pengujian	Pengujian sambungan *node* WSN ke laptop.
Objek Pengujian	Komunikasi data serial *node* WSN dari dan ke laptop; Serial Monitor Arduino; LED *indicator* Arduino Nano.
Tujuan Pengujian	Memastikan laptop dan *node* WSN dapat berfungsi untuk melakukan upload program dan pengamatan pada serial monitor Arduino.
Prosedur Pengujian	1. Penguji memasangkan *node* WSN pada laptop menggunakan USB serial. 2. Penguji mengaktifkan Arduino IDE pada laptop. 3. Penguji mengamati indikator LED Arduino. 4. Penguji mengamati COM PORT pada Serial Monitor. 5. Penguji melakukan *upload* kode program.

6.1.2 Hasil Pengujian Sambungan *Node* WSN ke Laptop (PC)

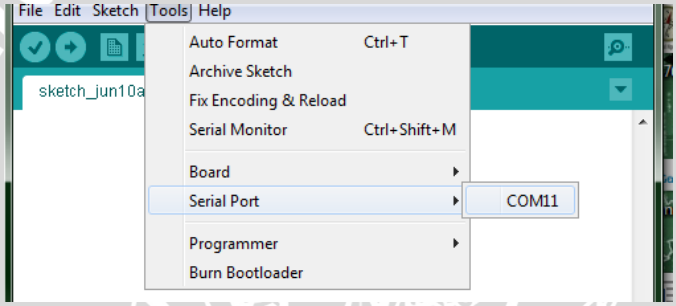
Setelah semua komponen terpasang dengan benar, pengujian akan dilakukan dengan mengamati beberapa parameter berikut.

1. LED power Arduino yang menyala seperti ditunjukkan pada gambar 6.1
2. Serial port Arduino yang aktif, dapat diakses dengan menu tool > Serial Port pada Arduino IDE, ditunjukkan pada gambar 6.2

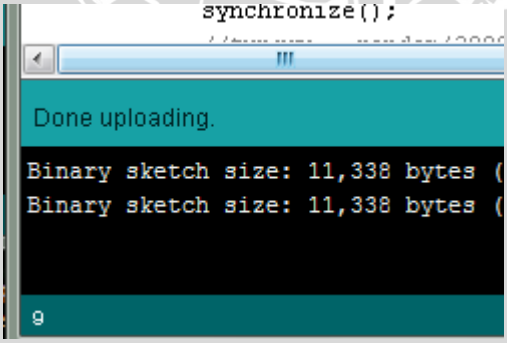
3. Proses Upload berhasil, ditunjukkan dengan indicator "*Done uploading*" pada Arduino IDE, setelah proses upload program



Gambar 6.1 Indikator LED Arduino



Gambar 6.2 Serial port Arduino



Gambar 6.3 Indikator proses upload sukses Arduino IDE Pengujian Fungsional

Dari hasil pengamatan, tampak bahwa Arduino Nano berjalan baik, dan Arduino IDE menunjukkan tidak ada permasalahan.

6.1.3 Analisis pengujian sambungan *node* WSN ke laptop (PC)

Dari pengujian yang telah dilakukan, didapatkan hasil analisis bahwa seluruh komponen dapat berjalan dengan baik. *Node* WSN dapat berkomunikasi dengan laptop baik untuk keperluan *upload* program, maupun untuk monitoring.

6.2 Pengujian Fungsional

Pengujian fungsional digunakan untuk mengetahui bahwa beberapa tahapan sistem dapat berjalan sesuai dengan tujuan yang diharapkan, serta untuk menguji tingkat kehandalan sistem. Beberapa tahapan dari keseluruhan sistem yang diamati dalam pengujian ini yaitu keberhasilan Fase *Discovery*, tingkat keberhasilan waktu tunggu acak pada Fase Sinkronisasi, dan keberhasilan pengiriman TDMA.

6.2.1 Fungsionalitas Fase *Discovery*

Pengujian fase *discovery* bertujuan untuk mendapatkan hasil pengamatan atas keberhasilan setiap *node* yang baru diaktifkan untuk memasuki hierarki sistem yang telah dibuat dan menjadi *child* dari suatu *node* yang memberikan paket *discovery* padanya. Setelah memasuki suatu *level* pada hierarki, *node* tersebut akan mengganti alamatnya untuk keperluan pengiriman data secara individual, dan tidak menggunakan sistem *broadcast* seperti pada pengiriman Paket *Discovery.* Sehingga pengujian juga dilakukan untuk mengetahui bahwa bilangan acak yang dihasilkan untuk alamat yang baru dapat memperkecil kemungkinan adanya kesamaan alamat suatu *node* dengan *node* lainnya. Prosedur pengujian untuk fungsionalitas Fase *Discovery* seperti ditunjukkan pada Tabel 6.2.

Tabel 6.2 Prosedur Pengujian Fungsionalitas Fase *Discovery*

Kasus Pengujian	Proses pembentukan hierarki dengan Fase *Discovery.*
Objek Pengujian	Kesesuaian penempatan sebuah *node* ditinjau dari alamat *parent* dan *level* hierarki; alamat acak baru yang berbeda dengan *node* lainnya dalam hierarki.
Tujuan Pengujian	Membuktikan Paket *Discovery* yang dikirimkan dapat membentuk hierarki secara bertahap pada tiap-tiap *level.*
Prosedur Pengujian	1. Penguji memasukkan kode program pada setiap *node* dalam pengujian (1 buah *Node Root* dan 2 buah *node* biasa). 2. Penguji mengaktifkan *Node Root* sebagai *node* pertama (Level 0). 3. Saat *node* melakukan pengiriman Broadcast *Discovery*, penguji mengaktifkan *node* calon level berikutnya. 4. Penguji mencatat alamat baru, alamat *parent* dan *level* dari *node* baru yang bergabung. 5. Kembali ke prosedur nomor 3 sampai semua *node* sudah bergabung dalam hierarki. 6. Pengujian dilakukan sebanyak 5 kali percobaan.

6.2.1.1 Hasil Pengujian Fungsionalitas *Fase Discovery*

Dari pengujian yang telah dilakukan, diperoleh hasil seperti pada gambar 6.4. Keseluruhan pengujian dirangkum pada Tabel 6.3.

Gambar 6.4 Tampilan Fase *Discovery*

Tabel 6.3 Hasil Pengujian Fungsionalitas Fase *Discovery*

Percobaan ke-	ID *Node*	Level *Node*	Alamat Lama	Alamat Baru	Alamat *Parent*
1	A	1	*node*	8525	1111 (*root*)
	B	2	*node*	8306	8525
2	A	1	*node*	4625	1111 (*root*)
	B	2	*node*	2427	4625
3	A	1	*node*	9739	1111 (*root*)
	B	2	*node*	6995	9739
4	A	1	*node*	1742	1111 (*root*)
	B	2	*node*	7397	1742
5	A	1	*node*	8723	1111 (*root*)
	B	2	*node*	6293	8723

Hasil percobaan seperti yang ditunjukkan dalam tabel 6.3, menunjukkan bahwa semua *node* berhasil memasuki hierarki dengan alamat acak yang dihasilkan berbeda dengan *node* lain. Alamat *parent* yang dimiliki sesuai dengan *node* pemberi *broadcast Discovery*, dan *level* sudah menunjukkan berada tepat di bawah *level parent.*

6.2.1.2 Analisis Pengujian Fungsionalitas Fase *Discovery*

Dari pengujian yang telah dilakukan, dapat diberikan analisis bahwa pembentukan hierarki sistem menggunakan *broadcast* Paket *Discovey* sudah dapat berjalan dengan baik. Sistem dipastikan dapat melakukan pembentukan hierarki tanpa skenario pengujian (semua *node* dapat aktif bersama-sama) jika

menggunakan kondisi dan jarak yang sesuai, yaitu harus ada minimal satu *node* yang dapat menjangkau *Node Root,* dan setiap *node* minimal harus dapat menjangkau satu *node* selain *Node Root.*

6.2.2 Fungsionalitas Fase Sinkronisasi

Pengujian fungsionalitas pada Fase Sinkronisasi bertujuan untuk mengamati keberhasilan setiap *node* melakukan proses sinkronisasi untuk menyamakan waktu lokal dengan waktu *Node Root*, pada proses permintaan sinkronisasi sampai serta proses *answer* sinkronisasi. Proses permintaan sinkronisasi ditandai dengan pengiriman paket *request* sinkronisasi kepada *Node Root*, sedangkan proses *answer* sinkronisasi ditandai dengan pengiriman paket *answer* sinkronisasi dari Root kepada *node* pengirim *request*.

Sebelum dimulai proses Sinkronisasi, setiap *node* harus menunggu selama waktu acak kurang dari 25 detik. Pengujian akan mengamati bahwa waktu acak yang dihasilkan tidak sama antara satu *node* dengan *node* lain, sehingga akan memperkecil kemungkinan adanya pengiriman *request* sinkronisasi secara bersamaan yang dapat menyebabkan peristiwa interferensi. Prosedur pengujian untuk Fase Sinkronisasi dapat dilihat pada tabel 6.4.

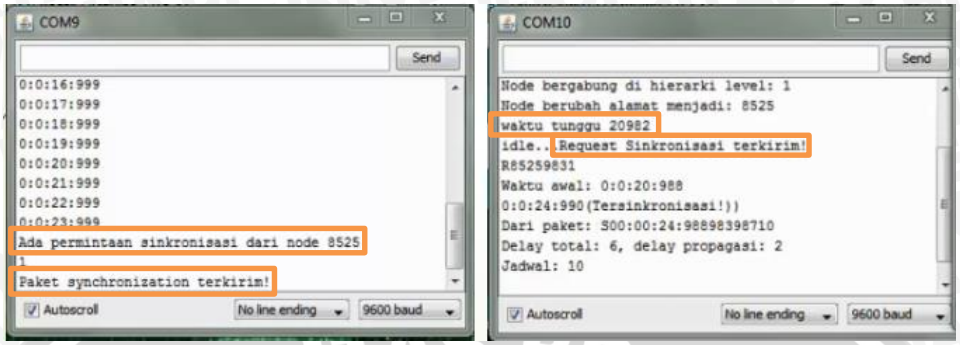
Tabel 6.4 Prosedur Pengujian Fungsionalitas Fase Sinkronisasi

Kasus Pengujian	Proses sinkronisasi waktu pada Fase Sinkronisasi.
Objek Pengujian	Keberhasilan *request* dan *answer* Paket Sinkronisasi antara sebuah *node* dengan *parent*-nya ditinjau dari pengaruh waktu tunggu acak.
Tujuan Pengujian	Mengamati kelancaran Paket *request-answer* Sinkronisasi yang dikirimkan.
Prosedur Pengujian	1. Penguji memasukkan kode program pada setiap *node* dalam pengujian (1 buah *Node Root* dan 2 buah *node* biasa). 2. Penguji menunggu semua *node* membentuk hierarki dengan Fase *Discovery.* 3. Penguji mencatat waktu tunda yang dihasilkan setiap *node*. 4. Penguji mengamati keberhasilan pengiriman *request* dan *answer* Sinkronisasi dengan dua *node* yang memiliki *parent* yang sama pada 1 menit pertama percobaan. 5. Pengujian dilakukan sebanyak 10 kali.

6.2.2.1 Hasil Pengujian Fungsionalitas Fase Sinkronisasi

Sampel proses pengujian Fase Sinkronisasi tampak pada gambar 6.5, dimana COM10 terlihat melakukan *request* ke *parent*-nya (COM9) setelah

memasuki kondisi *idle* selama waktu tunggu yang dihasilkannya sendiri. Kemudian COM9 sebagai *parent* terlihat membalas paket *request* yang diterima. Keseluruhan hasil pengujian ditunjukkan pada Tabel 6.5.



Gambar 6.5 Tampilan Fase Sinkronisasi

Tabel 6.5 Hasil pengujian Fungsionalitas Fase Sinkronisasi

Percobaan ke-	ID *Node*	Level *Node*	Waktu Tunda (detik)	Keberhasilan *request*	Keberhasilan *answer*
1	A	1	20,982	✓	✓
	B	1	15,773	✓	✓
2	A	1	19,259	✓	✓
	B	1	15,374	✓	✓
3	A	1	23,289	✓	✓
	B	1	20,089	✓	✓
4	A	1	15,537	✓	✓
	B	1	19,812	✓	✓
5	A	1	23,673	✓	✓
	B	1	22,790	✓	✓
6	A	1	16,691	✓	✓
	B	1	16,718	×	×
7	A	1	15,016	✓	✓
	B	1	14,269	✓	✓
8	A	1	22,539	✓	✓
	B	1	23,016	✓	✓

Tabel 6.5 (Lanjutan)

9	A	1	22,218	✓	✓
	B	1	20,749	✓	✓
10	A	1	20,544	✓	✓
	B	1	21,660	✓	✓

Hasil pada tabel 6.5, menunjukkan bahwa pengiriman *request* dan *answer* sinkronisasi berjalan lancar kecuali pada percobaan ke-6, dimana tidak ada aliran paket antara *node* B dan *parent*-nya. *Node* B tidak mengirimkan paket diduga karena waktu acak yang dihasilkan hanya berselisih 27 milidetik dari *Node* A. Namun, pada menit berikutnya, *Node* B berhasil melakukan *request* dan menerima *answer* sinkronisasi.

6.2.2.2 Analisis Pengujian Fungsionalitas Fase Sinkronisasi

Dari pengujian yang telah dilakukan, dapat dilakukan analisis yang membuktikan bahwa sinkronisasi dapat dilakukan setelah Proses *Discovery* selesai dilakukan, sehingga masing-masing *parent* akan bertanggung jawab untuk proses Sinkronisasi *child*-nya masing-masing. Salah satu kelemahan dari Algoritma TPSN adalah kemungkinan kecil terjadinya interferensi data pada proses *request* Sinkronisasi akibat waktu tunggu acak yang kemungkinan mendekati waktu *request node* lainnya. Kemungkinan interferensi bisa diperkecil dengan menambah rentang waktu tunggu acak yang dihasilkan, namun akan berakibat Fase Sinkronisasi yang berjalan lebih lama.

6.2.3 Fungsionalitas Fase TDMA

Pengujian fungsional TDMA digunakan untuk menunjukkan setiap *node* yang sudah tersinkronisasi akan mendapatkan *time slot* yang berbeda-beda. Untuk kemudian akan digunakan sebagai jadwal pengiriman paket. Selisih antar *time slot* bisa diubah-ubah dalam kode program. Prosedur pengujian ditunjukkan pada tabel 6.6.

Tabel 6.6 Prosedur Pengujian Fungsionalitas Fase TDMA

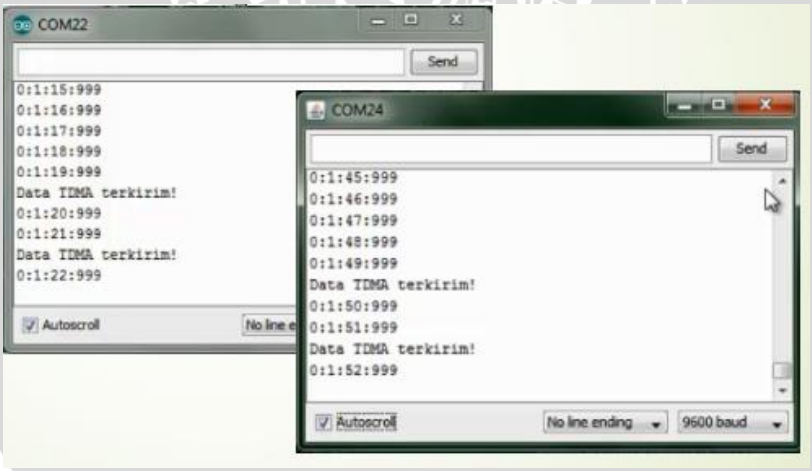
Kasus Pengujian	Proses pengiriman Paket TDMA.
Objek Pengujian	Keberhasilan pengiriman Paket TDMA tiap *node* sesuai *time slot* atau jadwal yang didapatkan.
Tujuan Pengujian	Mengamati kelancaran Paket TDMA yang dikirimkan.

Tabel 6.6 (Lanjutan)

Prosedur Pengujian	1. Penguji memasukkan kode program pada setiap *node* dalam pengujian (1 buah *Node Root* dan 2 buah *node* biasa). 2. Penguji menunggu semua *node* membentuk hierarki dengan Fase *Discovery.* 3. Penguji menunggu sistem menyelesaikan Fase Sinkronisasi. 4. Penguji mengamati keberhasilan pengiriman Paket TDMA sesuai jadwal dengan 2 pengirim dan 1 penerima. 5. Pengujian dilakukan selama 5 menit pada satu kali percobaan. 6. Pengujian dilakukan sebanyak 2 kali.

6.2.3.1 Hasil Pengujian Fungsionalitas Fase TDMA

Sampel proses pengujian Fase TDMA tampak pada gambar 6.6, dimana COM22 dan COM24 tertulis data TDMA terkirim. Keseluruhan hasil pengujian ditunjukkan pada Tabel 6.7.



Gambar 6.6 Tampilan Fase TDMA

Tabel 6.7 Hasil Pengujian Fungsionalitas Fase TDMA

Percobaan ke-	ID *Node*	Level *Node*	Time Slot	Waktu (MM:DD)	Status Pengiriman
1	A	1	10	01:10	✓
	B	1	20	01:20	✓
	A	1	10	02:10	✓

Tabel 6.7 (Lanjutan)

	B	1	20	02:20	✓
	A	1	10	03:10	✓
	B	1	20	03:20	✓
	A	1	10	04:10	✓
	B	1	20	04:20	✓
	A	1	10	05:10	✓
	B	1	20	05:20	✓
2	A	1	10	01:10	✓
	B	2	20	01:20	✓
	A	1	10	02:10	✓
	B	2	20	02:20	✓
	A	1	10	03:10	✓
	B	2	20	03:20	✓
	A	1	10	04:10	✓
	B	2	20	04:20	✓
	A	1	10	05:10	✓
	B	2	20	05:20	✓

Hasil pengujian menunjukkan semua Paket TDMA bisa terkirim sesuai jadwal yang diberikan.

6.2.3.2 Analisis Pengujian Fungsionalitas Fase TDMA

Dari pengujian yang telah dilakukan, maka dapat dilakukan analisis bahwa pengujian TDMA dapat membuktikan Fase *Discovery* dan Fase Sinkronisasi berhasil menunjang pengiriman Paket TDMA.

6.3 Pengujian Waktu Operasional

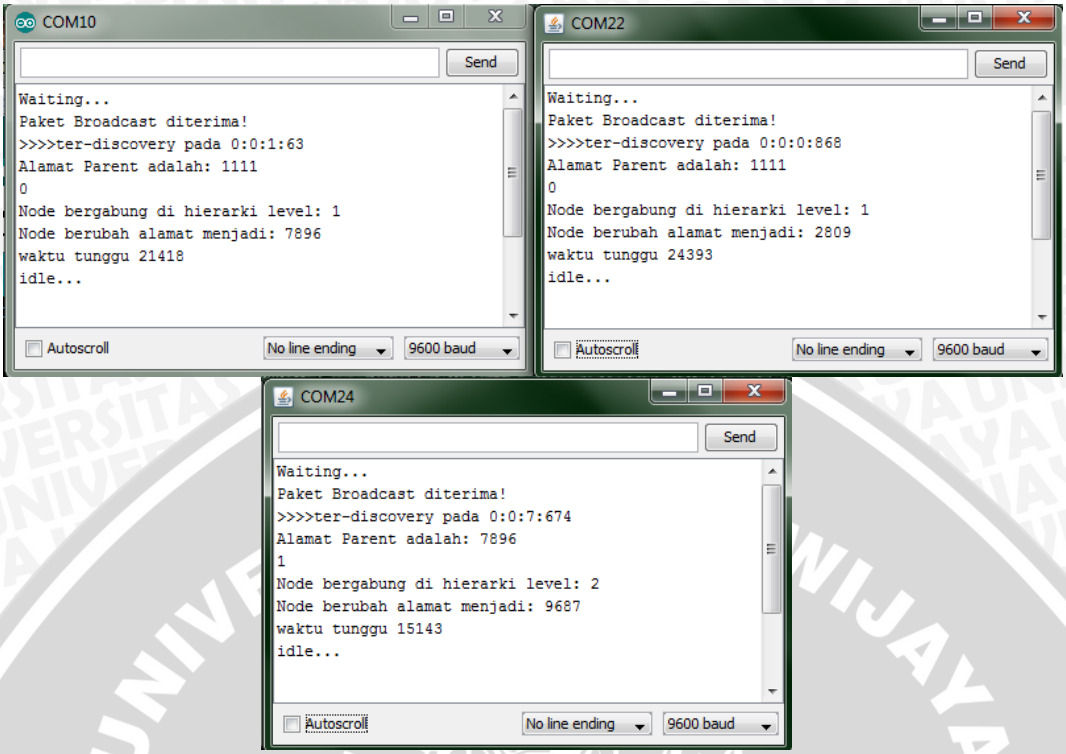
Pengujian waktu operasional digunakan untuk mengamati dan mengukur waktu yang dibutuhkan dari kedua fase: *discovery* dan sinkronisasi. Untuk melakukan pengujian waktu operasional, maka diperlukan prosedur seperti tertera pada tabel 6.8.

Tabel 6.8 Prosedur Pengujian Waktu Operasional

Kasus Pengujian	Pengukuran waktu operasional tiap-tiap fase sistem.
Objek Pengujian	Waktu yang dibutuhkan bagi tiap-tiap fase menyelesaikan prosesnya.
Tujuan Pengujian	Menentukan waktu rata-rata sistem beroperasi untuk kebutuhan non-fungsional.
Prosedur Pengujian	1. Penguji menambahkan variabel pengukur waktu eksekusi pada kode program, dan melakukan *upload* kode program ke Arduino Nano 2. Penguji mengaktifkan *node* berurutan sesuai *level* yang direncanakan, *node* diaktifkan setelah Fase *Discovery* pada hierarki urutan sebelumnya selesai dilakukan. (Urutan hierarki: 1 *Node* Root, 2 *node* level 1, 1 *node* level 2) 3. Penguji menunggu sampai Fase *Discovery* dan Sinkronisasi selesai dilakukan. 4. Penguji mencatat waktu eksekusi yang tercetak pada Serial Monitor.

6.3.1 Hasil Pengukuran Waktu *Discovery*

Proses *discovery* berguna untuk memberitahukan kepada setiap *node*, alamat dari *parent*-nya yang nanti akan dimintai paket sinkronisasi. Pengujian ini untuk mencatat waktu yang dibutuhkan bagi sebuah *node* untuk masuk ke hierarki, mengenali alamat *parent*-nya, dan mencatat levelnya sendiri. Waktu dihitung dari pertama *node* aktif sampai menerima paket *discovery* dari *parent*-nya. *Node level* 1 langsung menerima paket *discovery* dari *root* (*level* 0), dan *node* level 2 menerima paket *discovery* dari *node* level 1. Pada level 1, terdapat dua *Node*, sedangkan pada *level* 2 terdapat satu *Node*. Gambar 6.7 menunjukkan hasil percobaan 1, setelah ketiga *node* menerima paket *discovery*. COM10 dan COM22 menjadi level 1, dan COM24 menjadi level 2 yang merupakan *child* dari COM10. Percobaan diulang sampai tiga kali, dan hasilnya dapat dilihat pada table 6.1.



Gambar 6.7 *Node-node* setelah menerima paket *discovery*

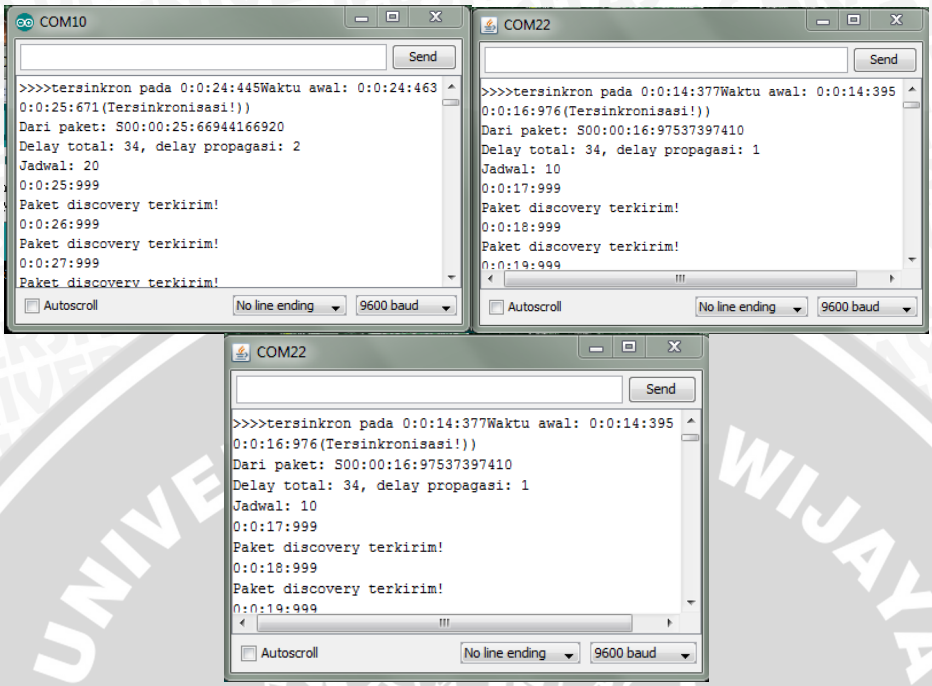
Tabel 6.9 Waktu untuk *discovery*

Percobaan ke-	Kode *Node*	Level *Node*	Waktu *Discovery*
1	A	1	1,630 detik
	B	1	0,868 detik
	C	2	7,674 detik
2	A	1	0,994 detik
	B	1	0,681 detik
	C	2	9,207 detik
3	A	1	0,710 detik
	B	1	0,293 detik
	C	2	9,260 detik

6.3.2 Hasil Pengukuran Waktu Sinkronisasi

Proses sinkronisasi dilakukan setelah proses *discovery* selesai. Setiap *node* akan menunggu selama kurun waktu yang acak guna memperkecil kemungkinan interferensi dengan *node* lain saat melakukan *request* sinkronisasi. Pada proses sinkronisasi, setiap *node* akan meminta *timestamp* ke *parent*-nya, untuk kemudian dicocokkan dengan waktu lokalnya sendiri. Percobaan pertama pada gambar 6.8 menunjukkan proses penerimaan paket balasan sinkronisasi dan

pencocokan waktu dari ketiga *node*. Percobaan untuk proses sinkronisasi tersebut dilakukan sebanyak tiga kali ditunjukkan pada tabel 6.10.



Gambar 6.8 *Node-node* setelah mendapatkan paket balasan sinkronisasi

Tabel 6.10 Waktu untuk sinkronisasi

Percobaan ke-	Kode *Node*	Level *Node*	Waktu dari awal *node* aktif	Waktu dari awal percobaan
1	A	1	24,445 detik	25,671 detik
	B	1	14,377 detik	16,976 detik
	C	2	21,849 detik	30,585 detik
2	A	1	23,462 detik	25,631 detik
	B	1	15,606 detik	19,263 detik
	C	2	22,626 detik	31,707 detik
3	A	1	14,936 detik	16,387 detik
	B	1	20,024 detik	23,002 detik
	C	2	20,907 detik	28, 893 detik

6.3.3 Analisis Pengukuran Waktu

Setelah pengujian pengukuran waktu dilakukan dan didapatkan hasilnya, maka dilakukan analisis bahwa Proses *Discovery* membutuhkan waktu rata-rata dari tiga pengujian sesuai perhitungan sebagai berikut.

1. Rata-rata waktu pembentukan level 1 (2 *node*):
   (1,630 + 0,868 + 0,994 + 0,681 + 0,710 + 0,293)/6 = 0,863 detik

2. Rata-rata waktu pembentukan level 2 (1 *node*):
   (7,674 + 9,207 + 9,260)/3 = 8,713 detik

Dari hasil pengujian Proses Sinkronisasi, didapatkan hasil pengujian rata-rata waktu yang dibutuhkan untuk sebuah level tersinkronisasi adalah sebagai berikut:

1. Rata-rata waktu sinkronisasi level 1 (2 *node*):
   - Dihitung dari awal *node* aktif:
     (24,445 + 14,377 + 23,462 + 15,606 + 14,936 + 20,024)/6 = 18, 809 detik
   - Dihitung dari awal percobaan:
     (25,671 + 16,976 + 25,631 + 19,263 + 16,387 + 23,002)/6 = 21,155 detik
2. Rata-rata waktu sinkronisasi level 2 (1 *node*):
   - Dihitung dari awal *node* aktif:
     (21,849 + 22,626 + 20,907)/3 = 21,794 detik
   - Dihitung dari awal percobaan:
     (30,585 + 31,707 + 28, 893) /3 = 30,395 detik

6.4 Pengujian fleksibilitas topologi jaringan

Pengujian fleksibilitas digunakan untuk menguji apakah program yang dibuat, tetap dapat digunakan jika struktur topologi jaringan diubah. Pengujian menggunakan 4 *node* dengan anggota tiap level berbeda dalam tiap skenario.

Pengujian fleksibilitas topologi menggunakan tiga skenario, yang masing-masing berbeda struktur topologinya. Ketiga skenario tersebut sebagai berikut.

1. 4 *node*, dengan susunan: 1 *Node Root*, 2 *node* level 1, 1 *node* level 2.
2. 4 *node*, dengan susunan: 1 *Node Root*, 3 *node* level 1.
3. 4 *node*, dengan susunan: 1 *Node Root*, 1 *node* level 1, 2 *node* level 2.

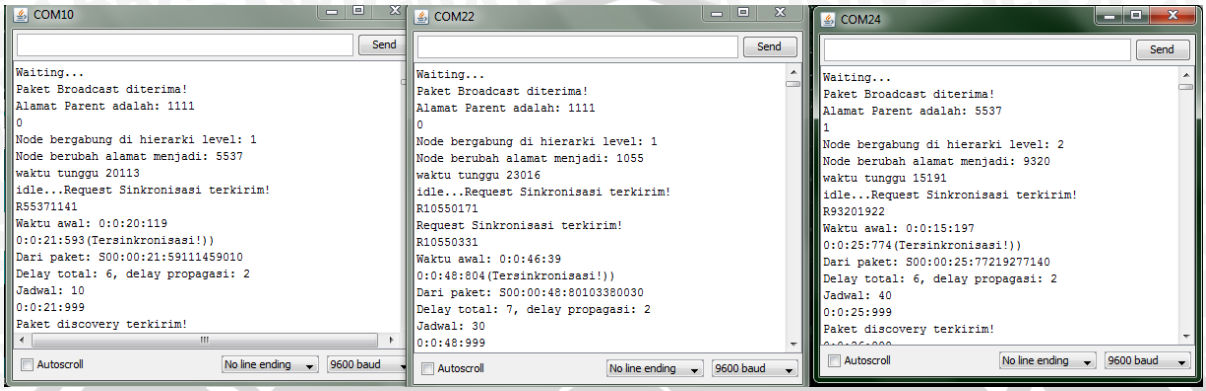
Pengamatan dilakukan dari keberhasilan pengiriman TDMA kepada *Node Root.* Untuk keterangan lengkap, ditunjukkan pada tabel 6.11.

Tabel 6.11 Prosedur Pengujian Fleksibilitas Topologi

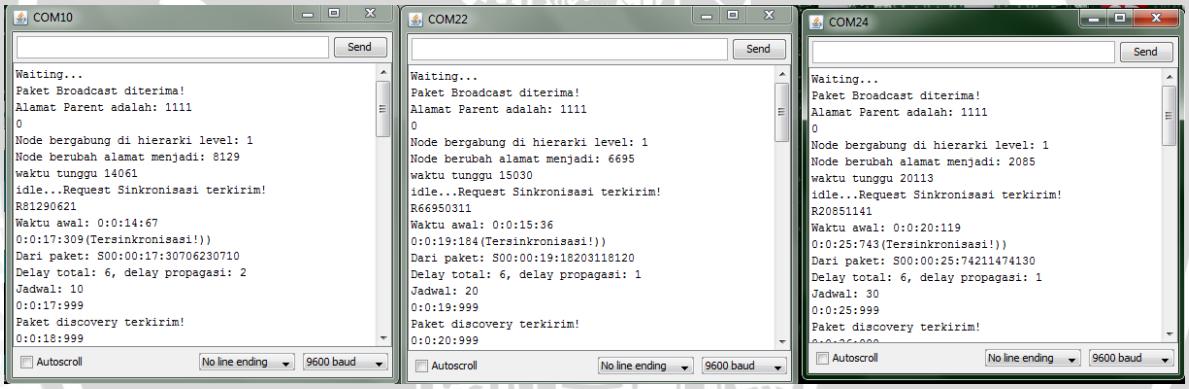
Kasus Pengujian	Pengujian fleksibilitas topologi jaringan.
Objek Pengujian	Keberhasilan pengiriman TDMA dengan berbagai struktur topologi berbeda.
Tujuan Pengujian	Mengetahui kemampuan program yang dibuat untuk mengatasi fleksibilitas dari struktur topologi yang bervariasi.
Prosedur Pengujian	1. Penguji memasukkan kode program ke *node* WSN. 2. Penguji melakukan skenario 1, 2, dan 3 secara bergantian. 3. Penguji mengamati keberhasilan pengiriman TDMA pada tiap skenario.

6.4.1 Hasil Pengujian Fleksibilitas Topologi

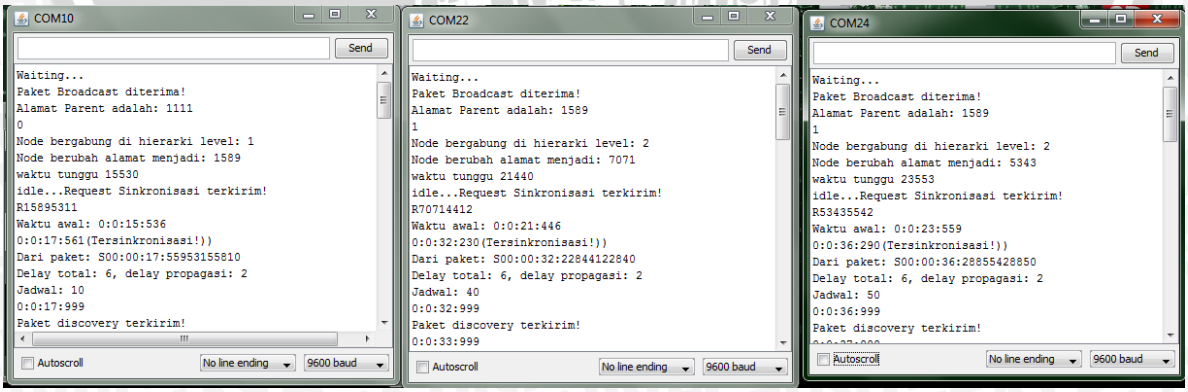
Tampilan *Serial Monitor* dari *node* level 1 dan level 2 hasil pengujian skenario 1, seperti ditunjukkan pada gambar 6.9, 6.10, dan 6.11, dimana ketiga *node* berhasil menempati level masing-masing. *Node* COM24 (alamat 9320) memasuki level 2 dan menjadi *child* dari *Node* COM10 (alamat 5537). Kemudian hasil pengujian ketiga skenario ditunjukkan pada tabel 6.12.



Gambar 6.9 Susunan *node-node* pengujian fleksibilitas skenario 1



Gambar 6.10 Susunan *node-node* pengujian fleksibilitas skenario 2



Gambar 6.11 Susunan *node-node* pengujian fleksibilitas skenario 3

Tabel 6.12 Pengamatan pengiriman TDMA dari 3 skenario

Skenario ke-	Kode *Node*	Level *Node*	Jadwal TDMA / Slot Waktu	Status Penerimaan TDMA pada *Node* Root
1	A	1	Detik ke-10	Diterima
	B	1	Detik ke-30	Diterima
	C	2	Detik ke-40	Diterima
2	A	1	Detik ke-10	Diterima
	B	1	Detik ke-20	Diterima
	C	1	Detik ke-30	Diterima
3	A	1	Detik ke-10	Diterima
	B	2	Detik ke-40	Diterima
	C	2	Detik ke-50	Diterima

6.4.2 Analisis Pengujian Fleksibilitas Topologi

Dari hasil pengujian, didapatkan analisis sebagai berikut, bahwa peralatan dapat bekerja dengan baik, dan sistem telah berjalan sesuai dengan keinginan dari proses awal sampai akhir bahkan ketika struktur jaringan diubah. Proses *Discovery* dapat menghasilkan suatu hierarki *tree* berdasarkan model pengalamatan pada suatu *node* yang dicatat oleh *child*-nya, dan juga menggunakan penomoran level yang urut.

Hasil dari pengujian fleksibilitas program yang dibuat, terhadap perubahan struktur topologi membuktikan bahwa program TPSN tersebut fleksibel serta masih bisa berjalan untuk melakukan sinkronisasi waktu dan pengiriman TDMA sesuai dengan penjadwalan atau *time slot* yang diberikan. Program tetap bisa digunakan selama *time slot* masih ada atau belum digunakan oleh suatu *node*. *Node* level 1 selalu mendapatkan *time slot* dari angka 10 dan kelipatan 10. *Node* level 2 selalu mendapatkan *time slot* dimulai dari *time slot* yang dia miliki ditambah 30, dan berlaku kelipatan 10.

Untuk beberapa kasus seperti pengujian pada Gambar 6.9, *node* dengan port COM22 seharusnya mendapatkan *time slot* 20, namun justru mendapatkan nilai 30 dikarenakan *node* tersebut gagal dalam mendapatkan *timestamp* sinkronisasi pada *request* yang pertama, sehingga harus mengulangi *request* berikutnya. Kegagalan *request* sinkronisasi bisa disebabkan oleh waktu tunggu suatu *node* yang hampir sama dengan *node* lain, yang berakibat adanya tubrukan data sewaktu proses pengiriman.

6.5 Pengujian Akurasi Waktu

Pengujian akurasi waktu digunakan untuk mengetahui tingkat akurasi sinkronisasi waktu pada pengiriman TDMA. Waktu akan diukur ketika paket TDMA diterima pada sisi penerima dan akan dibandingkan dengan *time slot,* dengan

asumsi *delay* propagasi tidak lebih dari 5 milidetik. Pengujian dilakukan menggunakan jarak antar *time slot* yang berbeda-beda pada tiap skenario. Pengujian akurasi waktu menggunakan 3 skenario sebagai berikut.

1. Skenario TDMA dengan jarak *time slot* 5 detik
2. Skenario TDMA dengan jarak *time slot* 3 detik
3. Skenario TDMA dengan jarak *time slot* 1 detik

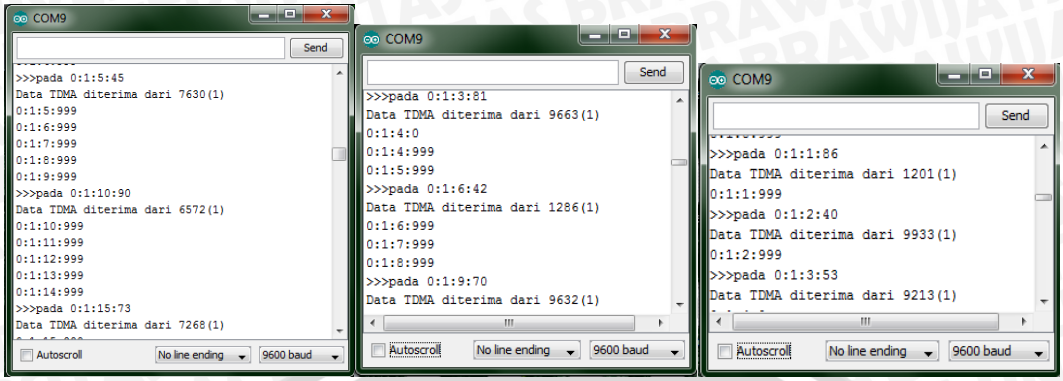
Pengujian secara keseluruhan terdapat pada tabel 6.13.

Tabel 6.13 Prosedur Akurasi Waktu

Kasus Pengujian	Pengujian pengukuran akurasi waktu.
Objek Pengujian	Waktu penerimaan dibandingkan waktu *time slot*.
Tujuan Pengujian	Mengetahui kemampuan ketepatan waktu program yang dibuat untuk melakukan skenario TDMA dengan jarak *time slot* yang semakin sempit.
Prosedur Pengujian	1. Penguji menambahkan variabel pengukur waktu pada program *Node* penerima (*Root*). 2. Penguji melakukan *upload* program pada semua *node* WSN. 3. Penguji mengamati sistem melakukan algoritma TPSN dan melakukan skenario pengiriman TDMA 4. Penguji mencatat waktu penerimaan dan membandingkan dengan *time slot* yang sudah ada. 5. Penguji mengulangi percobaan pada nomor 1 dengan skenario berbeda, sampai keseluruhan skenario selesai diuji coba.

6.5.1 Hasil Pengujian Akurasi Waktu

Dari pengujian yang dilakukan, didapatkan hasil seperti pada gambar 6.12. 6.12.a adalah hasil skenario 1, 6.12.b adalah hasil skenario 2, dan 6.12.c adalah hasil skenario 3. Secara keseluruhan, hasil pengujian terdapat pada tabel 6.14.



(a) (b) (c)

Gambar 6.12 Hasil Pengujian Akurasi Waktu

Tabel 6.14 Hasil Pengujian Akurasi Waktu

Skenario ke-	Jarak Slot Waktu	Kode *Node*	Level *Node*	Jadwal TDMA / Slot Waktu	Waktu Penerimaan (detik)
1	5	A	1	Detik ke-5	5,045
		B	1	Detik ke-10	10,090
		C	2	Detik ke-15	15,073
2	3	A	1	Detik ke-3	3,081
		B	1	Detik ke-6	6,042
		C	2	Detik ke-9	9,070
3	1	A	1	Detik ke-1	1,086
		B	1	Detik ke-2	2,040
		C	2	Detik ke-3	3,053

Dari tabel 6.14, dapat diambil kesimpulan bahwa semua *node* berhasil mengirim pada *time slot* yang telah dijadwalkan. Meskipun setelah dikurangi *delay* propagasi, dan hasilnya sedikit terlambat dari jadwal beberapa puluh milidetik, namun pengiriman TDMA berhasil dilakukan tanpa adanya interferensi.

6.5.2 Analisis Pengujian Akurasi Waktu

Dari pengujian yang telah dilakukan, dapat diambil analisis bahwa, pengiriman TDMA tetap bisa dilakukan meskipun menggunakan jadwal slot waktu yang sempit tanpa ada interferensi. Dan angka pengiriman sedikit terlambat beberapa puluh milidetik dari jadwal bisa disebabkan karena pengiriman waktu serial pada PC yang menimbulkan *delay* sewaktu akan dicetak

BAB 7 PENUTUP

Bab ini akan menjelaskan tetang kesimulan yang diperoleh dari hasil pengujian dan analisa hingga saran yang perlu ditambahkan untuk penelitian selanjutnya.

7.1 Kesimpulan

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

1. Perancangan sistem sesuai alur yang telah dibuat pada kerangka berpikir, dimana pengujian fungsionalitas menunjukkan sistem yang telah dibuat sudah memenuhi persyaratan kebutuhan, sehingga tidak perlu mengulangi untuk memperbaiki langkah perancangan perangkat keras dan perancangan perangkat lunak.
2. *Library* Mirf yang digunakan bisa melakukan skenario transmisi data, penerimaan data, pengaturan *channel,* pengalamatan berdasarkan ID, dan konfigurasi ukuran *payload* untuk mendukung data array yang digunakan pada sistem.
3. Metode pengacakan waktu dan alamat berhasil mendukung sebagian besar skenario TPSN dan TDMA, meskipun ada beberapa kegagalan kecil pada sejumlah skenario yang menjadi kelemahan, namun dapat dikurangi dengan jumlah byte acak yang ditambah, meskipun dengan resiko waktu eksekusi yang dapat berjalan lebih lama.
4. Pembentukan hierarki sampai level 2 kurang dari 10 detik. Dengan rata-rata level 1 kurang dari 2 detik, dan level 2 kurang dari 9 detik.
5. Rata-rata waktu yang dibutuhkan untuk melakukan sinkronisasi sampai level 1 adalah 21,155 sejak awal percobaan, dan level 2 membutuhkan waktu rata-rata sekitar 30,395 detik sejak awal percobaan.
6. Pengiriman TDMA berjalan sesuai *timeslot* yang direncakanan, dengan akurasi waktu diukur dari selisih penerimaan dibandingkan dengan jadwal pengiriman mencapai maksimum 0,090 detik, dan minimum 0,040 detik dari percobaan 9 kali pengiriman menggunakan jarak *slot* waktu yang berbeda-beda.
7. Program yang dibuat fleksibel, sehingga perubahan struktur hierarki tidak akan mempengaruhi keberhasilan algoritma TPSN dan pengiriman TDMA selama persediaan *time slot* masih ada.

7.2 Saran

Dari perancangan, implementasi dan pengujian sistem, sistem telah berjalan dengan baik dan semestinya. Namun sistem ini masih perlu adanya penyempurnaan pada beberapa aspek, seperti pada proses sinkronisasi yang tidak dapat mengenali adanya interferensi pengiriman data, keterbatasan jumlah *node* yang bisa ditambahkan, dan sebagainya. Maka penulis berharap adanya beberapa penyempurnaan untuk menghasilkan implementasi algoritma TPSN dan TDMA yang lebih sempurna dengan menggunakan metode yang berbeda. Serta penulis berharap program ini dapat dikembangkan untuk beberapa keperluan lainnya sehingga dapat digunakan pada sistem yang lebih besar.

DAFTAR PUSTAKA

Arduino, 2015. Product Arduino. [online] Tersedia di: <http://www.arduino.cc/en/Main/Products> [Diakses 20 April 2015]

Chantrell, N., 2013. *Experimenting with the nRF24L01+ 2.4GHz radios.* [Online] Tersedia di: https://nathan.chantrell.net/20130810/experimenting-with-the-nrf24l01-2-4ghz-radios/ [Diakses pada Maret 2016].

Chu, Y. et al., 2015. Application of reinforcement learning to medium access control for wireless sensor networks. *Engineering Applications of Artificial Intelligence,* pp. 23-32.

Elson, J., and Estrin, D. 2001*. Time synchronization for wireless sensor networks*. San Francisco, CA.

Elson, J., Girod, L., and Estrin, D. 2002. *Fine-grained network time synchronization using reference broadcasts*. In UCLA Technical Report 020008.

Faludi, Robert. 2010. *Building Wireless Sensor Networks.*

Kumar, S. & Chauhan, S., 2011. A Survey on Scheduling Algorithms for Wireless. *International Journal of Computer Applications.*

Kurose, J., and Ross, K. 2012. *Computer Networking: A Top-Down Approach Sixth Edition.* Boston, Addison-Wesley.

Mathivanan, N. 2007. *PC-Based Instrumentation: Concept and Practice.* New Delhi, PHI.

Sohraby, dkk. 2007. *Wireless Sensor Networks: Technology, Protocols, and Applications.* Canada. Wiley.

Stallings, William. 2005. *Wireless Communication and Networks Second Edition*. Pearson Education, Inc. Sebastopol. O'Reilly.