Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR – IF184802

RANCANG BANGUN CLUSTER-BASED PROTOKOL UNTUK PENGIRIMAN DATA SECARA ADAPTIF DENGAN PENGATURAN KEKUATAN TRANSMISI DAN MONITORING KETERSEDIAN ENERGI PADA LINGKUNGAN WIRELESS SENSOR NETWORK DENGAN nRF24L01

ZAHRI RUSLI
NRP 05111540000108

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

Dosen Pembimbing II
Tohari Ahmad, S.Kom., MIT., Ph.D.

TUGAS AKHIR – IF184802

RANCANG BANGUN CLUSTER-BASED PROTOKOL UNTUK PENGIRIMAN DATA SECARA ADAPTIF DENGAN PENGATURAN KEKUATAN TRANSMISI DAN MONITORING KETERSEDIAN ENERGI PADA LINGKUNGAN WIRELESS SENSOR NETWORK DENGAN nRF24L01

ZAHRI RUSLI
NRP 05111540000108

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

Dosen Pembimbing II
Tohari Ahmad, S.Kom., MIT., Ph.D.

(Halaman ini sengaja dikosongkan)

UNDERGRADUATE THESIS – IF184802

DESIGN AND DEVELOPMENT OF CLUSTER-BASED PROTOCOLS FOR ADAPTIVE DATA TRANSMITTING WITH TRANSMISSION POWER ARRANGEMENT AND ENERGY MONITORING IN WIRELESS SENSOR NETWORK ENVIRONMENT USING WITH nRF24L01

ZAHRI RUSLI
NRP 05111540000108

First Advisor
Waskitho Wibisono, S.Kom., M.Eng., Ph.D.

Second Advisor
Tohari Ahmad, S.Kom., MIT., Ph.D.

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

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

RANCANG BANGUN *CLUSTER-BASED* PROTOKOL UNTUK PENGIRIMAN DATA SECARA ADAPTIF DENGAN PENGATURAN KEKUATAN TRANSMISI DAN *MONITORING* KETERSEDIAN ENERGI PADA LINGKUNGAN WIRELESS SENSOR NETWORK DENGAN nRF24L01

TUGAS AKHIR

Diajukan untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Arsitektur Jaringan Komputer
Program Studi S-1 Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh:
ZAHRI RUSLI
NRP: 05111540000108

Disetujui oleh Pembimbing tugas akhir:

1. Waskitho Wibisono, S.Kom., M.Eng., Ph.D. .....
   (NIP. 197410222000031001) (Pembimbing 1)

2. Tohari Ahmad, S.Kom., MIT., Ph.D. .....
   (NIP. 197505252003121002) (Pembimbing 2)

SURABAYA
JUNI, 2019

(Halaman ini sengaja dikosongkan)

RANCANG BANGUN *CLUSTER-BASED* PROTOKOL UNTUK PENGIRIMAN DATA SECARA ADAPTIF DENGAN PENGATURAN KEKUATAN TRANSMISI DAN *MONITORING* KETERSEDIAN ENERGI PADA LINGKUNGAN WIRELESS SENSOR NETWORK DENGAN nRF24L01

Nama Mahasiswa	**:**	**Zahri Rusli**
NRP	**:**	**05111540000108**
Departemen	**:**	**Informatika FTIK ITS**
Dosen Pembimbing 1	**:**	**Waskitho Wibisono, S.Kom., M.Eng., Ph.D.**
Dosen Pembimbing 2	**:**	**Tohari Ahmad S.Kom., MIT., Ph.D.**

Abstrak

Wireless Sensor Network (WSN) adalah sebuah jaringan yang terdiri dari node-node yang di distribusikan secara random mapun strategis degan tujuan untuk mengamati suatu kejadian tertentu.

Modul Wireless nRF24l01 adalah sebuah modul komunikasi yang dapat digunakan untuk WSN. Modul ini memanfaatkan pita gelombang RF 2.4 GHz ISM (*Industrial Scientific and Medical*) dan menggunakan antar muka SPI (*Serial Peripheral Interface*) untuk berkomunikasi. Modul ini didesain untuk jaringan nirkabel yang membutuhkan daya sangat rendah. Sehingga cocok digunakan pada *node-node* dalam *wireless sensor network.*

Pada jaringan sensor yang akan di bua, diharapkan semua *node* dapat mengirimkan data ke *sever/coordinator*. Namun apabila setiap *node* mengirimkan langung kepada *coordinator* akan terjadi pemborosan dalam penggunaan *energy* dikarenakan kekuataran transmisi dari setiap *node* di atur menjadi maximal agar mampu mengrimkan data jarak jauh ke *coordinator.* Sehingga

perlunya ada *node* yang menjadi *cluster head* untuk mengrimkan data ke *coordinator*. Namun *node* yang menjadi *cluster head* akan membutuhkan *energy* yang sangat besar dikarenakan harus mengrimkan data jarak jauh dan besar, sehingga *node* akan cepat kehabisan *energy*. Sehingga dibutuhkan pergantian *cluster head* agar dapat menyeimbangkan penggunaan energi di dalam *cluster node*.

Metode yang penulis gunakan dalam melakukan pemilihan *cluster head* adalah dengan menentukan sebuah *node* yang akan menjadi cluster head pertama. Kemudian setelah beberapa saat node yang menjadi *cluster head* akan mengrimkan pesan berhenti menjadi *cluster head*. Kemudian setiap *node* akan melakukan *broadcast* jumlah energi yang digunakan yang di dapatkan dari sensor INA219. Setelah itu energi yang di *broadcast* di urutkan, dan node yang memiliki penggunaan energi terkecil akan menjadi *cluster head* berikutnya.

Hasil dari implementasi menggunakan metode pemilihan c*luster head* menggunakan parameter jumlah pengguan energi yang digunakan oleh masing-masing *node* terjadi penurunan *packet lost* dan penurunan penggunaan energi. Sehingga cocok untuk di implementasikan pada lingkungan *Wireless Sensor Network*.

Kata kunci: ***Wireless Sensor Network, Cluster Head, Cluster node, nRF24.***

BUILD CLUSTER-BASED PROTOCOLS FOR ADAPTIVE DATA TRANSMISSION BY SETTING THE POWER OF TRANSMITTING AND MONITORING AVAILABILITY ENERGY IN THE WIRELESS SENSOR NETWORK ENVIRONMENT WITH nRF24L01

Student's Name : Zahri Rusli
Student's ID : 05111540000108
Department : Informatika FTIK-ITS
First Advisor : Waskitho Wibisono, S.Kom., M.Eng., Ph.D.
Second Advisor : Tohari Ahmad, S.Kom., MIT., Ph.D.

Abstract

Wireless Sensor Network (WSN) is a network of nodes that are distributed strategically in a random manner that aims to observeaparticularevent.

Wireless Module nRF24l01 is a communication module that can be used for WSN. The module utilizes the 2.4 GHz RF band of ISM (Industrial Scientific and Medical) and uses the SPI (Serial Peripheral Interface) interface to communicate. This module is designed for wireless networks that require very low power. So it is suitable for use on the nodes in network sensor network.

The method used to make a choice on cluster head is by determining a node that will be the first cluster head. After a while, this cluster head node will send a packet message to stop being cluster head node. Then every nodes will be broadcasting their usage of total energy from sensor INA219. Those data of total energy will be sorted, and the node with smallest usage of total energy will be chosen as the next cluster head.

The result by implementing this method is : there is a decreasing amount of packet lost, and the energy used is also reduced. Therefore this method is convenient to be implemented in Wireless Sensor Network environment.

Keyword: Wireless Sensor Network, Cluster Head, Cluster node, nRF24.

KATA PENGANTAR

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Puji syukur kepada Allah Yang Maha Esa atas segala karunia dan rahmat-Nya sehingga penulis dapat menyelesaikan tugas akhir yang berjudul

"RANCANG BANGUN *CLUSTER-BASED* PROTOKOL UNTUK PENGIRIMAN DATA SECARA ADAPTIF DENGAN PENGATURAN KEKUATAN TRANSMISI DAN *MONITORING* KETERSEDIAN ENERGI PADA LINGKUNGAN WIRELESS SENSOR NETWORK DENGAN nRF24L01".

Harapan dari penulis, semoga apa yang tertulis di dalam buku tugas akhir ini dapat bermanfaat bagi pengembangan ilmu pengetahuan saat ini dan ke depannya, serta dapat memberikan kontribusi yang nyata.

Dalam pelaksanaan dan pembuatan tugas akhir ini tentunya sangat banyak bantuan yang penulis terima dari berbagai pihak, tanpa mengurangi rasa hormat penulis ingin mengucapkan terima kasih sebesar-besarnya kepada:

1. Allah SWT. Dan Nabi Muhammad SAW. yang telah membimbing penulis selama hidup.
2. Keluarga penulis (Ayah, Mama, Zahra Rusli, Ade Rusli dan keluarga penulis yang lain) yang selalu memberikan dukungan baik berupa doa, moral, dan material yang tak terhingga kepada penulis, sehingga penulis dapat menyelesaikan tugas akhir ini.
3. Bapak Waskitho Wibisono, S.Kom., M.Eng.,Ph.D. dan Bapak Tohari Ahmad, S.Kom., MIT., Ph.D. selaku Dosen Pembimbing penulis yang telah membimbing, memberikan nasihat, dan memotivasi penulis sehingga penulis dapat menyelesaikan tugas akhir ini.
4. Bapak Dr.Eng. Darlis Herumurti, S.Kom., M.Kom. selaku kepala Departemen Informatika ITS.

5. Bapak dan Ibu Dosen yang telah memberikan ilmunya selama penulis berkuliah di Informatika ITS.
6. Yayasan Karya Salemba Empat yang telah memberikan beasiswa kepada penulis selama penulis berkuliah di informatika ITS.
7. Teman-teman penulis GAES yang selalu memberikan semangat secara tidak langsung kepada penulis, selalu memberikan hiburan, selalu menemani hari-hari penulis saat senang maupun susah, dan juga menjadi keluarga baru penulis saat berkuliah di Departemen Informatika ITS.
8. Sahabat baik penulis yang tidak perlu disebutkan Namanya yang selalu support dan mendukung penulis selama bertahun-tahun.
9. Barisan para gebetan yang sudah mewarnai dan mengisi hidup penulis selama proses perkuliahan
10. Teman-teman dari keluarga besar Laboratorium NCC (Hero, Mak, Yoga, Dely, Zulfa, Ubut, Zayn, Ical, Azki, Nuza, Akmal, Siraj, Adin, Wasil) yang telah menemani, memberi semangat, doa, serta hiburan dikala penulis sedang jenuh saat pengerjaan tugas akhir ini.
11. Teman-teman para penghuni lantai 3 sahabat MLM KBJ (Tegar, Nopal, Pentol, Adib, Ivan, Byan, Hero, Huda, Rezky, Alvin, Djohan, Adit, Yasin) yang sudah menemani kehidupan penulis selama pengerjaan tugas akhir ini.
12. Teman Teman Internship ElifeSolutions Keluarga baru selama penulis internship di johor Malaysia (Adib, Rezky, Bg rey, Ulung, Daeng, Faiz, Rouf, Aam, Heng dan Ping)
13. Teman-teman minang yang selalu bermain bliard Bersama penulis (Didi, Harits, Arif, Evan, Adit, Budi).
14. Teman-teman angkatan 2015 (Masamalas) yang sudah menjadi saksi hidup perjalanan karir penulis selama berkuliah di Informatika ITS.
15. Ibu-Ibu CS yang sangat baik dan memastikan lantai 3 selalu menjadi tempat yang nyaman untuk penulis untuk hidup di lantai 3 departemen informatika.

16. Ibu Kantin lantai 3 yang selalu memastikan penulis untuk sarapan setiap pagi selama berkuliah di departemen informatika.
17. Untuk orang-orang yang tidak dapat disebutkan satu persatu oleh penulis dan pembaca buku tugas akhir ini.

Penulis telah berusaha sebaik-baiknya dalam menyusun tugas akhir ini. Namun, penulis memohon maaf apabila terdapat kekurangan, kesalahan maupun kelalaian yang telah penulis lakukan. Kritik dan saran yang membangun dapat disampaikan sebagai bahan perbaikan selanjutnya. Tetap semangat dalam menjalani kehidupan, jangan menyerah, karena Allah masih ingin melihat kita berjuang. Semoga kita semua selalu diberi kebahagiaan lahir dan batin dan kesuksesan dunia akhirat. Aamiin.

Surabaya, 25 Juni 2019

Zahri Rusli

(Halaman ini sengaja dikosongkan)

DAFTAR ISI

(Halaman ini sengaja dikosongkan)

DAFTAR GAMBAR

(Halaman ini sengaja dikosongkan)

DAFTAR TABEL

KODE SUMBER

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi komunikasi saat ini teleh berkembang dengan sangat pesat, salah satunya dibidang *Wireless Sensor Network* (WSN). WSN adalah sebuah jaringan yang terdiri dari node-node yang didistribusikan baik secara strategi maupun secara acak yang bertujuan untuk mengamati suatu kejadian tertentu. WSN sudah dianggap sebagai salah satu penemuan yang sangat penting karena harganya murah dan handal.

Modul Wireless nRF24L01 adalah sebuah modul komunikasi *short range* yang memanfaatkan pita gelombang RF 2.4GHz ISM (Industrial, Scientific and Medical) sebagai media komunikasinya. Modul ini menggunakan anatarmuka SPI (Serial Peripheral Interface untuk berkomunikasi dengan mikrokontroler sehingga modul dapat digunakan dengan mikrokontroler manapun yang mendukung SPI. Modul nRF24L01 menggunakan daya sangat rendah sehingga cocok digunakan pada node-node dalam jaringan sensor yang biasanya ditenagai daya baterai yang terbatas.

Pada jaringan sensor yang akan dibuat, diharapkan semua node dapat mengirimkan data ke sebuah server atau *coordinator*. Namun, apabila semua node mengirmkan langsung ke server atau *coordinator*, maka penggunaan energi dalam jaringan akan menjadi boros dikarenakan semua node akan menggunakan kekuatan transmisi tinggi untuk mengirim data ke server atau *coordinator*. Sehingga, perlu adanya node yang menjadi *Cluster Head* dalam jaringan untuk mengumpulkan data dari node lain sebelum mengirimkannya ke server atau *coordinator*, sehingga cukup satu *node* yang menggunakan kekuatan transmisi tinggi. Namun, *node* yang menjadi *cluster head* akan cepat kehabisan energi yang mengakibatkan energi dalam jaringan menjadi tidak seimbang. Maka diperlukan sebuah metoda pemilihan *cluster head* yang efisien untuk menangani permasalahan tersebut.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana mengatur kekuatan transmisi *cluster* member dan *cluster head* secara *adaptive* berdasarkan ketersedian energi pada node?
2. Bagaimana melakukan monitoring terhadap perubahan energi pada masing-masing *node*?

1.3 Batasan Permasalahan

Berdasarkan masalah yang diuraikan oleh penulis, maka batasan masalah pada tugas akhir ini adalah:

1. Menggunakan *platform* arduino sebagai mikrokontroller
2. Menggunakan modul nRF24L01 sebagai media transmisi data antar *node* pada jaringan sensor
3. Menggunakan *Sensor INA219* sebagai sensor untuk menguku jumlah energi yang terdapat pada masing-masing node.
4. Menggunakan baterai sebagai sumber energi pada setiap *node*

1.4 Tujuan

Tujuan dari pembuatan tugas akhir ini adalah sebagai berikut:

1. Membangun sebuah jaringan sensor nirkabel yang dapat berkoordinasi secara *adaptive* dan *dinamis* dalam pemilohan *cluster head* yang daoat menyeimbangkan penggunaan energi dalam jaringan tersebut.
2. Melakukan monitoring terhadap perubahan energi pada masing-masing *node.*

1.5 Manfaat

Manfaat yang diperoleh dari pengerjaan tugas akhir ini adalah menghasilkan sebuah jaringan sensor nirkabel yang dapat berkoordinasi secara *adaptive* dan *dinamis* sert memiliki penggunaan energi yang seimbang dan tahan lama.

1.6 Metodologi

Pembuatan tugas akhir ini dilakukan dengan menggunakan metodologi sebagai berikut:

1.6.1 Penyusunan Proposal Tugas Akhir

Proposal tugas akhir ini berisi gambaran tentang tugas akhir yang akan dibuat. Pendahuluan proposal tugas akhir meliputi hal yang menjadi latar belakang diajukannya usulan tugas akhir, rumusan masalah yang diangkat, batasan masalah yang menjadi konstrain dari tugas akhir, tujuan pembuatan tugas akhir, dan manfaat dari hasil tugas akhir. Di dalam proposal tugas akhir juga dijabarkan mengenai tinjauan pustaka yang menjadi referensi pendukung dalam pembuatan tugas akhir ini. Terdapat pula sub bab jadwal kegiatan yang menjelaskan jadwal pengerjaan tugas akhir.

1.6.2 Studi Literatur

Studi literatur yang dilakukan dalam pengerjaan Tugas Akhir ini adalah mengenai Rancang Bangun *Cluster-Based* Protokol untuk pengiriman data secara adaptif dengan pengaturan kekuatan transmisi dan *monitoring* ketersedian energy pada lingkungan *Wireless Sensor Network* dengan Menggunakan nRF2L01. Sehingga, studi literature ini dapat diterapkan pada perancangan jaringan sensor nirkabel yang dapat melakukan pemilihan *cluster head* secara *adaptive* dan *dinamis*.

1.6.3 Implementasi Sistem

Implementasi merupakan tahap untuk mengimplementasikan metode-metode yang sudah diajukan pada proposal tugas akhir. Untuk membangun algoritma yang telah dirancang sebelumnya, implementasi dilakukan dengan menggunakan NS-2 sebagai simulator jaringan, bahasa C/C++ sebagai bahasa pemrograman untuk uji coba mengimplementasikan metode yang sudah diajukan.

1.6.4 Pengujian dan Evaluasi

Pengujian dan evaluasi dari hasil Tugas Akhir ini akan diujicobakan dengan membangun jaringan sensor nirkabel menggunakan modul wireless nRF2L01 yang berlokasi di Departement Informatika Institut Teknologi Sepuluh Nopember Surabaya.

1.6.5 Penyusunan Buku

Pada tahap ini dilakukan penyusunan buku sebagai dokumentasi dari pelaksanaan tugas akhir yang mencakup seluruh konsep, teori, implementasi, serta hasil yang telah dikerjakan.

1.7 Sistematika Penulisan Laporan

Sistematika penulisan laporan tugas akhir adalah sebagai berikut:

1. Bab I. Pendahuluan
   Bab ini berisi penjelasan mengenai latar belakang, rumusan masalah, batasan permasalahan, tujuan, manfaat, metodologi, dan sistematika penulisan dari pembuatan tugas akhir.
2. Bab II. Tinjauan Pustaka
   Bab ini berisi kajian teori atau penjelasan dari metode, algoritma, *library*, dan *tools* yang digunakan dalam penyusunan tugas akhir ini. Kajian teori yang dimaksud berisi tentang penjelasan singkat mengenai *Mobile Ad Hoc Network* (MANET), *Zone Routing Protocol* (ZRP), *Network Simulator* 2 (NS-2), dan AWK.
3. Bab III. Perancangan Perangkat Lunak dan Perangkat Keras
   Bab ini berisi pembahasan mengenai desain dari jaringan sensor nirkable yang akan dibuat, meliputi arsitektur dann proses perangkat lunak dan perangkat keras.
4. Bab IV. Implementasi
   Bab ini menjelaskan implementasi dari desain dari jaringan yang akan dilakukan pada tahan desain, meliputi potongan *pseudocode* yang terdapat dalam perangkat ludan dan perangkat keras yang digunakan.

5. Bab V. Pengujian dan Evaluasi
   Bab ini berisi hasil uji coba dan evaluasi dari implementasi jaringan sensor nirkabel yang dibuat dengan melihat keluaran yang dihasilkan, analisa dan eveluasi untuk mengatahui kemampuan jaringan dan penggunaan energy pada jaringan sensor nirkabel.
6. Bab VI. Kesimpulan dan Saran
   Bab ini merupakan bab yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan, masalah-masalah yang dialami pada proses pengerjaan tugas akhir, dan saran untuk pengembangan tugas akhir ke depannya.
7. Daftar Pustaka
   Bab ini berisi daftar pustaka yang dijadikan literatur dalam tugas akhir.
8. Lampiran
   Dalam lampiran terdapat kode sumber program secara keseluruhan.

(Halaman ini sengaja dikosongkan)

BAB II
TINJAUAN PUSTAKA

Bab ini berisi pembahasan mengenai teori-teori dasar atau penjelasan dari metode dan alat yang digunakan dalam tugas akhir. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap program yang dibuat dan berguna sebagai penunjang dalam pengembangan riset yang berkaitan.

2.1 *Wireless Communication*

Wireless Communication atau Komunikasi nirkabel merupakan komunikasi terjadi antara dua devices yang di sebut *transmitter dan receiver* tanpa menggunakan perantara kabel.

Transmiter pada komunikasi nirkabel memiliki fungsi sebagai pengirim pesan informasi sedangkan *receiver* berfungsi sebagai penerima pesan informasi. *Transmiter* dan *receiver* pada perangkat *wireless* umumnya sudah terintegrasi menjadi satu perangkat. Sehingga dalam satu perangkat tersebut bisa berperan sebagai *transmitter* dan *receiver*[1].

Komunikasi nirkabel memiliki beberapa kelebihan yaitu penghematan dari sisi perangkat keras (hardware). Hal ini disebabkan karena komunikasi nirkabel tidak memerlukan kabel ketika berkomunikasi sehingga dapat menghemat biaya komunikasi kabel. Selain itu komunikasi nirkabel juga memiliki skalabilitas yang tinggi. Setiap devices yang ingin berkomunikasi dengan device yang lain tidak memerlukan instalasi atau persiapan hardware yang banyak seperti kabel dan alokasi port.[1]

Walaupun komunikasi nirkabel mempunnyai kelebihan dari sisi hardware dan kemudahan akses, komunikasi nirkabel lemah terhadap intervensi dari gangguan external. Gangguan external bisa seperti radiasi atau juga transmisi dari alat nirkabel lainnya. Selain itu untuk melayani banyak client jaringan nirkabel tidak bisa menerima seluruh transmisi dalam 1 waktu, sehingga dibutuhkan sebuah penjadwalan transmisi agar tidak terjadi tabrakan transmisi.

2.2 *Wireless Sensor Network*

Wireless sensor network (WSN) merupakan sekumpulan dari beberapa device kecil yang dilengkapi dengan sensing yang terintegrasi dan kemampuan komunikasi nirkabel, yang diharapkan dapat digunakan secara luas dalam berbagai aplikasi. Sensor ini dioperasikan dengan daya baterai dan energinya tidak selalu diperbaharui karena masalah biaya, lingkungan dan bentuknya[2].

Energi dalam WSN nodes digunakan pada CPU, sensor, dan radio yang dimana merupakan mengomsumsi energi terbanyak. Untuk mengoptimalkan penggunaan energi, identifikasi source yang paling besar menghabiskan energy dalam komunikasi sangatlah penting, seperti *collision, overhearing, control packet overhead*, dan *idle listening*. Salah satu tantangan untuk mencapai teknologi potensial ini yaitu dengan manajemen konsumsi energi yang efektif dalam device ini untuk memaksimalkan lifespan sebuah node dan akhirnya lifespan jaringan pada saat yang sama juga cukup memelihara kualitas dan kuantitas service[2].

2.3 Clustering Jaringan Sensor Nirkabel

Jaringan sensor nirkabel biasanya terdiri dari banyak node bisa mencapai puluhan bahkan ratusan node. Node-node ini dapat memperluas jangkuan sink dalam melakukan sensing, karena node yang berada diluar jangkauan dapat membantu memberi data dengan melakukan multihop ke node lainnya sampai kepada sink[3].

Melakukan multihop atau tranmisi jauh secara masive tanpa adanya organisasi tertentu akan menguras energi sangat besar yang malah membuat *lifespan* dari pada jaringan sensor terlalu pendek karena pemakaian energi yang tidak efisien. Karena itu jaringan sensor nirkabel dilakukan *clustering* dimana membagi node menjadi beberapa kelompok kecil. Di dalam *cluster* terdapat sebuah node yang menjadi komando node lain yang ada di dalam *cluster* yang disebut *cluster head*. *Cluster head* berfungsi dalam mengatur *cluster* node dari penjadwalan transmisi maupun sampai menjadi hop ke sink. Namun ada kalanya *cluster head* kehabisan

daya atau kehilangan fungsi sehingga dapat mengancam keseluruhan sistem, maka sangat penting untuk melakukan rotasi *cluster head* agar terhindar dari mati nya jaringan sensor nirkabel[3].

2.4 *Cluster Head*

Dalam jaringan sensor nirkabel terdapat node yang nantinya akan menjadi *cluster head* yang bertugas untuk mengumpulkan data dari *node-node* yang tergabung dan kemudian mengirimkannya ke *node server / coordinator*. *Cluster head* akan dipilih dan diganti di setiap putaran untuk meratakan penggunaan energi.[4]

Cluster Head juga bertugas menjadi komando dalam clsunter node. *Cluster head* juga mengatur penjadwalan transmisi setiap node untuk menghindari tubrukan transmisi yang berujung packet lost. Karena fungsinya yang vital *cluster head* harus awas terhadap penurunan energi yang telah digunakan. Saat penurunan energi melewati batas *cluster head* akan memberi instruksi kepada *cluster node* untuk mengganti *cluster head* dengan yang baru dari node lain.

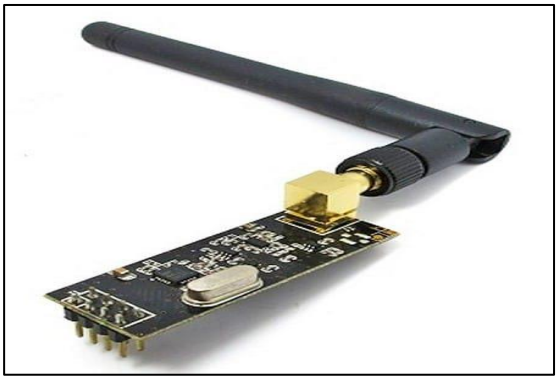
2.5 Modul nRF24L01

Modul Wireless nRF24L01 adalah modul komunikasi jarak jauh yang memanfaatkan pita gelombang RF (*Radio Frequency*) 2.4GHz ISM (Industrial, Scientific and Medical) yang didesain untuk jaringan nirkabel yang membutuhkan penggunaan daya sangat rendah. Modul ini berupa sebuah *chip transreceiver* tunggal yang memiliki *baseband logic Enhanced Shockburst*. Modul ini cocok digunakan dengan *Microcontroller* Arduino[4].

Modul nRF24l01 dapat bekerja sebagai *transmitter* dan juga sebagai *receiver*. Namun nRF tidak 9ias menjadi *transmitter* dan *receiver* dalam waktu yang bersamaan. Perlu adanya pergantian mode bila membutuhkan komunikasi dua arah antar node nRF. Selain itu melakukan pergantian mode dengan timing

yang pas diperlukan karena paket yang dating akan drop bila modul dalam mode transmisi saat bersamaan.

Diluar dari kelemahan tersebut medul nRF24L01 memiliki kelebihan dalam kehandalan. Portabilitas, dan pemakaian daya. Pemakaian daya yang rendah dan dimensi yang kecil membuat modul ini banyak dipakai dalam biadang yang memerlukan komunikasi nirkabel tidak terkecuali jaringan sensor nirkabel.



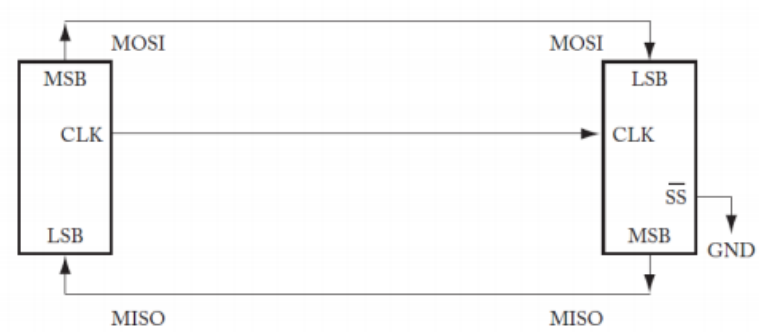
Gambar 2.1: Modul nRf24L01l

2.6 Serial Peripheral Interface (SPI)

Serial peripheral Interface merupakan merupakan komunikasi seri synchronous yang berarti harus menggunakan clock yang sama untuk mengsinkronisasi deteksi bit pada *receiver*. Biasanya hanya digunakan untuk komunikasi jarak pendek dengan mikrokontroler lain yang terletak pada papan rangkaian yang sama. Bus SPI dikembangkan untuk menyediakan komunikasi dengan kecepatan tinggi dengan menggunakan pin mikrokontroler yang sedikit.

SPI melibatkan *master* dan *slave*. Keduanya mengirimkan dan menerima data secara terus menurus, namun *master* bertanggung jawab untuk menyediakan sinyal *clock* untuk transfer data. Gambar 2.2 menunjukkan komunikasi antara *master* dan

slave pada komunikasi SPI. Master menyediakan *clock* dan data 8bit pada pin *master-out-slave-in* (MOSI) dimana data tersebut ditransfer satu bit per pulsa *clock* menuju pin MOSI pada *slave*. Delapan bit data juga diberikan dari slave ke master melalui pin *master-in-slave-out* menuju pin MISO pada master. Biasanya pin *SS* (slave select) diberi ground (active low) untuk menjadikannya sebagai slave.

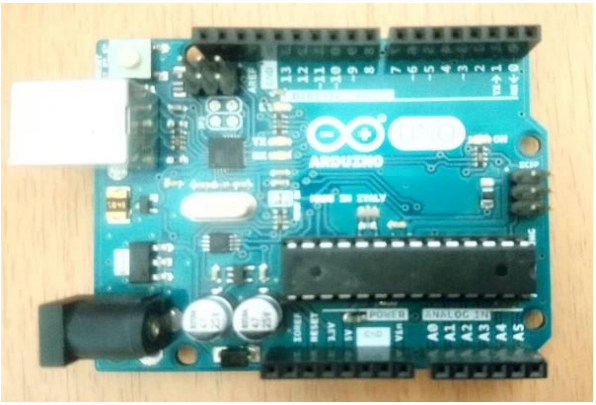


Gambar 2.2 Skema Komunikasi SPI

2.7 Arduino

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

Board arduino menggunakan banyak jenis mikroprosesor dan mikrokontroler. Board Arduino dilengkapi dengan set pin *input/output* (I/O) digital dan analog yang bisa digunakan untuk berkomunikasi dengan board lainnya atau modul expansi yang terdapat di pasaran. Board Arduino mempunyai antarmuka komunikasi serial, termasuk *Universal Serial Bus* (USB) yang digunakan untuk memprogram mikrokontroler Arduino melalui komputer. Mikrokontroler Arduino secara kusus diprogram menggunakan dialek dan fitur bahasa C dan C++. Tidak seperti board atau mikrokontroler pada umumnya yang menggunakan compiler toolchains tradisional untuk kompilasi program, Arduino memiliki *Integrated Development Environment* (IDE) sendiri dalam kompilasi maupun penulisa progrma pada board[5].



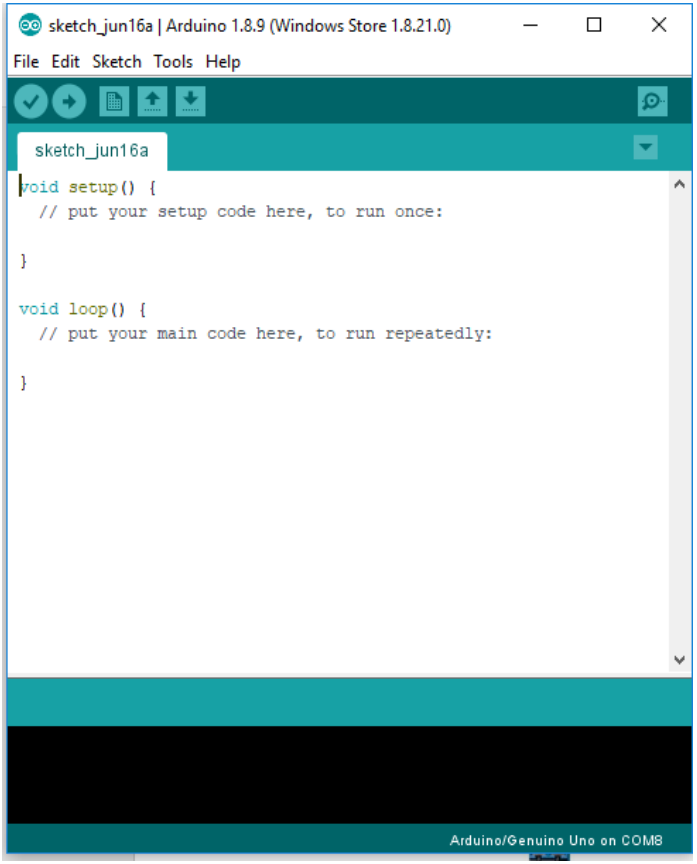
Gambar 2.3: Arduino Uno R3

2.8 Arduino IDE

Arduino IDE merupakan sebuah perangkat lunak yang digunakan sebagai tempat untuk menulis logika-logika dari suatu skema rangkaian yang terhubung dengan board Arduino. Arduino IDE dibangun dengan bahasa pemrograman Java dan bersifat cross-

platform. Barisan kode dalam Arduino IDE ditulis mengikuti aturan dari C/C++ dan baris kode ini disebut dengan istilah sketch.

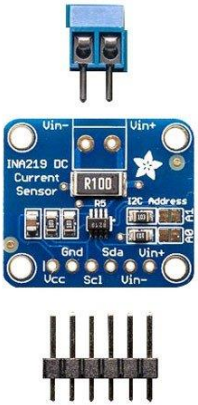
Arduino IDE dipakai karena memiliki kompatibilitas baik dengan semua perangkat Arduino. Arduino IDE mempunyai fitur deteksi otomatis bila ada Arduino yang dihubungkan ke komputer. Untuk melakukan debugging Arduino IDE memiliki fitur serial monitor untuk komunikasi dengan Arduino.



Gambar 2.4 Arduino IDE

2.9 Sensor INA219

Sensor INA219 adalah sensor yang di gunakan untuk mengukur tegangan dan arus DC pada arduino, Modul sensor ini merupakan modul yang didukung dengan kemampuan ukur yang mampu mengukur sumber beban yang sampai 26 Vdc dan arus 3,2 Ampere, modul ini juga dapat mengukur tegangan lewat komunikasi 12C dengan tingkat presisi 1%[6].



Gambar 2.5 Module Sensor INA219

2.10 PlatformIO

PlatformIO adalah toolchain yang digunakan untuk kompilasi program mikrontroler. PlatformIO bersifat *open-source* dan mendukung banyak platform sehingga dapat dijalankan dimanapun. PlatformIO sudah dilengkapi toolchains, debugger, framework sehingga memudahkan developer dalam mengembangkan embedded device.

PlatformIO bisa digunakan melalui command line interface. PlatformIO mendukung lebih dari 200 jenis mikrokontroler yang terhubung dengan repositori yang membuat toolchain mudah untuk

didapatkan dan up to date. Selain itu PlatformIO mendukung multi upload dimana kita bisa upload program ke banyak board sekaligus sehingga sangat cocok digunakan untuk memprogram device jaringan sensor nirkabel.

(***Halaman ini sengaja di kosongkan***)

BAB III
PERANCANGAN

Perancangan merupakan bagian penting dalam pembuatan perangkat lunak dan perangkat keras yang berupa perencanaan secara teknis dari sistem jaringan yang dibuat. Pada Bab ini akan dibahas mengenai perancangan dan implementasi rancang bangun *cluster-based* protocol untuk pengiriman data secara adaptif dengan pengaturan kekuatam transmisi dan *monitoring* ketersedian energi pada jaringan sensor nirkabel yang dibangun menggunakan Arduino sebagai nodenya, Menggunakan modul Wireless nRF24L01 untuk berkomunikasi antar node dan menggunakan sensor INA219 untuk melakukan *monitoring* terhadap energi.

3.1 Deskripsi Umum

Pada tugas akhir ini akan dibuat sebuah implementasi *wireless sensor network* menggunakan mikrokontroler Arduino. Setiap node masing-masing berisi modul nRF24L01 sebagai media *transmisi* dan modul INA219 sebagai media untuk melakukan monitoring energi pada setiap node.

Pada *wireless sensor network* yang akan dibuat terdiri dari 6 node. Satu node akan berperan manjadi *server/coordinator* yang berfungsi sebagai penerima data dari *node cluster,* sementara 5 node lainnya akan menjadi *cluster node*, Node *coordinator* akan terhubung dengan computer dan mendapat daya tetap dari computer, sedangkan *node cluster* menggunakan batrai sebagai penyuplai daya yang digunakan.

Dari ke 5 buah *cluster node* akan di pilih satu buah node secara bergantian untuk menjadi *cluster head* yang berfungsi untuk mengumpulkan data dari *node cluster* kemudian mengirimkan data langsung ke server. *Cluster head* dipilih secara bergantian setiap beberapa menit menggunakan parameter sisa energi yang tersisa pada batrai setiap node untuk menentukan siapa yang layak untuk menjadi *cluster head.*

3.2 Daftar Istilah

Daftar istilah yang sering digunakan pada buku tugas akhir ini dapat dilihat pada Tabel 3.1.

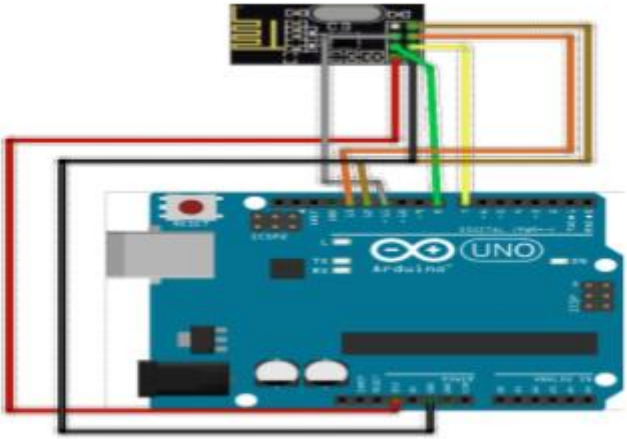
Table 3.1 Daftar Istilah

No.	Istilah	Penjelasan
1	*Node*	Sebuah titik redistribusi atau titik akhir dari suatu komunikasi.
2	*Cluster Node*	*Cluster node* merupakan kumpulan node-node yang terdapat dalam jaringan sensor nirkabel
3	*Cluster head*	*Cluster head* merupakan node yang menjadi pemimpin pada sebuah *cluster node* yang berfungsi sebagai pengirim packet dari *cluster node* ke *coordinator*.
4	*Coordinator*	*Coordinator* merupakan node yang berfungsi untuk mengumpulkan atau menerima packet pesan
5	*Packet lost*	*Packet lost* merupakan pesan yang hilang ketika transmisi data
6	*Packet Delivery Ratio* (PDR)	PDR merupakan teknik penghitungan perbandingan jumlah paket yang diterima oleh *node* tujuan dengan jumlah paket yang dikirim oleh *node* sumber.

3.3 Arsitektur Sistem

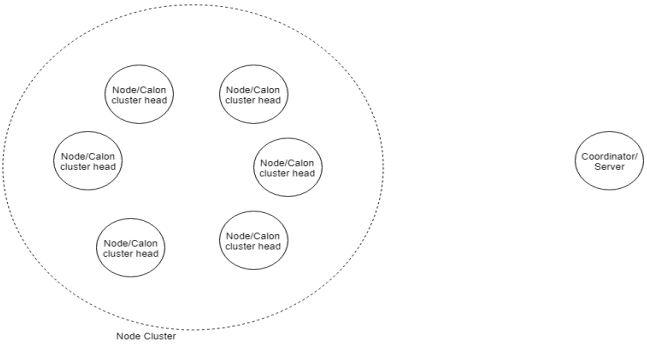
Pada sub bab ini akan di jelaskan mengenai arsitektur umum jaringan sensor nirkabel yang akan dibangun. Sistem ini terdiri dari 6 node yang terdiri dari 1 node sebagai *coordinator/server* dan 5 node lain sebagai *cluster* node, setiap node menggunakan *mikrocontroler* Arduino, modul nRF24L01 yang menggunakan SPI sebagai interface komunikasi dengan Arduino dan menggunakan modul INA219 sebagai Sensor untuk melakukan

monitoring terhadap penggunaan batrai yang ditunjukan pada gambar 3.1



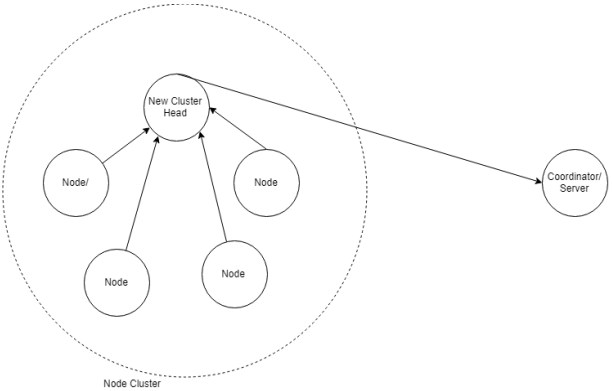
Gambar 3.1 Arduino dan nRF24L01

Setiap *cluster node* akan diimplementasikan pemilihan *cluster head* sehingga setiap node dapat bergantian menjadi *cluster head*. Selanjutnya *cluster node* dan *coordinator* ditempatkan terpisah dengan jarak minimun 50 meter. Setelah *cluster* hidup setiap *node cluster* akan menyamakan timer pada jaringan sensor nirkbel dan kemudian menentukan siapa yang layak untuk menjadi *cluster head*.

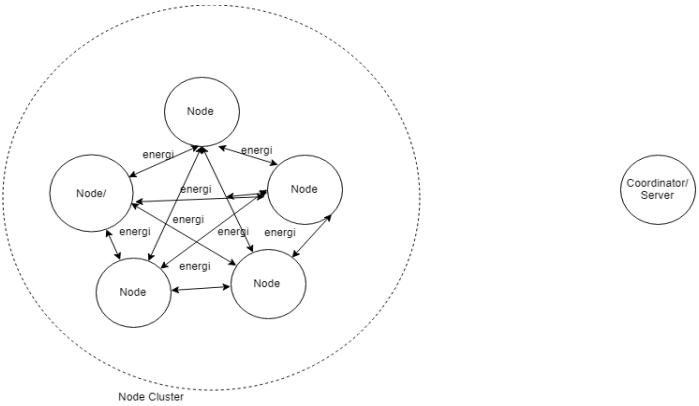


Gambar 3.2 Kondisi Awal *Node Cluster*

Pada Kondisi awal node 1 akan berperan untuk melakukan reset timer agar timer pada setiap *node* sama dengan *node* lain pada *cluster node*. Kemudian node 1 menjadi *cluster head* untuk yang pertama kali. Pada jaringan sensor nirkabel yang dibuat *cluster head* akan mengirimkan CH data yang berisikan address *cluster head* agar setiap node mengetahui alamat *cluster head* yang terpilih. Selanjutnya node akan mengirimkan data sensing mereka ke *cluster head* dan untuk node yan tidak menjadi *cluster head* kekuatan transmisi diturunkan menjadi minimal. *Cluster head* kemudian berperan untuk melanjutkan pengiriman data yang di terima dari masing masing node untuk diteruskan kepada *coordinator/server,* untuk *cluster head* kekuatan transmisi di ubah menjadi maksimal agar mampu mengirimkan data yang diterima ke *coordinator/server.* Selanjutnya *cluster head* akan dipilih berdasarkan energi yang tersisa pada batrai.



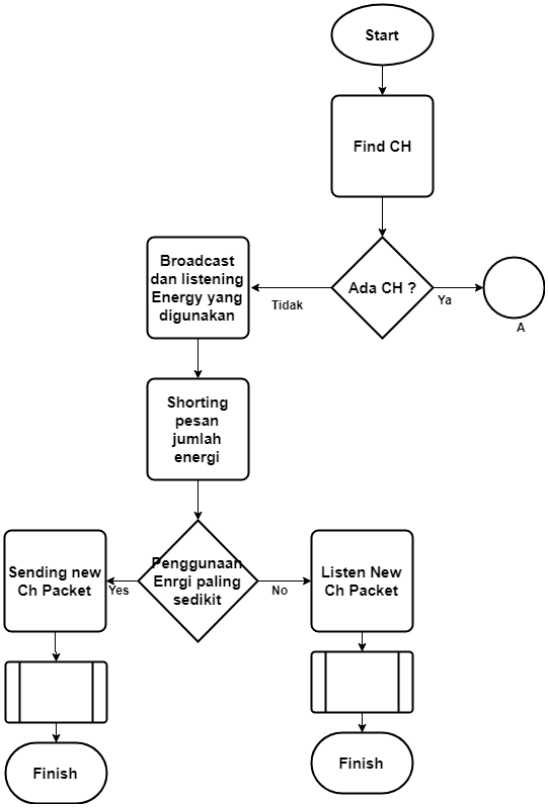
Gambar 3.3 Kondisi *cluster node* ketika cluster head sudah di tentukan



Text

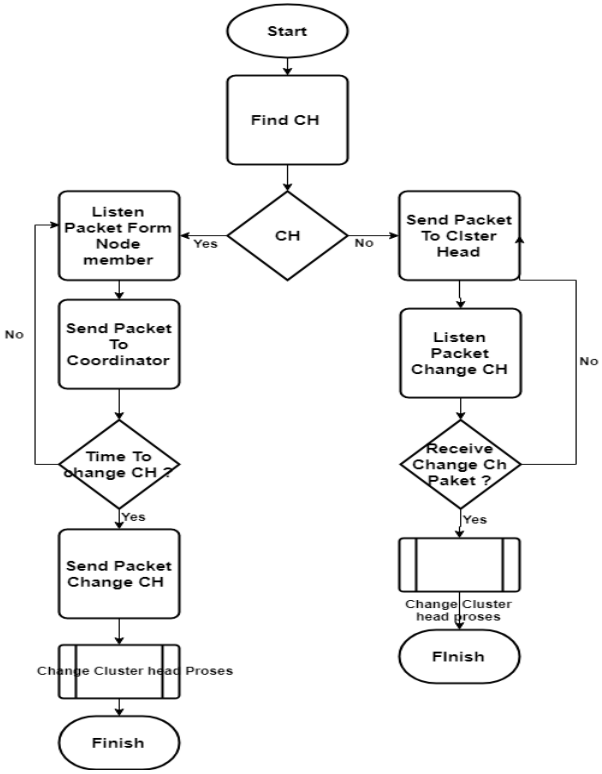
Gambar 3.4 Kondisi cluster node ketika broadceast sisa energi pada batrai untuk penentuan cluster head

Cluseter node pada saat tidak melakukan *transmisi* akan *listening* kemungkinan adanya pesan untuk pergantian *cluster head* yang di kirimkan oleh *cluster head.* Pesan pergantian *cluster head* menginstruksikan *cluster node* untuk berhenti melakukan *tranmsi* kepada *cluster head* yang sebelumnya, kemudian melakukan *broadcast* sisa energi yang terdapat pada batrai ke semua *cluster node* dan kemudian menentukan siapa yang layak untuk menjadi *cluster head*. Seperti yang di gambarkan pada gambar 3.4



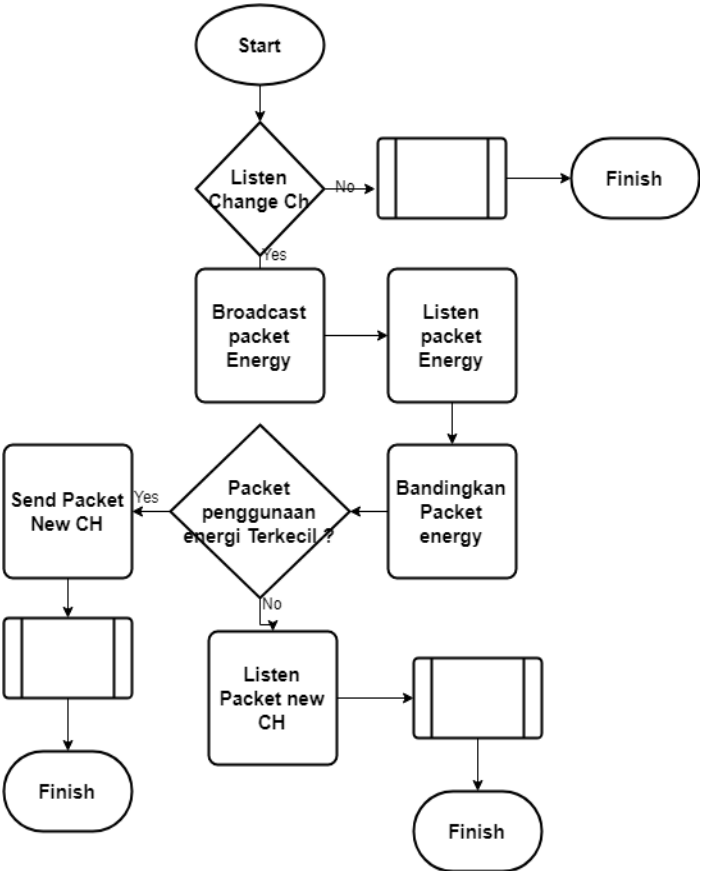
Gambar 3.5 *Flowchart* Jaringan sensor nirkabel pembentukan cluster node

Pada gambar 3.5 merupakan *flowchart* awal pembentukan *cluster node* dan *cluster node* belum memiliki cluster head sehingga di awal di tentukan dulu sebuah yang akan menjadi *cluster head* dengan cara setiap node dalam *cluster node* melakukan *broadcast* pesan jumlah energi yang digunakan masing-masing node, setelah itu masing-masing node akan melakukan shorting. Node yang menggunakan energi paling sedikit akan menjadi *cluster head* yang kemudian akan mengirimkan pesan kalau dia adalah *cluster head*.



Gambar 3.6 *Flowchart* pengiriman pesan dari *node cluster* ke *server*

Pada gambar 3.6 kondisi *cluster node* sudah memiliki *cluster head*. Node yang tidak menjadi *cluster head* akan mengirimkan packet pesan kepada *cluster head* yang kemudian pesan tersebut akan diteruskan oleh *cluster head* ke *server*. *Cluster head* juga akan mengirimkan pesan *stop* untuk melakukan pergantian *cluster head* sesuai dengan waktu yang telah di tentukan.



Gambar 3.7 *Flowchart* pergantian *cluster head*

Pada gambar 3.7 setelah *cluster head* mengirimkan pesan *stop, node* yang menjadi *cluster head* akan berhenti menjadi *cluster head* dan *node member* juga berhenti mengirimkan pesan kepada *cluster head*. Setelah itu semua node pada *cluster node* akan melakukan *broadcast* jumlah energi yang digunakan kemudian di shorting dan *node* yang paling sedikit menggunakan energi akan menjadi *cluster head* yang baru. Bisa juga dilihat pada Pseudocode pergantian *cluster head* di bawah.

```
1.  Start
2.  Arduino Timer start
3.  if(reset time =0)
4.       if(nodeId=1)
5.          Broadcast  reset_time  and  set  node  1  as
    cluste_head
6.       else
7.          Listen  reset_time  message  set  node  1  as
    cluste_head
8.  else
9.     if(cluster_headID > 0)
10.  if(cluster_head)
11. then
    Listening  message  from  node_cluster  and  send  to
    coordinator and send change_ch
12.  else
              Send data to cluster_head
       else
           Broadcast  current_energy  to  select  new
    cluster head
```

Gambar 3.8 Pseudocode Penentuan cluster head

(***Halaman ini sengaja di kosongkan***)

BAB IV
IMPLEMENTASI

Bab ini membahas mengenai implementasi sistem yang dilakukan berdasarkan rancangan yang telah dijabarkan pada bab sebelumnya. Implementasi berupa kode sumber untuk membangun program,spesifikasi hardwere. Implementasi yang dijelaskan dibagi menjadi lingkungan Implementasi perangkat keras dan lingkungan implementasi perangkat lunat pembangunan sistem.

4.1 Lingkungan Implementasi

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

4.1.1 Lingkungan Implementasi Perangkat Keras

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

Table 4.1 Table Lingkungan Implementasi Perangkat Keras

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

	• 2KB
Perangkat Wireless Transceiver	**Model:** • nRF24L01+SingleChip2.4GHz Transceiver **Manufaktur:** • Nordic Semiconductor **Tegangan:** • 1.9-3.6V **Frekuensi:** • 2.4GHzISMBand **Interface:** • SPI **Ukuran:** • 20-pin4x4QFNPackage
Perangkat Monitoring Energy	**Model:** • INA219 **Manufaktur:** • Adafruit **Maximun Ratings:** • VIN 0-26 V • Vcc 3.0 – 5.5 V • I_{max} 3.2 A **Accuracy:** • 2%(measured) **Ukuran:** • 25 x 22mm (1.0 x 0.87″)

4.1.2 Lingkungan Implementasi Perangkat Lunak

Pada bagian ini akan dibahas mengenai perangkat lunak apa saja yang dibutuhkan untuk membangun sitem.

Lingkungan implementasi perangkat lunak dari system yang akan dibangun secara lebih lengkap di jelaskan pada table 4.2 di bawah ini.

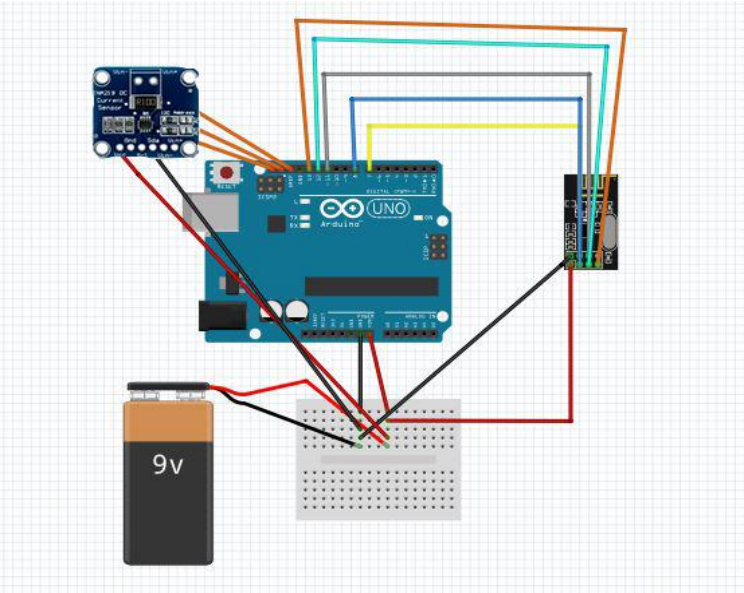
Table 4.2 Table Lingkungan Implemantasi Perangkat Lunak

Perangkat Lunak	Detail
Arduino IDE	Arduino IDE adalah sebuah IDE yang di gunakan untuk melakukan kompilasi terhadapa kode program Bahasa C untuk mikrokontroler Arduino serta digunakan juga untuk mengupload program ke dalam Arduino. Pada Arduino IDE juga bias melihat serial monitoring yang berfungsi untuk melihat hasil dari program yang berjalan pada arduino
Visual Studio Code	Visual studio code merupakan text editor yang digunkan untuk melakukan pemograman dan support terhadap IDE PlatformIO
PlatformIO	PlatformIO berisi toolchain yang digunakan untuk melakukan kompilasi kode program bahasa C untuk mikrokontroler Arduino dan mikrokontroler lainnya.
nRF24 Library	Library yang digunakan agar Arduino dapat berkomunikasi dengan modul nRF24L01.

	Library ini sudah mencakup fungsi-fungsi yang dibutuhkan seperti pengiriman data, pembacaan transmisi, ACK payloads, dan pengaturan kekuatan transmisi.
Adafruit INA219 Library	Library yang digunakan agar Arduino dapat berkomunikasi dengan modul INA219. Library ini sudah mencakup fungsi fungsi yang di butuhkan seperti mendapatkan Bus Voltage, Shunt Voltage dan Arus yang mengalir.

4.2 Implementasi *Cluster Node*

Pada bagain ini dijelaskan implementasi dari setiap *node cluster* yang digunakan. Setiap *cluster node* Arduino dihubungkan dengan modul wireless nRF24L01 melalui interface SPI. Baterai 9V digunakan untuk mentenagai node, Dan sensor INA219 digunakan untuk melakukan monitoring terhadap penggunaan energi. Untuk lebih jelas dapat melihat Gambar 4.1



Gambar 4.1 Rangkaian *Cluster Node*

Kutuk positif beterai dihubungakn ke pin *Vin* Arduino dan kutub negative ke pin *GND* untuk menyuplai daya. Kutub positif baterai di sambungkan ke pin *Vin+* sensor INA219 untuk mendapatkan energi yang digunakan. Untuk pin 5v di hubungan dengan pin *VCC* pada modul nRF24L01 dan pin *VCC* modul INA219.

4.3 Implementasi Code Program Pemilihan *Cluster Head*

Pada bab ini akan di implementasikan code program Pemilihan *Cluster Head* ke masing-masing node yang sudah di rangkai seperti Gambar 4.1. Dalam implementasi code program akan di bagi dalam beberapa bagian dan selanjutnya akan dijelaskan pada subbab-subbab tersendiri.

4.3.1 Global Variabel dan Type

Subbab ini membahas variable dan type data yang digunakan oleh *coordinator* untuk menunjang fungsi-fungsi lain dalam program.

```
Radio ← new RF24(7,8)

nodeId ← <node id different in every node>

self_addr ← <node nrf address different in every node>

sinkAddr ← 1000
bcAddr ← 2010

CH_ID ← 0; <cluster head id change if cluster head change>
CH_Addr ← <cluster head addres change if cluster head
change>

flag_ch ← false;
flag_reset_time ← false;
bc_status ← false;

energi ← <different in every node>
```

Kode Sumber 4.1 Global dan type variable cluster node

Pada kode sumber 4.1 terdapat deklarasi beberapa addres yang digunakan dalam transmisi. Seperti self_addres yang merupakan alamat dari masing masing node, broadcast addres yang digunakan semua node untuk broadcast pesan, serta sink_addres untuk pengiriman ke *coordinator* oleh *cluster head*. Terdapat juga *Cluster head addres* yang menjadi alamat *cluster head* yang akan berganti dengan self addres sesuai dengan node berapa yang sedang menjadi *cluster head*. Program menggunakan reset time untuk melakukan reset agar waktu setiap node sama. Juga terdapat variable energy yang merupakan energi yang digunakan yang di dapatkan dari sensor INA219.

4.3.2 Fungsi Setup

Fungsi ini adalah fungsi yang akan selalu dijalankan minimal satu kali setiap Arduino dinyalakan. Fungsi ini berguna untuk setup variabel atau apapun sebelum arduino menjalankan fungsi loop yang akan selalu dijalankan selama Arduino hidup.

```
Function setup ():
        call serial. begin ()
        call radio. begin ()
        call INA219. begin ()
```

Kode Sumber 4.2 Fungsi Setup

Saat Arduino menyala Arduino akan menjalankan fungsi setup maka akan mengaktifkan radio nRF dan mengaktifkan modul ina219.

4.3.3 Fungsi Transmisi

Fungsi transmisi adalah fungsi yang berisi instruksi untuk nRF agar melakukan transmisi data ke alamat yang ditujukan.

```
Function radio_send (addr , msg) :

        call radio. stopListening ()
        call radio. openWritingPipe (addr)

        call radio. write(msg, call sizeof (msg))
```

Kode Sumber 4.3 Fungsi Transmisi

Pada fungsi transmisi, sebelum melakukan pengiriman nRF diinstruksikan untuk berhenti melakukan *listening* dikarenakan nRF tidak bias melakukan *listening* dan *transmisi* diwaktu yag sama. Kemudian nRF diberikan alamat node yang akan dikirimkan pesan, dan fungsi *write* menginstruksikan nRF untuk menulis pesan dan melakukan *transmisi*.

4.3.4 Fungsi *Listening*

Fungsi *listening* adalah fungsi yang berisi instruksi untuk nRF agar menerima pesan yang telah dikirimkan oleh node lain.

```
function radio_listening (addr , msg) :
       call radio. openReadingPipe (0 , addr)      call
radio. startListening ()

       if radio. available :
       call radio. read (msg, call sizeof (msg))

       return true
return false
```

Kode Sumber 4.4 Fungsi *Listening*

Pada Kode Sumber 4.3 fungsi *listening*, Sebelum mendengarkan pesan transmisi dari node lain, sebuah *pipe* akan dibuka untuk masuknya transmisi dilanjutkan dengan memanggil *startListening* untuk memulai mendengarkan transmisi dari node lain. Selanjutnya program akan melakukan pengechekan apakah ada transmisi yang masuk, fungsi akan mengembalikan nilai *True* apabila dia menerima transmisi, kemudian akan dibaca data yang di terimanya, sebaliknya jika tidak menerima transmisi fungsi akan mengembalikan nilai *false*.

4.3.5 Fungsi Get Energi

Fungsi get energi adalah fungsi yang digunakan untuk mendapatkan energi yang digunakan oleh Arduino yang di ukur oleh sensor INA219.

```
Function getpower()

        sensor219.getBusVoltage_V()
        sensor219.getCurrent_mA();
        power = busVoltage * current/1000
```

Kode Sumber 4.5 Fungsi Get Power

4.3.6 Fungsi CountTime

Fungsi CountTime adalah fungsi yang digunakan untuk membuat realtime timer di Arduino, fungsi ini memanfaatkan perhitungan milis yang bias dilakukan Arduino.

```
Function countTime()
        ++xMili;
        if(xMili ==999)
                Then ++second;
                        xMili=0;
        if(second>59)
                then ++minute
                        second=0;
        if (minute>59)
                then ++hour
                        minute=0;
```

Kode Sumber 4.6 Fungsi Count Time

4.3.7 Fungsi Reset Time

Fungsi ini adalah fungsi yang akan dijalankan di awal, untuk menyamakan semua time pada *cluste node*, fungsi ini membuat setiap node yang menerima pesan untuk perintah *reset time*, timernya menjadi kembali 0 lagi.

```
Function ()

if(flag_reset_time==false)
   then
     if (nodeId==1)
       then
          Transmit(reset_time_mssg)
                return true
     else
                Listening(reset_time_mssg)
                return True
return false
```

Kode Sumber 4.7 Fungsi Reset Time

4.3.8 Fungsi Change *Cluster Head*

Pada fungsi ini akan dibahas fungsi dalam pemilihan *cluster head* pada *cluster node* tidak memiliki *cluster head*.

```
Function ()
        if (clusterHead)
                Transmit(stopCH)
                Then chstatus false;
        else
                Listen(stopCH)

                then
                        Broadcast(energi)

                Short_energi,
                        Energi_min= newClusterhead
                newClusterhead transmit(I,m CH)
```

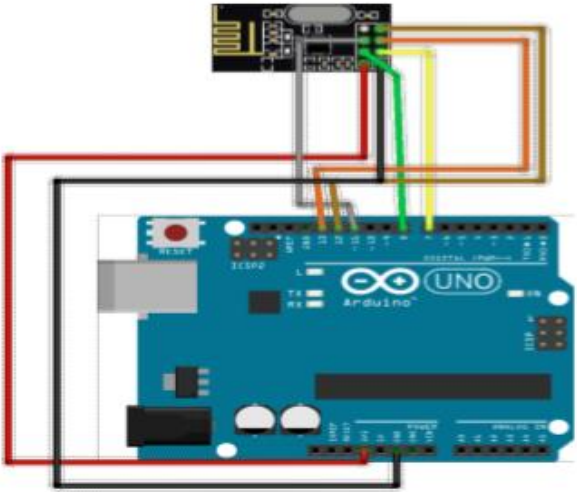
Kode Sumber 4.8 Fungsi Change Cluster Head

Fungsi ini node yang sebelumnya menjadi *cluster head* akan mengirimkan pesan ke *cluster node* untuk berhenti mengrimkan data dan pergantian *cluste head*. Setelah pesan itu diterima *cluster head* menjadi tidak ada, dan semua node akan malakukan *broadcast* jumlah energi yang digunakan masing-masing ke *cluster node*, kemudian masing-masing node akan melakukan shorting energi tersebut, node yang menggunakan energi paling sedikit akan menjadi *cluster head* dan addres CH di rubah menjadi alamat dari node tersebut.

4.4 Implementasi *Coordinator*

Pada bagian ini akan di jelaskan implementasi simulasi dari *coordinator*. Seperti yag di jelaskan sebelumnya *coordinator* berperan sebagai penerima data dari *cluster node* sehingga dia tidak melakukan transmisi data. *Coordinator* hanya melakukan *listening* dari *cluster node*. Rangkaian yang di gunakan oleh *coordinator* sama seperti rangkaian pada *cluster node* dengan menggunakan *Mikrokontroller* Arduino dan dihubungkan dengan

modul *wireless* nRF24L01 melalui interface SPI, namun tidak menggunakan baterai dan modul INA219 dikarenakan sumber yang digunakan langsung menggunakan USB dan tidak diperlukan untuk mengukur sisa energi. Rangkaian akan terlihat seperti pada gambar 4.2



Gambar 4.2 Rangkaian Coordinator

4.5 Implementasi Code Program *Coordinator*

Pada bab ini akan di jelaskan implementasi code program yang di gunakan pada *coordinator*. Dalam implementasi code pada *coordinator* fungsi yang di gunakan lebih sederhana, dikarenakan pada *coordinator* hanya perlu melakukan *listening* packet pesan yang akan dikirimkan oleh *cluseter node* dan kemudian menampilkannya pada serial monitor, code dibagi dalam 2 bagian

yaitu bagian setup dan bagian loop. Setiap implementasi code program akan dibahas dalam subbab berikut.

4.5.1 Global Variabel dan Type

Subbab ini membahas variable dan type data yang digunakan oleh *coordinator* untuk menunjang fungsi-fungsi lain dalam program.

```
 sink_addr  <-- 1000

radio <- new RF24 (7,8)
```

Kode Sumber 4.9 Global Variable Coordinator

Pada kode sumber diatas sink_addr adalah alamat yang akan di listen oleh coordinator untuk menerima pesan dari *cluster node*, sementara untuk variable radio digunakan untuk menginisialisasikan modul *wireless* nRF24L01 berada pada pin 7 dan 8 pada *mikrokontroller* Arduino.

4.5.2 Fungsi Setup

Fungsi ini adalah fungsi yang pasti akan selalu dilanja minimal satu kali setiap Arduino dinyalakan. Fungsi ini berguna untuk setup variable atau apapun sebelum Arduino menjalankan fungsi loop yang akan selalu berjalan selama Arduino hidup.

```
function setup ():

 call radio. begin ()

 call radio. openReadingPipe (0, sink_addr)
 call radio. startListening ()
```

Kode Sumber 4.10 Fungsi Setup Pada Coordinator

Fungsi ini hanya bertujuan untuk inisialisasi variable dan nRF. Karena *coordinator* hanya menunggu data dari *cluster node*

maka pada code pada *coordinator* di set untuk melakukan *listening* ke alamat *sink addres* yang telah di set pada global variable sejak *coordinator* dihidupkan.

4.5.3 Fungsi Loop

Fungsi ini adalah fungsi yang akan selalu dijalankan berulang-ulang selama Arduino beroperasi. Fungsi loop merupakan fungsi utuma pada Arduino. Semua perintah yang akan dilaksanakan di instruksikan dalam fungsi ini.

```
function loop () :

       data <- new dataSt
              if radio . available :
              call radio . read (data , sizeof ( data )
)
              print data . message
              print data . id_packet
```

Kode Sumber 4.11 Fungsi Loop pada coordinator

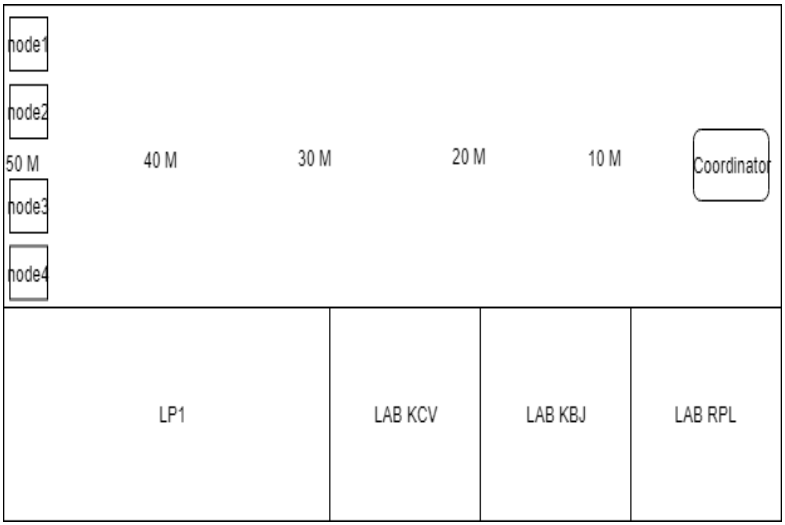
Fungsi ini bertujuan agar *coordinator* menunggu pesan dari *cluster node* dan kemudian mencetak pesan tersebut apabila dia menerima pesan dari *cluster node. Coordinator* akan melakukan hal tersebut selama Arduino masih beroperasi.

BAB V
UJI COBA DAN EVALUASI

Bab ini berisi penjelasan mengenai scenario uji coba yang dilakukan dan evaluasi terhadap pemilihan *cluster head* secara dinamis dan pemilihan daya transmisi secara dinamis serta monitoring terhadap penggunaan energi. Hasil uji coba didapatkan dari implementasi yang dijelaskan pada Bab 4 dengan scenario yang berbeda. Bab ini berisikan pembahsan mengenai lingkungan pengujian data pengujian, uji kinerja dan hasil pengujian yang digunakan untuk Bab selanjutnya.

5.1 Lingkungan Pengujian

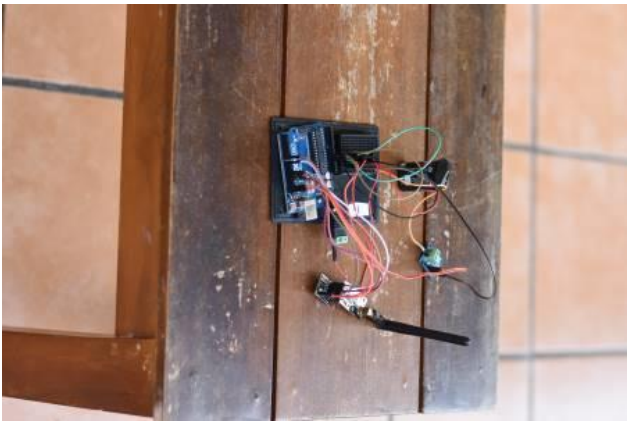
Lingkungan pengujian *system* pengiriman data secara adaptif dengan pengaturan kekuatan *transmisi* dan *monitoring* ketersedian energi akan dibuat 2 skenario uji coba yaitu pengiriman data yang menggunakan pemilihan *cluster head* dan satu lagi pengiriman tanpa menggunakan *cluster head*. Skenario *Cluster node* yang yang tidak menggunakan pemilihan *cluster head* semua *node* akan langsung mengirimkan pesan secara langsung kepada *coordinator* tanpa melalui sebuah *cluster head*, Sehingga semua *node* pada *cluster node* kekuatan *transmisi* nya di *setting* menjadi maksimal agar mampu mengirimkan data secara langsung kepada *coordinator*.Skenario *cluster node* yang menggunakan pemilihan *cluster head* menggunakan metode seperti yang sudah dibahas pada bab III dalam tugas akhir ini, pengiriman ke *coordinator* diwakilkan oleh sebuah *cluster head* .



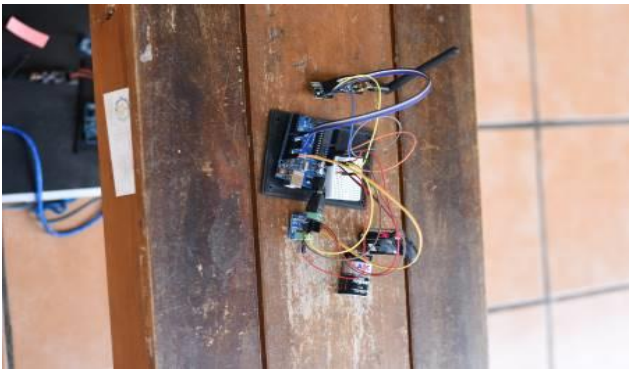
Gambar 5.1 Denah Uji Coba

Gambar 5.2 Lokasi Uji Coba

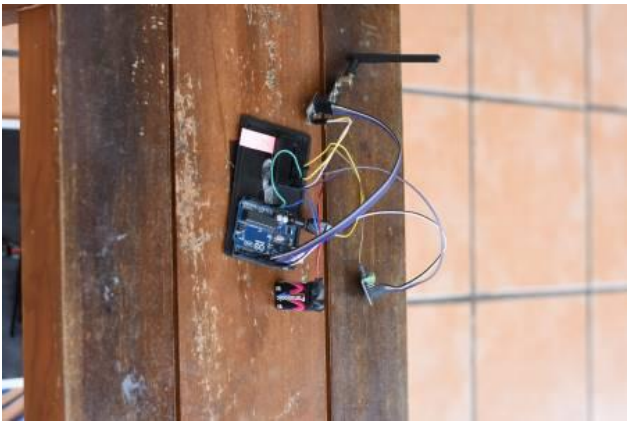
Setiap pengujian scenario dilakukan di lapangan basket di Departemen Informatika ITS, jarak antar *cluster node* dengan *coordinator* kurang lebih 50 meter. Dalam setiap skenario uji coba akan menggunakan 2,3,4 dan 5 *node*.Dan masing-masing *node* di beri jarak secara acak dengan jarak 1 sampai dengan 3 meter.



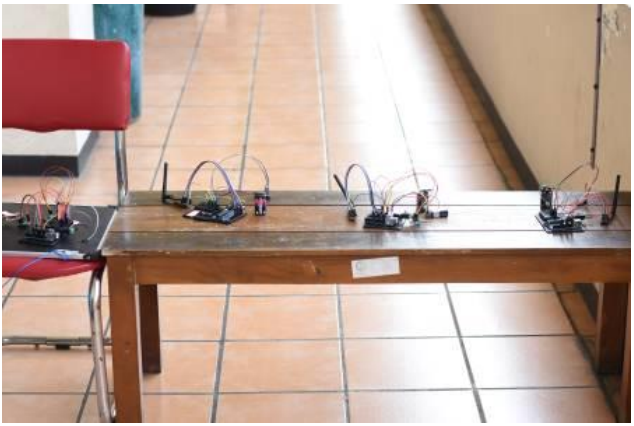
Gambar 5.3 Lingkungan pengujian node 1



Gambar 5.4 Lingkunga pengujian node 2



Gambar 5.5 Lingkungan pengujian node 3



Gambar 5.6 Linkungan pengujian cluster node

Gambar 5.7 Lingkungan pengujian *coordinator/server*

Dalam lingkunga uji coba *cluster node* seluruh *node* akan memakai beterai yang sama yaitu baterai Panasonic Rechargeable berbentuk kotak dengan voltase 9V dan kapasitas 220mAh. Baterai ini dipilih karena mampu mentenagai Arduino beserta modul nRF24L01 cukup lama dan bias di charger ulang sehingga dapat digunakan berulang. Baterai ini tidak memiliki resistor pullup di dalamnya sehingga penurunan kapasitas baterai menyebabkan penurunan cukup signifikan dari voltase baterai. Hal ini dapat menjadi acuan dalam menghitung performa *cluster node* dalam penggunaan daya baterai.

Gambar 5.8 Baterai Panasonic *Rechargeable 9V*

Untuk alat pengukur voltase beterai penulis menggunakan multimeter digital. Multimeter digital dipilih karena mudah dalam penggunaan dan pembacaan hasil agar perhitungan lebih akurat.

Gambar 5.9 Multimeter Digital

5.2 Skenario Uji Coba

Sebelum melakukan uji coba, perlu ditentukan scenario yang akan digunakan dalam proses uji coba. Dengan scenario yang dibuat kita akan menguji apakah perangkat yang di buat sudah berjalan sesuai dengan yang di rancang dan benar, dan menentukan performa pada masing-masing scenario. Pada uji coba ini kita akan menguji scenario manakah yang memiliki hasi lebih baik, mulai dari efisiensi dalam penggunaan energi dan jumlah *packet lost* yang terjadi. Terdapat 2 macam scenario uji coba sebagai berikut:

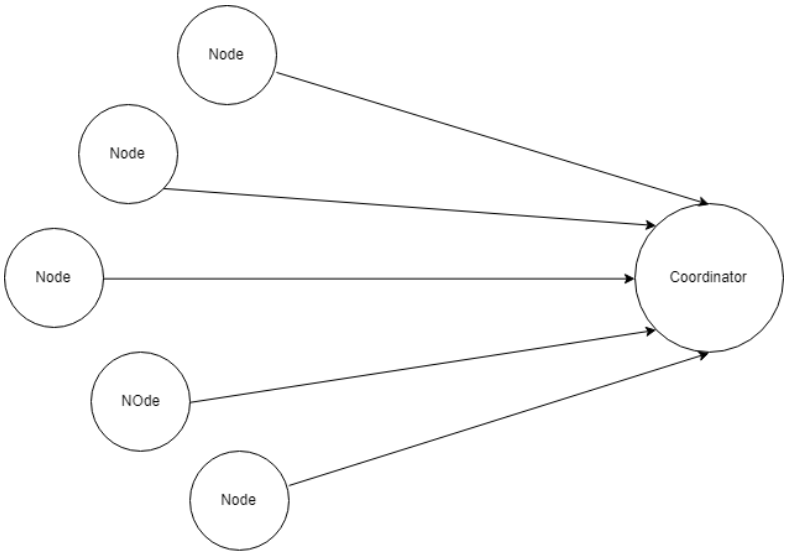
1. Pengujian performa *cluster node* saat transmisi data langsung ke *coordinator* tidak menggunakan pemilihan *cluster head* dan monitoring terhadap penggunaan daya. Dilihat dari *packet delivery ratio* dan penurunan daya pada baterai di semua *node*.
2. Pengujian performa *cluster node* saat transmisi data ke *coordinator* menggunakan pemilihan *cluster head* dengan menggunakan parameter energi yang digunakan pada setiap *node*. Dilihat dari dari *packet delivery ratio* dan penurunan daya pada baterai di semua *node.*

Dalam melakukan uji coba antara *cluster* dan *coordinator* akan diberikan jarak bervariasi pada tiap test, namun variasi jarak akan tetap dibuat cukup jauh agar nRF tetap memerlukan daya transmisi tinggi untuk menjangkaunya, untuk jarak terjauh adalah kurang lebih 50meter.

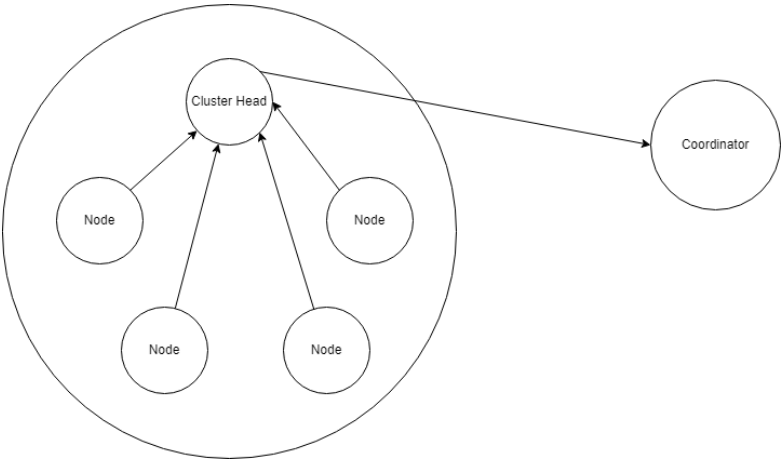
Modul nRF yang menggunakan daya transmisi tinggi akan menghabiskan baterai yang lebih banyak daripada modul nRF yang menggunakan daya transmisi rendah. Hal ini akan membuat perbedaan penggunaan energi antar *cluster* yang dapat kita ukur untuk menentukan selisih penurunan energinya.

Jarak yang telah ditentukan antara *cluster* dan *coordinator* akan membuat kemungkinan adanya *loss* saat transmisi paket. Paket loss dapat dikarenakan habisnya energi saat gelombang radio merambat atau tubrukan transmisi antara *node* yang

kebetulan mengirim transmisi secara bersamaan. Hal ini dapat kita gunakan untuk mengukur *Packet Delivery Ratio* atau PDR.



Gambar 5.10 *Cluster node* Tanpa pemilihan *Cluster Head*



Gambar 5.11 Cluster node dengan pemilihan Cluster Head

5.2.1 Skenario Uji Coba 1

Dalam scenario uji coba yang pertama kita akan menghitung performa *cluster node* tampa menggunakan *cluster head* dan tanpa mengatur kekuatan transmisi secara dinamis masing-masing *node* pada *cluster node*. Pada uji coba ini setiap *node* pada *cluster node* akan mengirimkan secara langsung packet pesan kepada *coordinator* sehingga kekuatan transmisinya di set maximal, Paket pesan yang akan dikirimkan masing-masing *node* berisikan *node* ID dan nomor pesan, Ini digunakan untuk mengetahui packet lost yang terjadi saat pengiriman langsung kepada *coordinator* Setiap *node* akan mengirimkan 25 paket ke *coordinator* sehingga ada total 100 paket yang akan dikirimkan ke *coordinator*. Di *coordinator* kita akan melihat jumlah *packet lost* yang terjadi, Jarak antara *node* dan *coordinator* dalam uji coba ini adalah 20 sampai 50meter dengan penambahan jarak 10meter setiap uji coba.

Untuk penurunan daya baterai setiap baterai akan di isi ulang penuh dan diukur terlebih dahulu sebelum melakukan uji coba. Setelah uji coba selesai baterai aka diukur lagi dan di data.

Hasil uji coba pengiriman dengan metode pemilihan *cluster head* menggunakan 2 *node* terdapat pada table 5.1, 5.2 dan 5.3

Table 5.1 Perhitungan jumlah *packet loss* pada *cluster node* tanpa pemilihan *cluster head*

Jarak (Meter)	**Paket Dikirim**	**Packet Lost**	**Persentase Packet Lost**	**PDR**
20	100	0	0%	100%
30	100	0	0%	100%
40	100	0	0%	100%
50	100	0	0%	100%

Dari hasil uji coba yang di lakukan menggunakan metode *direct transmision* dengan menggunakan 2 node rata-rata *packet delivery ratio* nya adalah 100%.

Table 5.2 Penurunan voltase baterai pada *cluster node* tanpa pemilihan *cluster head*

Node	**Voltase Awal**	**Voltase akhir**	**Penurunan Voltase**
1	9.50	8.44	1.11
2	9.50	8.53	0.97
Rata-Rata Penurunan Voltase			1.04

Dari hasil uji coba yang di lakukan menggunakan metode *direct transmision* dengan menggunakan 2 node rata-rata penurunan voltase pada baterai sebesar 1.04V.

Table 5.3 Penurunan arus baterai pada *cluster node* tanpa pemilihan *cluster head*

Node	**Arus Awal**	**Arus akhir**	**Penurunan Arus**
1	1.50	0.67	0.83
2	1.50	0.55	0.95
Rata-Rata Penurunan Arus			0.89

Dari hasil uji coba yang di lakukan menggunakan metode *direct transmision* dengan menggunakan 2 node terjadi penurunan arus pada baterai sebesar 0.89A.

Hasil uji coba pengiriman dengan metode pemilihan *cluster head* menggunakan 2 *node* terdapat pada table 5.4, 5.5 dan 5.6

Table 5.4 Perhitungan jumlah *packet loss* pada *cluster node* tanpa pemilihan *cluster head*

Jarak (Meter)	**Paket Dikirim**	**Packet Lost**	**Persentase Packet Lost**	**PDR**
20	100	0	0%	100%
30	100	1	1%	99%
40	100	4	3%	96%
50	100	5	5%	93%

Dari hasil uji coba yang di lakukan menggunakan metode *direct transmision* dengan menggunakan 3 node rata-rata *packet delivery ratio* nya adalah 97,5%.

Table 5.5 Penurunan voltase baterai pada *cluster node* tanpa pemilihan *cluster head*

Node	**Voltase Awal**	**Voltase akhir**	**Penurunan Voltase**
1	9.50	8.46	1.04
2	9.50	8.39	1.11
3	9.50	8.50	1.00
Rata-Rata Penurunan Voltase			1.05

Dari hasil uji coba yang di lakukan menggunakan metode *direct transmision* dengan menggunakan 3 node rata-rata penurunan voltase pada baterai sebesar 1.05V.

Table 5.6 Penurunan arus baterai pada *cluster node* tanpa pemilihan *cluster head*

Node	**Arus Awal**	**Arus akhir**	**Penurunan Arus**
1	1.50	0.57	0.93
2	1.50	0.65	0.85
3	1.50	0.54	0.96
Rata-Rata Penurunan Arus			0.911

Dari hasil uji coba yang di lakukan menggunakan metode *direct transmision* dengan menggunakan 3 node terjadi penurunan arus pada baterai sebesar 0.911A.

Hasil Uji coba pengiriman dengan metode *direct transmision* menggunakan 4 node terdapat pada table 5.7, 5.8 dan 5.9

Table 5.7 Perhitungan jumlah *packet loss* pada *cluster node* tanpa pemilihan *cluster head*

Jarak (Meter)	**Paket Dikirim**	**Packet Lost**	**Persentase Packet Lost**	**PDR**
20	100	0	0%	100%
30	100	1	1%	99%
40	100	4	4%	96%
50	100	7	7%	93%

Dari hasil uji coba yang di lakukan menggunakan metode *direct transmision* dengan menggunakan 4 node rata-rata *packet delivery ratio* nya adalah 97%.

Table 5.8 Penurunan voltase baterai pada *cluster node* tanpa pemilihan *cluster head*

Node	**Voltase Awal**	**Voltase akhir**	**Penurunan Voltase**
1	9.50	8.41	1.09
2	9.50	8.33	1.17
3	9.50	8.42	1.18
4	9.50	8.45	1.15
Rata-Rata Penurunan Voltase			1.15

Dari hasil uji coba yang di lakukan menggunakan metode *direct transmision* dengan menggunakan 4 node rata-rata penurunan voltase pada baterai sebesar 1.15V.

Table 5.9 Penurunan arus baterai pada *cluster node* tanpa pemilihan *cluster head*

Node	**Arus Awal**	**Arus akhir**	**Penurunan Arus**
1	1.50	0.68	0.88
2	1.50	0.53	0.97
3	1.50	0.50	1.00
4	1.50	0.60	0.90
Rata-Rata Penurunan Arus			0.93

Dari hasil uji coba yang di lakukan menggunakan metode *direct transmision* dengan menggunakan 4 node terjadi penurunan arus pada baterai sebesar 0.93A.

Hasil uji coba pengiriman dengan metode pemilihan *cluster head* menggunakan 2 *node* terdapat pada table 5.10, 5.11 dan 5.12

Table 5.10 Perhitungan jumlah *packet loss* pada *cluster node* dengan pemilihan *cluster head*

Jarak (Meter)	**Paket Dikirim**	**Packet Lost**	**Persentase Packet Lost**	**PDR**
20	100	0	0%	100%
30	100	1	1%	99%
40	100	5	5%	95%
50	100	7	7%	93%

Dari hasil uji coba yang di lakukan menggunakan metode *direct transmision* dengan menggunakan 5 node rata-rata *packet delivery ratio* nya adalah 96,75%.

Table 5.11 Penurunan voltase baterai pada *cluster node* dengan pemilihan *cluster head*

Node	**Voltase Awal**	**Voltase akhir**	**Penurunan Voltase**
1	9.50	8.41	1.09
2	9.50	8.43	1.07
3	9.50	8.38	1.12
4	9.50	8.29	1.21
5	9.50	8.37	1.13
Rata-Rata Penurunan Voltase			1.16

Dari hasil uji coba yang di lakukan menggunakan metode *direct transmision* dengan menggunakan 5 node rata-rata penurunan voltase pada baterai sebesar 1.16V.

Table 5.12 Penurunan arus baterai pada *cluster node* tanpa pemilihan *cluster head*

Node	**Arus Awal**	**Arus akhir**	**Penurunan Arus**
1	1.50	0.64	0.88
2	1.50	0.55	0.95
3	1.50	0.50	1.00
4	1.50	0.57	0.93
5	1.50	0.58	0.92
Rata-Rata Penurunan Arus			0.95

Dari hasil uji coba yang di lakukan menggunakan metode *direct transmision* dengan menggunakan 5 node terjadi penurunan arus pada baterai sebesar 0.95A.

5.2.2 Skenario Uji Coba 2

Dalam scenario uji coba yang ke 2 kita akan menghitung performa pada *cluster node* dengan menggunakan pemilihan *cluster head* dan pengaturan kekuatan transmisi secara *dinamis* dengan menggunakan parameter penggunaan energi pada masing masing *node* sebagai penentu *cluster head.* Pada uji coba ini setiap *node* akan mengirimkan pesan ke *cluster head* terlebih dahalu sebelum kemudia di kirim ke *coordinator*. Cara untuk mengetahui packet lost adalah dengan memberikan penomoran pada saat pengiriman dan di sertakan *node* ID sehingga kita bias mengetahui berapa jumlah *packet lost* serta dari *node* mana *packet lost* itu berasal. Setiap *node* akan mengirimkan pesan sebanyak 25 paket sehingga total 100 paket yang akan di kirimkan ke *coordinator*. Jarak antara *cluster node* dengan coordinator dalam uji coba ini adalah 20 sampai 50 meter, dengan penambahan jarak 10meter setiap uji coba.

Untuk penurunan daya baterai setiap baterai akan di isi ulang penuh dan diukur terlebih dahulu sebelum melakukan uji coba. Setelah uji coba selesai baterai aka diukur lagi dan di data.

Hasil Uji coba pengiriman dengan menggunakan *cluster head* dengan menggunakan 2 node terdapat pada table 5.13, 5.14 dan 5.15

Table 5.13 Perhitungan jumlah *packet loss* pada *cluster node* dengan Pemilihan *cluster head*

Jarak (Meter)	**Paket Dikirim**	**Packet Lost**	**Persentase Packet Lost**	**PDR**
20	100	0	0%	100%
30	100	0	0%	100%
40	100	0	0%	100%
50	100	0	0%	100%

Dari hasil uji coba yang di lakukan menggunakan metode pemilihan *cluster head* dengan menggunakan 2 node rata-rata *packet delivery ratio* nya adalah 100%.

Table 5.14 Penurunan voltase baterai pada *cluster node* dengan pemilihan *cluster head*

Node	**Voltase Awal**	**Voltase akhir**	**Penurunan Voltase**
1	9.50	8.70	0.80
2	9.50	8.69	0.81
Rata-Rata Penurunan Voltase			0.805

Dari hasil uji coba yang di lakukan menggunakan metode pemilihan *cluster head* dengan menggunakan 2 node rata-rata penurunan voltase pada baterai sebesar 0.805V.

Table 5.15 Penurunan arus baterai pada *cluster node* dengan pemilihan *cluster head*

Node	**Arus Awal**	**Arus akhir**	**Penurunan Arus**
1	1.50	0.69	0.81
2	1.50	0.71	0.79
Rata-Rata Penurunan Voltase			0.80

Dari hasil uji coba yang di lakukan menggunakan metode pemilihan *cluster head* dengan menggunakan 2 node terjadi penurunan arus pada baterai sebesar 0.80A.

Hasil Uji coba pengiriman dengan menggunakan *cluster head* dengan menggunakan 3 node terdapat pada table 5.16, 5.17 dan 5.18

Table 5.16 Perhitungan jumlah *packet loss* pada *cluster node* dengan pemilihan *cluster head*

Jarak (Meter)	**Paket Dikirim**	**Packet Lost**	**Persentase Packet Lost**	**PDR**
20	100	0	0%	100%
30	100	0	0%	100%
40	100	1	1%	99%
50	100	2	2%	98%

Dari hasil uji coba yang di lakukan menggunakan metode pemilihan *cluster head* dengan menggunakan 3 node rata-rata *packet delivery ratio* nya adalah 99,3%.

Table 5.17 Penurunan voltase baterai pada *cluster node* dengan pemilihan *cluster head*

Node	**Voltase Awal**	**Voltase akhir**	**Penurunan Voltase**
1	9.50	8.51	0.99
2	9.50	8.85	0.65
3	9.50	8.80	0.70
Rata-Rata Penurunan Voltase			0.79

Dari hasil uji coba yang di lakukan menggunakan metode pemilihan *cluster head* dengan menggunakan 3 node rata-rata penurunan voltase pada baterai sebesar 0.79V.

Table 5.18 Penurunan arus baterai pada *cluster node* dengan pemilihan *cluster head*

Node	**Arus Awal**	**Arus akhir**	**Penurunan Arus**
1	1.50	0.75	0.75
2	1.50	0.71	0.79
3	1.50	0.66	0.84
Rata-Rata Penurunan Arus			0.79

Dari hasil uji coba yang di lakukan menggunakan metode pemilihan *cluster head* dengan menggunakan 3 node terjadi penurunan arus pada baterai sebesar 0.79A.

Hasil Uji coba pengiriman dengan menggunakan *cluster head* dengan menggunakan 4 node terdapat pada table 5.19, 5.20 dan 5.21

Table 5.19 Perhitungan jumlah *packet loss* pada *cluster node* dengan pemilihan *cluster head*

Jarak (Meter)	**Paket Dikirim**	**Packet Lost**	**Persentase Packet Lost**	**PDR**
20	100	0	0%	100%
30	100	0	0%	99%
40	100	2	2%	98%
50	100	4	4%	96%

Dari hasil uji coba yang di lakukan menggunakan metode pemilihan *cluster head* dengan menggunakan 4 *node* rata-rata *packet delivery ratio* nya adalah 98,5%.

Table 5.20 Penurunan voltase baterai pada *cluster node* dengan pemilihan *cluster head*

Node	**Voltase Awal**	**Voltase akhir**	**Penurunan Voltase**
1	9.50	8.51	0.99
2	9.50	8.77	0.73
3	9.50	8.62	0.88
4	9.50	8.88	0.62
Rata-Rata Penurunan Voltase			0.81

Dari hasil uji coba yang di lakukan menggunakan metode pemilihan *cluster head* dengan menggunakan 4 *node* rata-rata penurunan voltase pada baterai sebesar 1.04V.

Table 5.21 Penurunan arus baterai pada *cluster node* dengan pemilihan *cluster head*

Node	**Arus Awal**	**Arus akhir**	**Penurunan Arus**
1	1.50	1.50	0.73
2	1.50	1.50	0.70
3	1.50	1.50	0.74
4	1.50	1.50	0.63
Rata Rata Penurunan Arus			0.80

Dari hasil uji coba yang di lakukan menggunakan metode pemilihan *cluster head* dengan menggunakan 4 node terjadi penurunan arus pada baterai sebesar 0.80A.

Hasil Uji coba pengiriman dengan menggunakan *cluster head* dengan menggunakan 5 node terdapat pada table 5.22, 5.23 dan 5.24

Table 5.22 Perhitungan jumlah *packet loss* pada *cluster node* dengan pemilihan *cluster head*

Jarak (Meter)	**Paket Dikirim**	**Packet Lost**	**Persentase Packet Lost**	**PDR**
20	100	0	0%	100%
30	100	0	0%	100%
40	100	2	2%	98%
50	100	4	4%	96%

Dari hasil uji coba yang di lakukan menggunakan metode pemilihan *cluster head* dengan menggunakan 5 node rata-rata *packet delivery ratio* nya adalah 98,5%.

Table 5.23 Penurunan voltase baterai pada *cluster node* dengan pemilihan *cluster head*

Node	**Voltase Awal**	**Voltase akhir**	**Penurunan Voltase**
1	9.50	8.41	1.09
2	9.50	8.73	0.77
3	9.50	8.82	0.68
4	9.50	8.73	0.77
5	9.50	8.85	0.65
Rata-Rata Penurunan Voltase			0.80

Dari hasil uji coba yang di lakukan menggunakan metode pemilihan *cluster head* dengan menggunakan 5 node rata-rata penurunan voltase pada baterai sebesar 0.80V.

Table 5.24 Penurunan arus baterai pada *cluster node* dengan pemilihan *cluster node*

Node	**Arus Awal**	**Arus akhir**	**Penurunan Arus**
1	1.50	0.70	0.80
2	1.50	0.73	0.77
3	1.50	0.65	0.85
4	1.50	0.63	0.87
5	1.50	0.75	0.75
Rata-Rata Penurunan Arus			0.81

Dari hasil uji coba yang di lakukan menggunakan metode pemilihan *cluster head* dengan menggunakan 5 node terjadi penurunan arus pada baterai sebesar 0.81A.

5.3 Evaluasi Umum Skenario Uji Coba

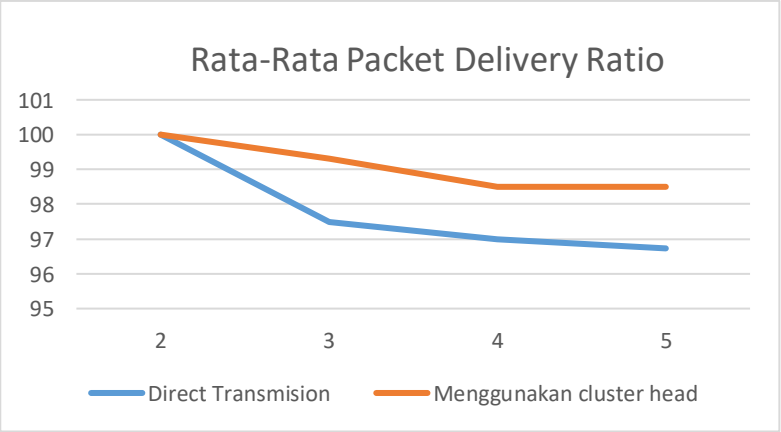
Berikut adalah evaluasi dari hasil uji coba dari masing skenario. Terdapat 2 point yang akan di evaluasi yaitu *Packet Delivery Ratio* dan penurunan daya baterai.

5.3.1 *Packet Delivery Ratio*

Dari uji coba yang di lakukan terjadi penurunan jumlah *packet lost* ketika menggunakan metode pemilihan *cluster node* dibandingkan dengan *Direct Transmision*. Berikut adalah table hasil uji coba secara keseluruhan untuk *packet delivery ratio:*

Table 5.25 Hasil uji coba keseluruhan untuk *packet delivery ratio*

Jumlah Node	*Direct Transmision*	Menggunakan *Cluster node*
2	100%	100%
3	97,5%	99,3%
4	97,0%	98,5%
5	96,73%	98,5%
Rata-Rata	66,07%	57,25%



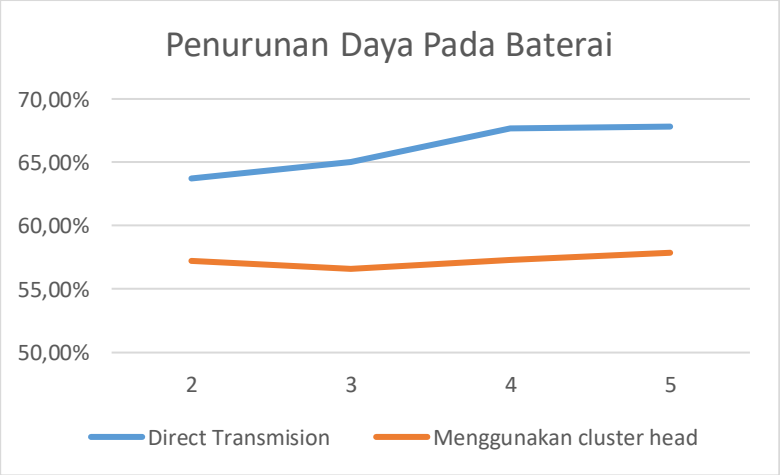
Gambar 5.12 Grafik rata-rata *packet delivery ratio*

5.3.2 Penurunan Daya Baterai

Dari uji coba yang di lakukan terjadi penghematan penggunaan energi ketika menggunakan metode pemilihan *cluster node* dibandingkan dengan *Direct Transmision*. Berikut adalah hasil uji coba secara keseluhan untuk penurunan daya baterai:

Table 5.26 Hasil uji coba untuk penurunan daya baterai

Jumlah Node	*Direct Transmision*	Menggunakan *Cluster node*
2	63,73 %	57,2%
3	65,02%	56,60%
4	67,70%	57,31%
5	67,82%	57,87%
Rata-Rata	66,07%	57,24%



Gambar 5.13 Grafik penurunan daya pada baterai

(Halaman ini sengaja dikosongkan)

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas mengenai kesimpulan yang diperoleh dari tugas akhir yang telah dikerjakan dan saran terkait pengembangan dari tugas akhir ini yang dapat dilakukan pada masa yang akan datang.

6.1 Kesimpulan

Kesimpulan yang diperoleh dari hasil uji coba dan evaluasi pada tugas akhir ini adalah sebagai berikut:

1. Dengan menggunakan metode pemilihan *cluster node* dan melakukan monitoring terhadap penggunaan energi serta pengaturan kekuatan transmisi dapat digunakan sebagai salah satu metode untuk mengurangi *packet lost* sebesar 1.27% sehingga terjadi peningkatan *packet delivey ratio* sebesar 1.27% di banding dengan menggunakan *Direct transmission.*
2. Metode pemilihan *cluster head* dengan menggunakan parameter energi yang digunakan oleh masing-masing node serta pengaturan kekuatan transmisi secara *dinamis* lebih cocok digunakan untuk jaringan sensor nirkabel untuk dapat mengurangi penggunaan daya baterai sebesar 8.82% dibanding menggunakan *Direct Transmision.*
3. Metode pemilihan *cluster head* dan melakukan monitoring terhadap penggunaan energi serta pengaturan kekuatan *transmisi* dapat diterapkan pada lingkungan nyata dan menggunakan sensor nyata apabila jarak antar *node* masih dalam jangkauan radio modul nRF24L01.

6.2 Saran

Saran yang diberikan dari hasil uji coba dan evaluasi pada tugas akhir ini adalah sebagai berikut:

1. Menambahkan *security* dalam pengamanan pengiriman packet pesan

2. Melakukan implementasi *protocol* dengan penggunaan studi kasus yang nyata.

DAFTAR PUSTAKA

[1] B. Mao *et al.*, "A novel non-supervised deep-learning-based network traffic control method for software defined wireless networks," *IEEE Wirel. Commun.*, vol. 25, no. 4, pp. 74–81, 2018.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Comput. networks*, vol. 38, no. 4, pp. 393–422, 2002.

[3] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proceedings of the 33rd annual Hawaii international conference on system sciences*, 2000, pp. 10--pp.

[4] U. J. Shobrina, "Analisis Kinerja Pengiriman Data Modul Transceiver NRF24l01, Xbee dan Wifi ESP8266 Pada Wireless Sensor Network," Universitas Brawijaya, 2017.

[5] A. Rahmat, "Belajar Pemrograman Dasar Arduino." [Online]. Available: https://kelasrobot.com/belajar-pemrograman-dasar-arduino/. [Accessed: 15-Jun-2019].

[6] Henrysbench, "INA219 Arduino Current Sensor Voltmeter Tutorial." [Online]. Available: http://henrysbench.capnfatz.com/henrys-bench/arduino-current-measurements/ina219-arduino-current-sensor-voltmeter-tutorial-quick-start/. [Accessed: 15-Jun-2019].

(Halaman ini sengaja dikosongkan)

LAMPIRAN A

KODE SUMBER

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <Wire.h>
#include <Adafruit_INA219.h> // You will need to download this library

Adafruit_INA219 sensor219; // Declare and instance of INA219

RF24 radio(7, 8);
//status
const int nodeId=3;
int CH_ID=0;
int ch_status=0;
int ChAddr = 0;
int sinkStatus = 0;

//addres variabel
const int rxAddr = 5000;
const int rxAddr1 = 2001;
const int rxAddr2 = 2002;
const int rxAddr3 = 2003;
const int rxAddr4 = 2004;
const int sinkAddr = 1000;
const int bcAddr = 2010;


//ina219 variabel
float busVoltage = 0;
float current = 0; // Measure in milli amps
float power = 0;
float newpower=0;
int energi1 = 70;
int energi2 = 60;
int energi3 = 90;
int energi4 = 80;
```

```
36.
37. //counter
38. int scSentCH =0;  //counter untuk node yang mengi
    rimkan pesan ke cluster head
39. int MntStopCH =0; //counter untuk ch mengrimkan p
    esan stop
40. int bcMinute = 0; //counter untuk node yang mengi
    rimkan pesan broadcast
41. int bcSecond = 0; // //counter untuk node waiting
     broadcast
42. int bcCounter = 0; // untuk menunggu pesan broadc
    ast
43. bool bcStatus = false; //untuk mengecheck pesan b
    c
44.
45. int a = 0, xMili = 0, xSecond = 0, xMinute = 0, x
    Hour = 0, currentSecond = 0, currentMilis=0;
46. unsigned long wTime;
47. int start=0;
48. int flag_reset_time = 0;
49. void setup()
50. {
51.
52.   xMili = 0; xSecond = 0; xMinute = 0; xHour = 0;

53.
54.
55.   while (!Serial);
56.   Serial.begin(9600);
57.   sensor219.begin();
58.
59.   radio.begin();
60. //  radio.openReadingPipe(0, rxAddr);
61. //  radio.startListening();
62.
63. }
64.
65.
66. void countTime() {
67.   ++xMili;
68.   if (xMili == 999) {
69.     ++xSecond;
```

```
70.     xMili = 0;
71.   }
72.   if ((xSecond > 59)) {
73.     ++xMinute;
74.     xSecond = 0;
75.     currentSecond = -1;
76.   }
77.   if (xMinute > 59) {
78.     ++xHour;
79.     xMinute = 0;
80.
81.   }
82.
83. }
84.
85.
86. void getPower(){
87.   busVoltage = sensor219.getBusVoltage_V();
88.   current = sensor219.getCurrent_mA();
89.   newpower = abs(busVoltage * current/1000);
90. }
91.
92. void tapListening() {
93.
94. //   radio.openReadingPipe(0, rxAddr);
95. //   radio.startListening();
96.
97.   if (radio.available())
98.   {
99.     char rtext[25];
100.           radio.read(&rtext, sizeof(rtext));
101.
102.              if(rtext[0] == 'R'){
103.                 xSecond = 0;
104.                 xMinute = 0;
105.                 xHour = 0;
106.                 xMili = 0;
107.                 flag_reset_time = 1;
108.                 if(rtext[2]=='1'){
109.                 ChAddr = rxAddr1 ;
110.                 CH_ID = 1;
111.                 }
```

```
112.              }
113.              else if(rtext[0]=='S'){
114.                 ch_status = 0;
115.                 CH_ID = 0;
116.              Serial.println(rtext);
117.             }
118.
119.           Serial.println(rtext);
120.          }
121.
122.       }
123.
124.       void tapListening(int addres) {
125.
126.          radio.openReadingPipe(0, addres);
127.          radio.startListening();
128.         // Serial.println("masuk");
129.         if (radio.available())
130.         {
131.            char rtext[25];
132.           radio.read(&rtext, sizeof(rtext));
133.           if(rtext[0] == 'R'){
134.                 xSecond = 0;
135.                 xMinute = 0;
136.                 xHour = 0;
137.                 xMili = 0;
138.                 flag_reset_time = 1;
139.                 if(rtext[2]=='1'){
140.                 ChAddr = rxAddr1 ;
141.                 CH_ID = 1;
142.                 }
143.           }
144.           else if(rtext[0]=='S'){
145.                ch_status = 0;
146.                CH_ID = 0;
147.             Serial.println(rtext);
148.            }
149.            else if(rtext[0]=='B'){
150.                Serial.println(rtext);
151.                 if(rtext[2]=='1'){
152.                   energi1 = atoi(&rtext[3]);
153.                 }
```

```
154.                    else if(rtext[2]=='2'){
155.                      energi2 = atoi(&rtext[3]);

156.                    }
157.                    else if(rtext[2]=='3'){
158.                      energi3 = atoi(&rtext[3]);

159.                    }
160.                    else if(rtext[2]=='4'){
161.                      energi4 = atoi(&rtext[3]);

162.                    }
163.                    Serial.println(energi1);
164.                    Serial.println(energi2);
165.                    Serial.println(energi3);
166.                    Serial.println(energi4);
167.
168.
169.                }
170.                else if(rtext[0]=='I'){
171.                    String msg = "CH ADALAH NODE :";

172.                    msg.concat(rtext[2]);
173.                    Serial.println(msg);
174.                 }
175.                  else if(rtext[0]=='D'){
176.                     transmit(sinkAddr,rtext);
177.                     Serial.println(rtext);
178.                  }
179.
180.
181.              Serial.println(rtext);
182.            }
183.          }
184.
185.
186.          void transmit(String Message)
187.          {
188.
189.            radio.setRetries(15, 15);
190.            radio.stopListening();
191.            radio.openWritingPipe(rxAddr);
```

```
192.
193.          String c = Message;
194.          char pesan[25];
195.          c.toCharArray(pesan, 25);
196.
197.          int  rNumber  = random(1,5);
198.          delay(rNumber);
199.
200.          radio.write(&pesan, sizeof(pesan));
201.          radio.openReadingPipe(0, rxAddr);
202.          radio.startListening();
203.
204.            if(pesan[0] == 'R'){
205.                xSecond = 0;
206.                xMinute = 0;
207.                xHour = 0;
208.                xMili = 0;
209.                flag_reset_time = 1;
210.                ch_status =1;
211.                CH_ID = 1;
212.                ChAddr = rxAddr1;
213.              }
214.              Serial.println(ChAddr);
215.        }
216.
217.
218.
219.
220.        void transmit(int sendAddr, String Message
    )
221.        {
222.
223.          radio.setRetries(15, 15);
224.          radio.stopListening();
225.          radio.openWritingPipe(sendAddr);
226.
227.          String c = Message;
228.          char pesan[25];
229.          c.toCharArray(pesan, 25);
230.
231.          int  rNumber  = random(1,5);
232.          delay(rNumber);
```

```
233.
234.          radio.write(&pesan, sizeof(pesan));
235.          radio.openReadingPipe(0, sendAddr);
236.          radio.startListening();
237.             if (pesan[0] == 'S') {
238.                  ch_status = 0;
239.                  CH_ID =0;
240.              }
241.
242.        }
243.
244.
245.        void loop()
246.        {
247.           power=power+newpower;
248.          getPower();
249.
250.          currentSecond = xSecond;
251.          countTime();
252.
253.          if(flag_reset_time==0) {
254.              tapListening();
255.
256.                 if((xSecond == 3) && (xMili==1)&&(
    nodeId==1))
257.                    {
258.                      char text[25];
259.                      String ResetTime = "R#";
260.                      ResetTime.concat(nodeId);
261.                      transmit(ResetTime);
262.                      Serial.println(ResetTime) ;

263.                    }
264.                    if(((xSecond % 1)==0) && (xMil
    i==1))
265.                    {
266.                      char text[25];
267.                       String ResetTime = String(x
    Hour, DEC) + ":" + String(xMinute, DEC) + ":" + S
    tring(xSecond, DEC);
268.                       ResetTime.concat(power);
```

```
                    transmit(sinkAddr,ResetTime)
;
                    Serial.println(ResetTime) ;

                    }

          }

        else if ((flag_reset_time == 1)&& (CH_ID
==0)){
            tapListening(bcAddr);
            radio.setPALevel(RF24_PA_LOW);

            if((abs(xSecond-bcSecond))>1){
                bcSecond=xSecond;
                bcCounter++;
                if (bcCounter==4){
                    bcStatus = true;
                  }
                Serial.print(bcCounter);
                Serial.println(bcStatus);
              }
            if (((xMinute % 1)==0) && (xSecond =
= 1) && ((abs(xMinute-bcMinute))>=1) )
                {
                char text[25];
                String bcEnergy = "B#";
                bcEnergy.concat(nodeId);
                bcEnergy.concat(power);
                transmit(bcAddr,bcEnergy);
                Serial.println(bcEnergy) ;
                bcMinute = xMinute;
                }
          }
        else if ((ch_status==1)&&(flag_reset_tim
e ==1) && (CH_ID!=0)){
            tapListening(ChAddr);
            radio.setPALevel(RF24_PA_HIGH);
                if (((xMinute % 1)==0) && (xSeco
nd == 1) && ((abs(xMinute-MntStopCH))>=0) )
                {
                    char text[25];
```

```
305.                        String stopCH = "S#";
306.                        stopCH.concat(nodeId);
307.                        transmit(ChAddr,stopCH);
308.                        Serial.println(stopCH) ;
309.                        MntStopCH=xMinute;
310.
311.                    }else if ((xSecond == 59) && (xM
    ili==1) )
312.                    {
313.                        char text[25];
314.                        String sendTosink = "I#";
315.                        sendTosink.concat(nodeId);
316.                        transmit(sinkAddr,sendTosink
    );
317.                        Serial.println(sendTosink) ;

318.                    }
319.
320.            }
321.          else if ((ch_status==0)&&(flag_reset_tim
    e ==1)&& (CH_ID!=0)){
322.              tapListening();
323.              radio.setPALevel(RF24_PA_LOW);
324.                  if  (((xSecond %3) ==0) && ((abs
    (xSecond-scSentCH))>2) )
325.                  {
326.                      char text[25];
327.                      String sendtoCH = "DATA#";
328.                      sendtoCH.concat(nodeId);
329.                      transmit(ChAddr,sendtoCH);
330.                      scSentCH=xSecond;
331.                      Serial.println(sendtoCH) ;
332.                  }
333.
334.            }else if (sinkStatus == 1){
335.                tapListening(sinkAddr);
336.              }
337.
338.
339.             if(bcStatus==true){
340.                if (energi1<=energi2 && energi1<=e
    nergi3 && energi1<=energi4){
```

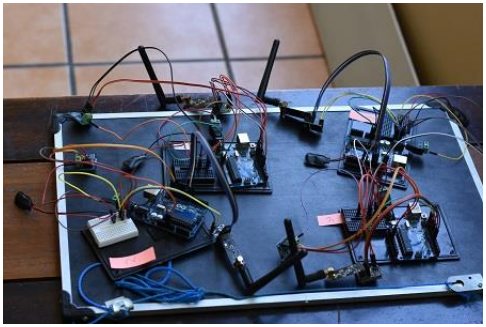
```
341.                          CH_ID = 1;
342.                          ChAddr= rxAddr1;
343.                          if(nodeId==1){
344.                            ch_status=1;
345.                             char text[25];
346.                             String sendTosink = "I#
    ";
347.                             sendTosink.concat(nodeI
    d);
348.                             transmit(bcAddr,sendTos
    ink);
349.
350.                          }else{
351.                            ch_status=0;
352.                           }
353.                      bcStatus=false;
354.                      Serial.print("CH1");
355.                      }else if (energi2<=energi1 &&
    energi2<=energi3 && energi2<=energi4){
356.                          CH_ID = 2;
357.                          ChAddr= rxAddr2;
358.                          if(nodeId==2){
359.                            ch_status=1;
360.                                char text[25];
361.                                String sendTosink =
    "I#";
362.                                sendTosink.concat(no
    deId);
363.                                transmit(bcAddr,send
    Tosink);
364.
365.                          }else{
366.                            ch_status=0;
367.                           }
368.                            bcStatus=false;
369.                            Serial.print("CH2");
370.                      }else if (energi3<=energi1 &&
    energi3<=energi3 && energi3<=energi4){
371.                          CH_ID = 3;
372.                          ChAddr= rxAddr3;
373.                          if(nodeId==3){
374.                            ch_status=1;
```

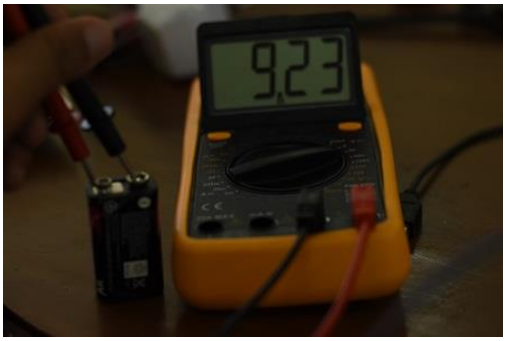
```
375.                                char text[25];
376.                                String sendTosink =
    "I#";
377.                                sendTosink.concat(no
    deId);
378.                                transmit(bcAddr,send
    Tosink);
379.
380.                            }else{
381.                              ch_status=0;
382.                             }
383.                             bcStatus=false;
384.                             Serial.print("CH3");
385.                        }else if (energi4<=energi1 &&
    energi4<=energi2 && energi4<=energi2){
386.                            CH_ID = 4;
387.                            ChAddr= rxAddr4;
388.                            if(nodeId==4){
389.                              ch_status=1;
390.                                   char text[25];
391.                                   String sendTosink =
     "I#";
392.                                   sendTosink.concat(n
    odeId);
393.                                   transmit(bcAddr,sen
    dTosink);
394.
395.                            }else{
396.                              ch_status=0;
397.                             }
398.                             bcStatus=false;
399.                             Serial.print("CH4");
400.                       }
401.                       bcCounter=0;
402.                       Serial.println(bcStatus);
403.                       Serial.println(bcCounter);
404.                       Serial.print(ChAddr);
405.             }
406.
407.
408.
409.         delay(1);
```

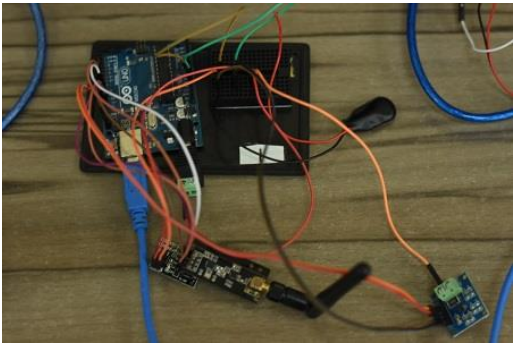
```
410.        }
```

LAMPIRAN B

DOKUMENTASI IMPLEMENTASI




9,23



(Halaman ini sengaja dikosongkan)

BIODATA PENULIS

Zahri Rusli lahir di Solok Pada tanggal 16 Februari 1998. Penulis menempuh Pendidikan formal di MIN Kota Solok (2003-2006), SDN 39 Koto Baru Solok (2006-2009) MTsN Koto Baru Solok (2009-2012), SMAN 1 Kota Solok (2012-2015), dan Informatika ITS Surabaya (2015-2019). Bidang studi yang diambil oleh penulis saat berkuliah di Departemen Informatika ITS adalah Komputasi Berbasis Jaringan (KBJ). Penulis aktif dalam organisasi Himpunan Mahasiswa Teknik Computer-Informatika (2016-2017), Paguyuban Beasiswa Karya Salemba Empat ITS Surabaya (2016-2018), Dan Future Leader For Anti Coruption Surabaya (2015-2016). Penulis juga aktif dalam kegiatan kepanitian seperti SCHEMATICS 2016 - 2017 Divisi NST, FTIF FESTIVAL 2017 sebagai Staff Ahli Acara, dan Try Out Nasional Karya Salembat Empat 2017 sebagai Ketua Regional Surabaya. Penulis pernah menjalani kerja praktik di Elifesolutions Johor periode Januari-Februari 2018, magang di Elifesolutions Johor periode Juli-September 2018. Selama berkuliah, penulis juga menjadi administrator di Laboratorium Komputasi Berbasis Jaringan. Penulis dapat dihubungi melalui nomor *handphone* 082288023466 atau di email zahri.rusli@gmail.com.