ITS
Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR - KI141502

PERANCANGAN DAN IMPLEMENTASI PEMILIHAN CLUSTER HEAD, KOORDINASI ANTAR NODE SENSOR, DAN PENGATURAN KEKUATAN TRANSMISI SECARA DINAMIS PADA JARINGAN SENSOR NIRKABEL DENGAN MODUL WIRELESS NRF24L01

I GEDE PUTU NOBBY ASWI PHALA
NRP 05111440000048

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

Dosen Pembimbing II
Dr. Eng. Radityo Anggoro,S.Kom.,M.Sc

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya, 2018

(Halaman ini sengaja dikosongkan)

TUGAS AKHIR - KI141502

PERANCANGAN DAN IMPLEMENTASI PEMILIHAN CLUSTER HEAD, KOORDINASI ANTAR NODE SENSOR, DAN PENGATURAN KEKUATAN TRANSMISI SECARA DINAMIS PADA JARINGAN SENSOR NIRKABEL DENGAN MODUL WIRELESS NRF24L01

I GEDE PUTU NOBBY ASWI PHALA
NRP 05111440000048

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

Dosen Pembimbing II
Dr. Eng. Radityo Anggoro,S.Kom.,M.Sc

JURUSAN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya, 2018

(Halaman ini sengaja dikosongkan)

UNDERGRADUATE THESIS - KI141502

DESIGN AND IMPLEMENTATION CLUSTER HEAD SELECTION NODES COORDINATION AND DYNAMIC ARRANGEMENT OF TRANSMISSION POWER IN WIRELESS SENSOR NETWORKS BASED ON NRF24L01 MODULE

I GEDE PUTU NOBBY ASWI PHALA
NRP 05111440000048

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

Supervisor II
Dr. Eng. Radityo Anggoro,S.Kom.,M.Sc

Department of INFORMATICS
Faculty of Information and Communication Technology
Institut Teknologi Sepuluh Nopember
Surabaya, 2018

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

PERANCANGAN DAN IMPLEMENTASI PEMILIHAN CLUSTER HEAD, KOORDINASI ANTAR NODE SENSOR, DAN PENGATURAN KEKUATAN TRANSMISI SECARA DINAMIS PADA JARINGAN SENSOR NIRKABEL DENGAN MODUL WIRELESS NRF24L01

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Berbasis Jaringan
Program Studi S1 Jurusan Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh :

I GEDE PUTU NOBBY ASWI PHALA
NRP: 05111440000048

Disetujui oleh Dosen Pembimbing Tugas Akhir:

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

Dr. Eng. Radityo Anggoro,S.Kom.,M.Sc ..... (Pembimbing 2)
NIP: 198410162008121002

SURABAYA
JULI 2018

(Halaman ini sengaja dikosongkan)

PERANCANGAN DAN IMPLEMENTASI PEMILIHAN CLUSTER HEAD, KOORDINASI ANTAR NODE SENSOR, DAN PENGATURAN KEKUATAN TRANSMISI SECARA DINAMIS PADA JARINGAN SENSOR NIRKABEL DENGAN MODUL WIRELESS NRF24L01

Nama	:	**I GEDE PUTU NOBBY ASWI PHALA**
NRP	:	**05111440000048**
Jurusan	:	**Informatika FTIK**
Pembimbing I	:	**Waskitho Wibisono, S.Kom.,M.Eng.,Ph.D.**
Pembimbing II	:	**Dr. Eng. Radityo Anggoro,S.Kom.,M.Sc**

Abstrak

Wireless Sensor Network (WSN) adalah sebuah jaringan yang terdiri dari node-node yang didistribusikan baik secara strategis mauun secara random yang bertujuan 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 antarmuka SPI (Serial Peripheral Interface) untuk berkomunikasi. Modul ini didesain untuk jaringan nirkabel yang membutuhkan daya sangat rendah. Sehingga cocok digunakan pada node-node dalam jaringan sensor network.

Pada jaringan sensor yang akan dibuat, diharapkan semua node dapat mengirimkan data ke sebuah server / coordinator.

Namun, apabila semua node mengirim langsung ke server / coordinator, maka penggunaan energi dalam jaringan akan menjadi boros. Sehingga perlu adanya node yang menjadi cluster head dalam jaringan untuk mengumpulkan data dari node lain sebelum mengirimkannya ke server / coordinator. Namun, node yang menjadi cluster head harus mengirimkan data dalam jumlah besar sehingga node akan cepat kehabisan energi dan mengakibatkan energi dalam jaringan menjadi tidak seimbang. Dikarenakan besarnya penggunaan energi tersebut, maka diperlukan sebuah metode pemilihan cluster head yang efisien untuk menangani permasalahan tersebut.

Metode yang penulis gunakan dalam melakukan pemilihan cluster head adalah dengan menentukan sebuah node yang akan menjadi cluster head pertama. Kemudian setelah beberapa waktu berlalu, node yang menjadi cluster head akan menentukan node yang akan menjadi cluster head selanjutnya dan mengirimkan broadcast ke node-node yang lain untuk memberitahukan node cluster head yang baru.

DESIGN AND IMPLEMENTATION CLUSTER HEAD SELECTION NODES COORDINATION AND DYNAMIC ARRANGEMENT OF TRANSMISSION POWER IN WIRELESS SENSOR NETWORKS BASED ON NRF24L01 MODULE

Name	:	**I GEDE PUTU NOBBY ASWI PHALA**
NRP	:	**05111440000048**
Major	:	**Informatics FTIK**
Supervisor I	:	**Waskitho Wibisono, S.Kom.,M.Eng.,Ph.D.**
Supervisor II	:	**Dr. Eng. Radityo Anggoro,S.Kom.,M.Sc**

Abstract

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

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.

In the sensor network to be created, it is expected that all nodes can transmit data to a server / coordinator. However, if all nodes send directly to the server / coordinator, then the use of energy in the network will be wasteful. So it is necessary to node a cluster head in the network to collect data from another node before sending it to the server / coordinator. However, the node

that becomes the cluster head must transmit large amounts of data so that the nodes will quickly run out of energy and cause the energy in the network to be unbalanced. Due to the enormous use of these energies, an efficient cluster head selection method is needed to address the problem.

The method that the author uses in choosing a cluster head is to determine a node that will be the first cluster head. Then after some time, the cluster head will determine which node will be the next cluster head and send broadcast to the other nodes to notify the new cluster head node.

KATA PENGANTAR

Astungkara, segala puji dan syukur bagi Ida Sang Hyang Widhi Wasa, yang telah melimpahkan nikmat, dan rejeki-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul **PERANCANGAN DAN IMPLEMENTASI PEMILIHAN CLUSTER HEAD, KOORDINASI ANTAR NODE SENSOR, DAN PENGATURAN KEKUATAN TRANSMISI SECARA DINAMIS PADA JARINGAN SENSOR NIRKABEL DENGAN MODUL WIRELESS NRF24L01** . yang merupakan salah satu syarat dalam menempuh ujian sidang guna memperoleh gelar Sarjana Komputer. Selesainya Tugas Akhir ini tidak terlepas dari bantuan dan dukungan beberapa pihak, sehingga pada kesempatan ini penulis mengucapkan terima kasih kepada:

1. Ida Sang Hyang Widhi Wasa, karena atas izin-Nya lah penulis dapat menyelesaikan Tugas Akhir dengan baik.
2. Kedua orang tua, Made Yudiastika serta Ni Made Ayu Wiratningsih, terima kasih atas doa dan bantuan moral dan material selama penulis belajar di departemen Informatika ITS.
3. Bapak Waskitho Wibisono, S.Kom., M.Eng., Ph.D. selaku pembimbing I yang telah membimbing dan memberikan motivasi, nasehat dan bimbingan dalam menyelesaikan Tugas Akhir ini sekaligus dosen wali penulis yang telah memberikan arahan, masukan dan motivasi kepada penulis.
4. Bapak Dr. Eng. Radityo Anggoro, S.Kom., M.Sc. selaku pembimbing II yang telah membimbing dan memberikan motivasi, nasehat dan bimbingan dalam menyelesaikan Tugas Akhir ini dan selaku koordinator Tugas Akhir di departemen Informatika ITS.

5. Bapak Darlis Herumurti, S.Kom., M.Kom. selaku kepala departemen Informatika ITS.
6. Seluruh dosen dan karyawan Teknik Informatika ITS yang telah memberikan ilmu dan pengalaman kepada penulis selama menjalani masa studi di ITS.
7. Ibu Eva Mursidah dan Ibu Sri Budiati yang selalu mempermudah penulis dalam peminjaman buku di RBTC.
8. Seluruh rekan – rekan TC 2014 yang sudah mendukung penulis selama perkuliahan.
9. Teman-teman seperjuangan RMK NCC/KBJ, yang telah menemani dan menyemangati penulis.
10. Rekan – rekan kontrakan E-103 Widhi, Dharmawan, Fintanto, Fahmi, dan Hanendyo yang sudah memberi dukungan dan motivasi kepada penulis.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki banyak kekurangan sehingga dengan kerendahan hati penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan ke depan.

Surabaya, Mei 2018

DAFTAR ISI

(Halaman ini sengaja dikosongkan)

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

DAFTAR GAMBAR

(Halaman ini sengaja dikosongkan)

DAFTAR KODE SUMBER

(Halaman ini sengaja dikosongkan)

BAB I

PENDAHULUAN

Pada bab ini akan dipaparkan mengenai garis besar Tugas Akhir yang meliputi latar belakang, tujuan, rumusan dan batasan permasalahan, metodologi pembuatan Tugas Akhir, dan sistematika penulisan.

1.1 Latar Belakang

Perkembangan teknologi komunikasi saat ini telah berkembang dengan sangat pesat, salah satunya dibidang *Wireless Sensor Network* (WSN). WSN adalah sebuah jaringan yang terdiri dari *node* - *node* yang didistribusikan baik secara strategis 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. [1]

Modul *Wireless* nRF24L01 adalah sebuah modul komunikasi jarak jauh yang memanfaatkan pita gelombang RF 2.4GHz ISM (*Industrial, Scientific and Medical*) sebagai media komunikasinya. Modul ini menggunakan antarmuka 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 hanya ditenagai daya baterai yang terbatas. [2]

Pada jaringan sensor yang akan dibuat, diharapkan semua *node* dapat mengirimkan data ke sebuah *server / coordinator*. Namun, apabila semua node mengirim langsung ke *server / coordinator*, maka penggunaan energi dalam jaringan akan menjadi boros dikarenakan semua *node* akan menggunakan kekuatan transmisi tinggi untuk mengirim data sampai *server* atau koordinator. Sehingga, perlu adanya node yang menjadi

cluster head dalam jaringan untuk mengumpulkan data dari *node* lain sebelum mengirimkannya ke *server / 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. Dikarenakan besarnya penggunaan energi tersebut, maka diperlukan sebuah metode pemilihan *cluster head* yang efisien untuk menangani permasalahan tersebut.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini adalah sebagai berikut:

1. Bagaimana membangun sistem pemilihan *cluster head*, koordinasi antar *node*, dan pemilihan kekuatan transmisi secara dinamis menggunakan modul nRF24L01?
2. Bagaimana menerapkan metode pemilihan *cluster head* dan pengaturan kekuatan transmisi secara dinamis pada jaringan sensor yang dapat meminimalkan pemakaian energi?
3. Bagaimana menerapkan *time slot* untuk untuk meningkatkan efisiensi pengiriman paket?

1.3 Batasan Permasalahan

Permasalahan yang dibahas pada tugas akhir ini memiliki batasan sebagai berikut:

1. Jaringan yang digunakan adalah jaringan sensor nirkabel menggunakan modul *wireless* nRF24L01.
2. Jumlah *node* yang digunakan sebanyak 7 dengan 1 sebagai *server / coordinator* dan 6 sebagai *end device / cluster head* yang dipilih secara bergantian.
3. Mikrokontroler yang digunakan adalah Arduino UNO R3.

1.4 Tujuan

Tujuan pembuatan tugas akhir ini adalah sebagai berikut:

1. Membangun sebuah jaringan sensor nirkabel yang dapat berkoordinasi secara dinamis dalam pemilihan *cluster head* yang dapat menyeimbangkan penggunaan energi dalam jaringan tersebut.
2. Penerapan *scheduling* transmisi setiap *node* dengan *time slot* untuk meningkatkan efektifitas transmisi setiap *node*.

1.5 Manfaat

Hasil dari pengerjaan Tugas Akhir ini memiliki manfaat untuk menghasilkan sebuah jaringan sensor nirkabel yang dapat berkoordinasi secara dinamis dan 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

Tahap awal tugas akhir ini adalah menyusun proposal tugas akhir. Proposal Tugas Akhir ini berisi tentang perencanaan penerapan metode pemilihan *cluster head* sebagai solusi optimal dari permasalahan dalam pemilihan *cluster head* dan penyeimbangan energy dalam jaringan sensor nirkabel.

1.6.2 Studi Literatur

Studi literatur yang dilakukan dalam pengerjaan Tugas Akhir ini adalah mengenai implementasi metode *cluster head*

selection pada jaringan sensor nirkabel untuk menentukan node yang akan dijadikan *cluster head* dalam jaringan. Selain itu, juga dilakukan studi literatur mengenai implementasi pemilihan cluster head menggunakan energi *node* sebagai acuan. Sehingga, studi literatur ini dapat diterapkan pada perancangan jaringan sensor nirkabel yang dapat melakukan pemilihan *cluster head* secara dinamis.

1.6.3 Implementasi Metode

Implementasi metode yang dipakai dalam tahapan membangun jaringan ini adalah untuk memilih *cluster head* dari *node-node* yang ada dengan menggunakan tingkat energi dari masing-masing node sebagai variabel penentuan. Jika energi *node cluster head*, sudah mencapai batas minimum energi, maka akan dilakukan pemilihan *cluster head* yang baru. Kemudian jika semua node memiliki energi dibawah variabel minimum, maka nilai variabel akan diturunkan.

1.6.4 Pengujian dan Evaluasi

Pengujian dan evaluasi dari hasil Tugas Akhir ini akan diujicobakan dengan membangun jaringan sensor nirkabel menggunakan modul *wireless* nRF24L01 yang berlokasi di jurusan Teknik Informatika Institut Teknologi Sepuluh Nopember Surabaya.

1.6.5 Penyusunan Buku

Pada tahap ini disusun buku sebagai dokumentasi dari pelaksanaan tugas akhir yang mencangkup 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 berisikan penjelasan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika penulisan dari pembuatan tugas akhir.
2. Bab II. Tinjauan Pustaka
   Bab ini meliputi dasar teori dan penunjang yang berkaitan dengan pokok pembahasan dan mendasari pembuatan Tugas Akhir ini.
3. Bab III. Perancangan Perangkat Lunak dan Perangkat Keras
   Bab ini membahas desain dari jaringan yang akan dibuat meliputi arsitektur dan proses perangkat lunak dan keras.
4. Bab IV. Implementasi
   Bab ini membahas implementasi dari desain jaringan yang akan dilakukan pada tahap desain, meliputi potongan *pseudocode* yang terdapat dalam perangkat lunak dan perangkat keras yang digunakan.
5. Bab V. Hasil Uji Coba dan Evaluasi
   Bab ini membahas uji coba dari jaringan yang dibuat dengan melihat keluaran yang dihasilkan, analisa dan evaluasi untuk mengetahui kemampuan jaringan.
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 solusi ke depannya.
7. Daftar Pustaka
   Bab ini berisi daftar pustaka yang dijadikan literatur.

(Halaman ini sengaja dikosongkan)

BAB II

TINJAUAN PUSTAKA

2.1 Komunikasi Nirkabel

Wireless communication atau komunikasi nirkabel merupakan komunikasi yang terjadi antara dua *devices* yang disebut dengan *transmitter* dan *receiver* tanpa menggunakan perantara kabel.[1]

Transmitter pada komunikasi nirkabel berfungsi sebagai pengirim pesan informasi sedangkan *receiver* berfungsi sebagai penerima dari pesan informasi. *Transmitter* dan *receiver* pada perangkat *wireless* umumnya sudah terintegrasi menjadi satu perangkat. Sehingga perangkat tersebut dapat bertindak sebagai pengirim atau penerima pesan.[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 mempunyai 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 meneriam seluruh transmisi dalam 1 waktu, sehingga dibutuhkan sebuah penjadwalan transmisi agar tidak terjadi tabrakan transmisi.

2.2 Jaringan Sensor Nirkabel *(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.[3]

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 energi 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*. [3]

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*.[4]

Melakukan *multihop* atau tranmisi jauh secara masif tanpa adanya organisasi tentu akan menguras energi sangat besar yang malah membuat *lifespan* daripada 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 member* 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 matinya jaringan sensor.[4]

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 member* untuk mengganti *cluster head* dengan yang baru dari *node* lain.

2.5 *Time Slot*

Time Slot adalah paket yang berisi waktu pengiriman transmisi setiap *node*. *Time Slot* dikirimkan oleh *cluster head* kepada setiap *node* untuk membuat penjadwalan transmisi setiap

node. *Time slot* berguna agar tidak terjadi tabrakan transmisi antara *node*.

2.6 Penggunaan Variabel Energi dalam pemilihan *Cluster Head*

Variabel energi digunakan sebagai batas syarat minimum sebuah *node* dapat dijadikan sebagai *cluster head*. Apabila sebuah *node* yang terpilih menjadi *cluster head* memiliki tingkat energi dibawah variabel yang ditentukan, maka proses pemilihan *cluster head* baru akan dilakukan. Proses pemilihan dilakukan dengan membandingkan energi pada node calon *cluster head* dengan variabel energi, kemudian dipilih *node* dengan energi terbanyak.[4]

Ada kalanya metode LEACH tidak menjamin terpilihnya hanya satu *cluster head* sehingga terjadinya kegagalan koordinasi pada *cluster* yang disebabkan oleh terpilihnya lebih dari satu *cluster head*.

2.7 LEACH (Low-Energy Adaptive Clustering Hierarchy)

LEACH (Low-Energy Adaptive Clustering Hierarchy) LEACH adalah sebuah protokol *clustering* yang menggunakan pengacakan pergantian *cluster head* lokal untuk pemerataan pemakaian energi dalam jaringan sensor. LEACH menggunakan koordinasi lokal dalam satu *cluster* untuk memungkinkan jaringan yang mempunyai skalabilitas dan ketahanan yang baik pada jaringan dinamis. LEACH terdiri dari dua fase yaitu *setup phase* saat *cluster* meorganisir tiap node untuk menentukan yang menjadi *cluster head*, dan *steady-state phase* saat cluster head telah terpilih dan transmisi data. [4]

Saat awal *cluster* dibuat *cluster* akan mencari *node* yang akan menjadi *cluster head*. Setelah *cluster head* terpilih *cluster member* akan mengirim alamat dirinya kembali ke *cluster head*

untuk memberitahu bahwa dirinya termasuk dalam *cluster* yang sama dengan *cluster head*. Dalam fase ini *cluster head* harus tetap masuk ke mode *listening*.

Setelah *cluster head* menerima pesan semua *cluster member* yang akan menjadi *cluster*nya *cluster head* akan membuat *Time Slot* untuk penjadwalan transmisi tiap *node*. Pembuatan *time slot* dibuat cukup untuk semua *cluster* melakukan transmisi dalam satu detik agar transmisi data terlalu terlambat. Setelah *Time Slot* dibuat maka *cluster head* akan mengirim *time Slot* ke masing-masing *cluster member*.

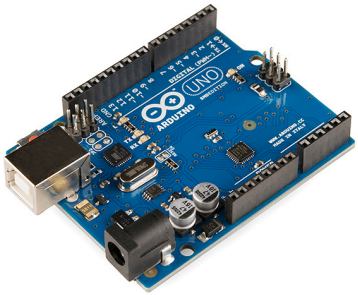
Setelah *cluster* terbentuk dan penjadwalan tiap *node* telah dibuat *cluster member* akan mulai melakukan transmisi data ke *cluster head*. *Cluster member* akan memakai daya transmisi rendah untuk pengiriman data ke *cluster head* karena jarak *cluster head* dekat sehingga tidak diperlukan daya besar untuk melakukan transmisi. *cluster member* hanya melakukan transmisi pada waktu yang telah ditentukan untuknya, selain itu node akan mematikan *transmitter* untuk menghemat daya. Sedangkan *cluster head* harus menggunakan daya transmisi tinggi untuk menjangkau *sink* yang letaknya jauh, selain itu *cluster head* harus tetap dalam mode *listening* untuk mendapatkan data dari *cluster member*.

2.8 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.[5]

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 program pada board.[5]



Gambar 2.1: Arduino Uno R3

2.9 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.[2]

Modul nRF24l01 dapat bekerja sebagai *transmitter* dan juga sebagai *receiver*. Namun nRF tidak bisa menjadi *transmitter* dan *receiver* dalam waktu yang bersamaan. Perlu adanya pergantian mode bila membutuhkan komunikasi dua arah antar node nRF. Selain itu melakukan pergantian mode dengan timing yang pas diperlukan karena paket yang datang akan drop bila modul dalam mode transmisi saat bersamaan.

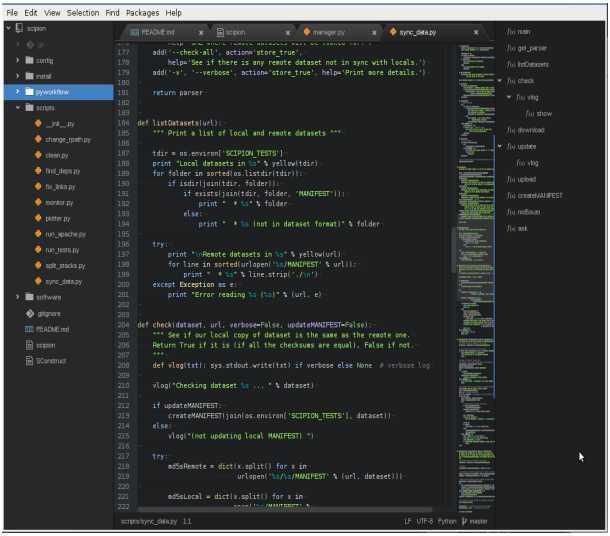
Diluar dari kelemahan tersebut modul nRF24L01 memiliki kelebihan dalam kehandalan. portabilitas, dan pemakaian daya. Pemakaian daya yang rendah dan dimensi yang kecil membuat modul ini banyak dipakai dalam biadang yang memerlukan komunikasi nirkabel tidak terkecuali jaringan sensor nirkabel.

Gambar 2.2: Modul nRf24L01

2.10 Atom

Atom adalah *code editor* gratis dan *open source* untuk sistem operasi macOS, Linux, dan Microsoft Windows dengan dukungan *plug-ins* dan *embedded Git Control*. Atom merupakan aplikasi desktop yang dibangun menggunakan teknologi web. Sama seperti Atom, paket extensinya juga menggunakan *free software licences* sehingga siapapun boleh menggunakannya dengan bebas dan di maintenen oleh komunitas. [6]

Atom mendukung banyak bahasa pemrograman seperti C/C++, Java, Python, Javascript, dan lainnya yang dapat ditambahkan melalui paket *plugin* yang di tulis dengan NodeJs. Karena fleksibilitas, portabilitasnya maka Atom sering dipakai untuk editor perangkat lunak *hardware* mikrokontroler termasuk Arduino menggunakan plug-ins PlatformIO.



Gambar 2.3: Atom Text Editor

2.11 Arduino Uno Revision 3

Arduino Uno adalah salah satu produk berlabel Arduino yang sebenarnya adalah suatu papan elektronik yang mengandung mikrokontroler ATmega328, sebuah keping yang secara fungsional bertindak seperti sebuah komputer. Piranti ini dapat dimanfaatkan untuk mewujudkan rangkaian elektronik dari yang sederhana hingga yang kompleks. Pengendalian LED hingga pengendalian robot dapat diimplementasikan dengan menggunakan papan yang berukuran cukup kecil ini. Bahkan dengan penambahan komponen tertentu, piranti ini bisa digunakan untuk pemantauan jarak jauh melalui internet. [5]

2.12 Arduino IDE

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

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

2.13 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 *embeded 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.

BAB III

PERANCANGAN SISTEM

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 metode *cluster head* pada sebuah jaringan sensor nirkabel yang dibangun menggunakan Arduino sebagai *node*nya dan menggunakan modul *Wireless* nRF24L01 untuk berkomunikasi antar *node*.

3.1 Deskripsi Umum

Jaringan sensor nirkabel menggunakan mikrokontroler Arduino dan modul nRF24L01. Terdapat 7 buah *node* yang masing - masing berisi modul nRF24L01. Satu node akan menjadi *server / coordinator* yang menerima data dari *node cluster* dan 6 node lain akan menjadi *cluster node*. *Node coordinator* akan terhubung dengan komputer dan mendapat suplai daya tetap, sedangkan *node cluster* akan ditenagai oleh baterai.

3.2 LEACH (Low-Energy Adaptive Clustering Hierarchy)

LEACH adalah sebuah protokol *clustering* yang menggunakan pengacakan pergantian *cluster head* lokal untuk pemerataan pemakaian energi dalam jaringan sensor. LEACH menggunakan koordinasi lokal dalam satu *cluster* untuk memungkinkan jaringan yang mempunyai skalabilitas dan ketahanan yang baik pada jaringan dinamis. LEACH terdiri dari dua phase yaitu *setup phase* saat *cluster* meorganisir tiap node untuk menetukan yang menjadi *cluster head*, dan *steady-state phase* saat *cluster head* telah terpilih dan transmisi data.

Saat awal *cluster* dibuat *cluster* akan mencari *node* yang akan menjadi *cluster head*. Setelah *cluster head* terpilih *cluster*

node akan mengirim alamat dirinya kembali ke *cluster head* untuk memberitahu bahwa dirinya termasuk dalam cluster yang sama dengan *cluster head*. Dalam fase ini *cluster head* harus tetap masuk ke mode *listening*.

3.2.1 Cluster Node Scheduling

Setelah *cluster head* menerima pesan semua *cluster node* yang akan menjadi *cluster*nya *cluster head* akan membuat *Time Slot* untuk penjadwalan transmisi tiap *node*. Pembuatan *time slot* dibuat cukup untuk semua *cluster* melakukan transmisi dalam satu detik agar transmisi data terlalu terlambat. Setelah *Time Slot* dibuat maka *cluster head* akan mengirim *time Slot* ke masing-masing *cluster node*.

Scheduling digunakan karena nRF tidak bisa menjadi *transmitter* dan *receiver* bersamaan, sehingga *scheduling* dibuat agar transmisi node tidak saling bertabrakan satu sama lain.

3.2.2 Setup Phase

Dalam fase ini setiap *node* akan menentukan apakah dirinya akan menjadi *cluster head* berdasarkan dari persentase prioritas tiap node menjadi *cluster head*, persentase prioritas telah ditentukan saat *node* di program. Penentuan menjadi *cluster head* atau tidak dengan cara mencari bilangan acak desimal antara 0 sampai 1. Bilangan acak tersebut apabila lebih kecil dari *threshold T(n)*, node tersebut menjadi *cluster head*. *Cluster head* yang terpilih akan melakukan *broadcast* ke node lainnya untuk memberitahu bahwa dialah *cluster node* saat itu. *Threshold* T(n) dicari menggunakan persamaan berikut:

$$T(n) = \begin{cases} \frac{P}{1-P*(rmod\frac{1}{P})} & n \in G \\ 0 & otherwise \end{cases}$$

Dimana *P* merupakan persentase *cluster head*, *r* adalah jumlah putaran saat ini, dan *G* adalah himpunan node yang belum terpilih jadi *cluster head*.

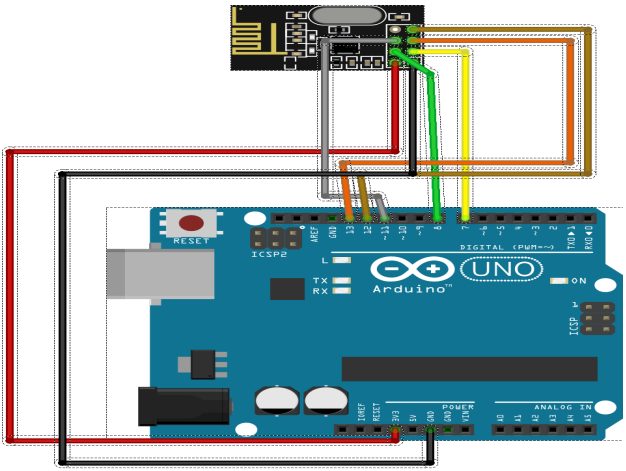
Terkadang ada kalanya *cluster head* yang terpilih lebih dari 1 node dalam 1 *cluster*, dalam tugas akhir ini *cluster head* yang terpilih akan membandingkan jumlah energi mereka dan *node* yang menjadi *cluster head* ditentukan dari energi yang paling besar.

3.2.3 Steady-state Phase

Fase ini merupakan fase dimana *cluster* memulai melakukan transmisi. *Cluster head* akan menentukan jadwal transmisi kepada setiap *node* pada *cluster* guna menghindari terjadinya tabrakan transmisi antar *node*. *Cluster head* akan terus melakukan *listening* dan mengirim semua transmisi *cluster* ke *coordinator*. Pada fase ini *cluster head* akan mengatur kekuatan transmisi penuh untuk menjangkau transmisi ke koordinator. Sedangkan untuk node lain mengatur kekuatan transmisi minimal untuk menghemat energi. Dalam fase ini energi *cluster head* akan cepat habis dikarenakan menggunakan transmisi penuh. Pergantian *cluster head* akan dilakukan bila penurunan energi baterai *cluster head* sudah melebihi batas atau waktu *cluster head* yang telah ditentukan habis dan cluster akan kembali lagi ke *setup phase*.

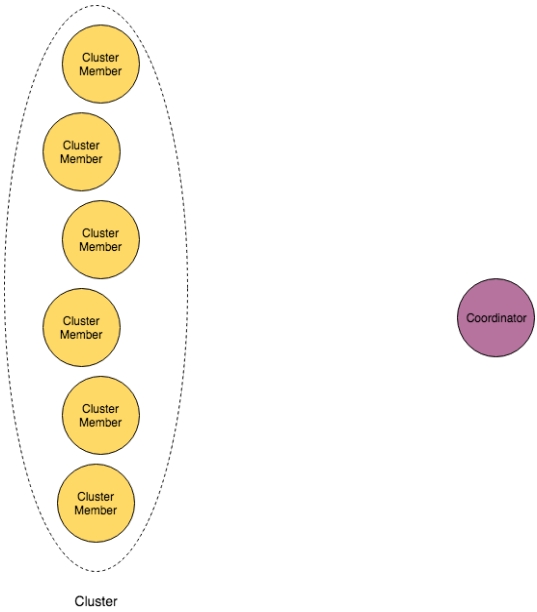
3.3 Arsitektur Sistem

Sub bab ini akan dijelaskan mengenai arsitektur umum sistem yang akan dibangun. Sistem ini terdiri dari 6 Arduino sebagai *cluster member* dan 1 Arduino sebagai *coordinator*, menggunakan *wireless* modul nRF24L01. Modul nRF24L01 menggunakan SPI sebagai *interface* komunikasi dengan Arduino ditunjukan pada Gambar 3.1.



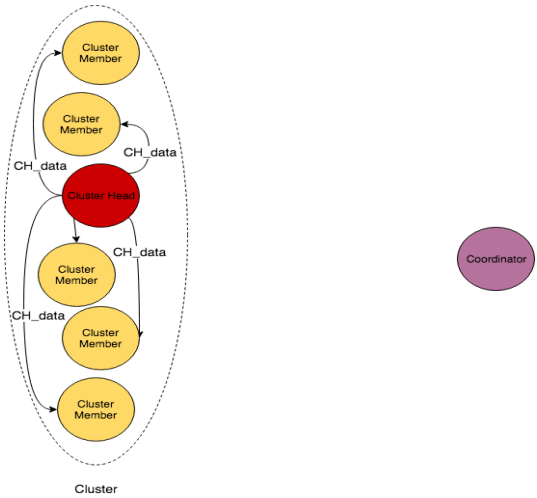
Gambar 3.1: Arduino dan nRF24L01

Setiap cluster node akan diimplemetasikan pemilihan Cluster Head sehingga setiap node dapat bergantian menjadi Cluster Head. Selanjutnya *cluster member* dan *coordinator* ditempatkan terpisah dengan jarak minimum 50 meter. Setelah *cluster* hidup setiap *node cluster* akan masuk ke Setup phase dan menetukan apakah dirinya menjadi *cluster head*.

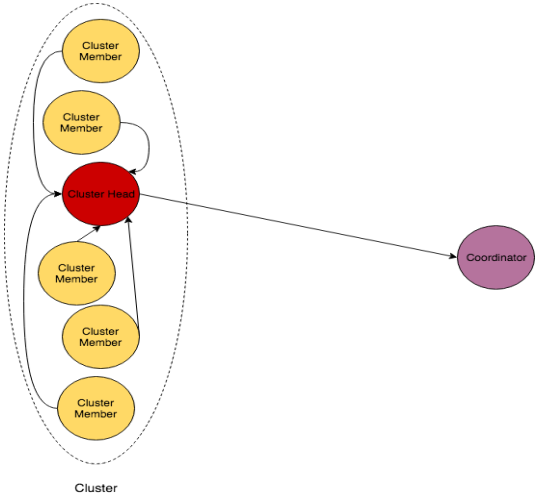


Gambar 3.2: Kondisi awal *cluster*

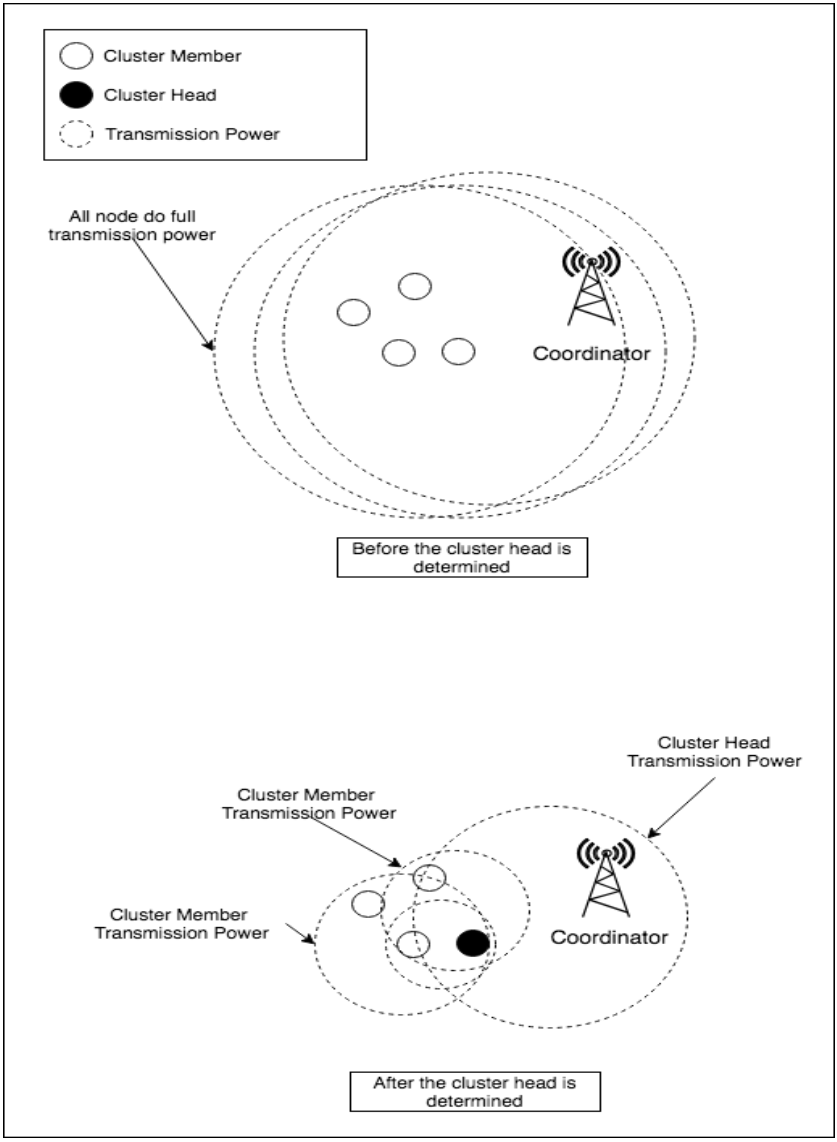
Cluster head yang terpilih akan mengirimkan paket CHDATA ke setiap *node*. Paket CHDATA berisi alamat *cluster head*. Selanjutnya setiap *node* akan mengirimkan data *sensing* mereka ke *cluster head* dan menurunkan kekuatan transmisi menjadi minimal. Selanjutnya *cluster head* menaikan kekuatan transmisinya menjadi maksimal dan mengirim paket-paket yang diterimanya ke *coordinator*. Bila *cluster head* yang terpilih lebih dari satu maka node yang mempunyai sisa energi terbesar yang akan dijadikan *cluster head*. Gambaran perubahan kekuatan transmisi *node* dapat dilihat pada Gambar 3.5.



Gambar 3.3: *Cluster head* terpilih mengirim paket CHDATA ke setiap node *cluster*

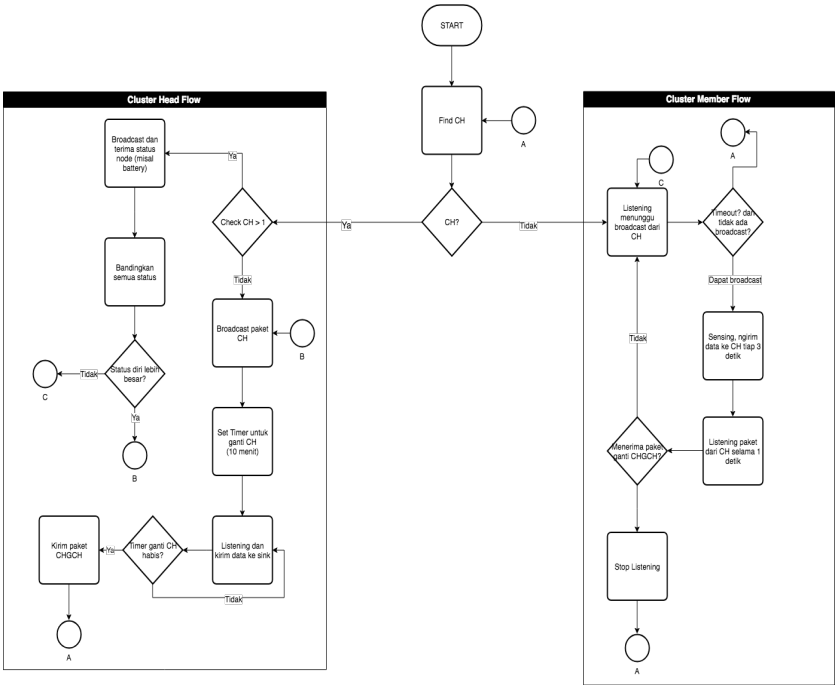


Gambar 3.4: *Cluster head* meneruskan paket dari *cluster* node



Gambar 3.5: Perubahan kekuatan transmisi node *cluster* sebelum dan sesudah pemilihan *cluster head*

Cluster member saat tidak melakukan transmisi akan *listening* kemungkinan adanya paket CHGCH yang dikirim oleh *cluster head*. Paket CHGCH adalah paket yang mengintruksikan *cluster member* untuk mengganti *cluster head*. Paket CHGCH akan di *broadcast* oleh *cluster head* bila penurunan energi *cluster head* melewati batas atau batas waktu *cluster head* telah habis. Untuk lebih jelasnya dapat melihat Gambar 3.6.



Gambar 3.6: LEACH based cluster head selection flowchart

BAB IV

IMPLEMENTASI

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

4.1 Lingkungan Implementasi

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

4.1.1 Lingkungan Implementasi Perangkat Keras

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

Tabel 4.1: Lingkungan Implementasi Perangkat Keras

Perangkat	**Detail**
Perangkat Mikrokontroler	**Mikrokontroler:** • Atmega 328

Tabel 4.1: Lingkungan Implementasi Perangkat Keras

Perangkat	Detail
	Model: • Arduino UNO R3 **Tegangan:** • 5 - 12 V **Memory Flash:** • 32 KB **SRAM:** • 2 KB
Perangkat Wireless Transceiver	**Model:** • nRF24L01+ Single Chip 2.4GHz Transceiver **Manufaktur:** • Nordic Semiconductor **Tegangan:** • 1.9 - 3.6 V **Frekuensi:** • 2.4 GHz ISM Band **Interface:** • SPI **Ukuran:** • 20-pin 4x4 QFN Package

4.2 Lingkungan Implementasi Perangkat Lunak

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

implementasi perangkat lunak dari sistem yang akan dibangun secara lebih lengkap dijelaskan pada Tabel 4.2 di bawah ini.

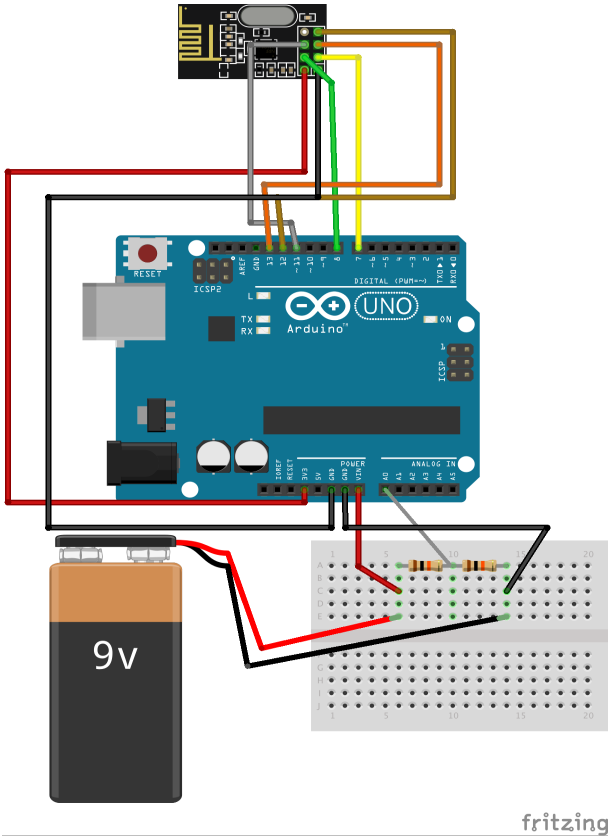
Tabel 4.2: Lingkungan Implementasi Perangkat Lunak

Perangkat Lunak	Detail
Atom	*Text editor* yang digunakan untuk menulis *code* program Arduino.
PlatformIO	PlatformIO berisi *toolchain* yang digunakan untuk melakukan kompilasi kode program bahasa C untuk mikrokontroler Arduino dan mikrokontroler lainnya.
nRF24 Library	*Librar*y 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.
CountUpDownTimer Library	*Library* yang digunakan untuk membuat penghitung waktu mundur. Library ini digunakan karena Arduino tidak mempunyai IC untuk penghitung waktu sendiri. *Library* ini memanfaatkan pencacah pada Arduino yang menghitung berapa milidetik Arduino berjalan.

4.3 Implementasi *Cluster Member*

Pada bagian ini dijelaskan implementasi dari setiap *node cluster* yang digunakan. Setiap *cluster member* Arduino

dihubungkan dengan modul *wireless* nRF24L01 melalui *interface* SPI. Baterai 9 volt digunakan untuk mentenagai node. Untuk lebih jelas dapat melihat Gambar 4.1.



Gambar 4.1: Rangkaian *cluster member*

Kutub positif baterai dihubungkan ke pin *Vin* Arduino dan kutub negatif ke pin *GND* untuk menyuplai daya. Kutub positif dan negatif baterai disambung ke pin A0 dan di berikan resistor 10kOhm yang membuat sedikit aliran listrik baterai mengalir ke

pin A0, melalui pin A0 Aduino dapat membaca kapasitas baterai.

4.4 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.

4.4.1 Global Variabel dan Type

Subbab ini membahas variabel dan type data yang akan dibuat dalam menunjang fungsi-fungsi lain dalam program.

```
n <- 2
r <- 1
p <- 0.48

broadcast_addr <- 1000
self_addr <- <Node nrf Address diffrent in
    every node>

sink_addr <- 3003
change_ch_message = "CNGNH"

flagCH <- false
flag_ch_found <- false

packet_send <- 0

packet_id_counter <- 0
```

```
random_number <- Randomize number beetween
    1 to 100

structure data_CH:
     CH_addr
     status

structure dataSt:
     addr
     msg
     id_packet

ch_lain <- new data_CH
ch_now <- new data_CH

gmsg <- new dataSt

T <- new CountUpDownTimer(DOWN)
T1 <- new CountUpDownTimer(UP,HIGH)

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

Kode Sumber IV.1: Global Variabel dan Type

Sesuai *code* diatas program akan mendeklarasikan beberapa *address* yang akan digunakan dalam transmisi. Seperti *broadcast address* yang digunakan oleh semua *node* untuk transmisi *broadcast*, *self address* untuk *node* itu sendiri, dan *sink address* alamat *coordinator*. Program juga menggunakan 2 buah *timer* T dan T1 untuk melakukan penghitugan waktu. Variabel radio digunakan untuk mengontrol kerja dari nRF24l01.

4.4.2 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))

    packet_send <- packet_send + 1
```

Kode Sumber IV.2: Fungsi Transmisi

Sebelum melakukan pengiriman nRF diinstruksikan untuk berhenti melakukan *listening* dikarenakan nRF tidak bisa melakukan transmisi dan *listening* bersamaan. Selanjutnya nRF diberikan alamat node yang akan di kirimkan transmisi dan fungsi *write* menginstrusikan nRF melakukan transmisi. Di dalam fungsi juga terdapat *counter* yang digunakan untuk mencatat berapa banyak paket yang pernah ditransmisikan.

4.4.3 Fungsi *Listening*

Fungsi berikut digunakan untuk mendengar dan membaca transmisi yang datang ke node.

```
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 IV.3: Fungsi *Listening*

Sebelum mendengar transmisi sebuah *pipe* akan dibuka untuk masuknya transmisi dilanjutkan dengan memanggil *startListening* untuk memulai mendengar transmisi dari *node* lain. Selanjutnya program akan mengekek apakah ada transmisi masuk, bila iya maka transmisi akan dibaca datanya dan fungsi mengmbalikan nilai *True* bila tidak maka tidak dilakukan pembacaan dan fungsi membalikan nilai *False*.

4.4.4 Fungsi Cluster Head

Fungsi ini adalah fungsi perhitungan yang didasarkan pada rumus LEACH untuk mencari apakah *node* tersebut akan menjadi *cluster head* pada putaran tersebut.

```
function findT():
    phase1 <- r * fmod(1, p)
    phase2 <- 1 - p * phase 1
    t = p / phase2

    return t

function getRandom():
    call randomSeed(analogRead(A0))
    return random(1,100) / 100.0

function findCH():
    if r < n:
        r = 1
```

```
t <- call findT()
ran <- call getRandom()

if ran < t:
  return True
else:
  return False
```

Kode Sumber IV.4: Fungsi Penentuan Cluster Head

4.4.5 Fungsi Wait for CH

Fungsi ini digunakan untuk *node* menunggu transmisi dari *cluster head* yang terpilih pada *setup phase*. Fungsi ini dibuat agar *node* yang tidak menjadi calon *cluster head* menunggu *cluster head* untuk memberi intruksi transmisi.

```
call setTimer(0, 0, 30)
ch_now.status <- 0

while T.Seconds <> 0:
    if radio_listening(broadcast_addr,
       ch_lain):
        if ch_lain.status = -1:
            ch_now.CH_addr <- ch_lain.
               CH_addr
            ch_now.status <- ch_lain.status

            flag_ch_found <- True

            break
```

Kode Sumber IV.5: Fungsi Wait for CH

Saat fungsi ini dijalankan node akan melakukan *listening* selama 30 detik. Jika ada terdapat transmisi maka *node* akan mengecek variabel status dalam transmisi. Jika status -1 artinya *cluster head* telah ditentukan dan node akan menyimpan alamat *cluster head* yang baru dan keluar dari *loop*.

4.4.6 Fungsi Setup Phase

Fungsi ini adalah fungsi persiapan *cluster member* sebelum melakukan transmisi data. Fungsi ini dimulai dari pencarian kandidat *cluster head*, memutuskan node yang menjadi *cluster head* bila lebih dari satu *cluster head* terpilih. Fungsi ini akan dipanggil pertama kali pada setiap putaran.

```
function setup_phase():
    is_iam_ch <- call findCH()

    if is_iam_ch = True:
        flagCH <- True
        data <- new data_CH
        data.CH_addr <- self_addr
        data.status <- packet_send

        call setTimer(0,0,10)

        while T.Seconds <> 0:
            if T.Seconds mod 2 = 0:
                call radio_send(
                    broadcast_addr, &data)

                if T.Seconds mod 3 = 0:
                  if radio_listening(
                    broadcast_addr,
```

```
                    ch_lain) = true:
                     if ch_lain > data:
                         flagCH <– False

                             break

                       else:
                         flagCh <– True

else is_iam_ch = False:
   call wait_for_ch()

if flagCH = True:
   call tell_i_am_ch()
```

Kode Sumber IV.6: Fungsi Setup Phase

Saat *node* berada dalam *setup phase node* akan memutuskan apakah dirinya akan jadi *cluster head* di putaran ini atau tidak menggunakan metode LEACH. Jika ternyata *node* menjadi *cluster head* node tidak langsung *broadcast* bahwa dirinya adalah *cluster head* baru, namun melakukan *listening* selama 10 detik untuk kemungkinan adanya *cluster head* yang terpilih lebih dari 1. Variabel flagCH adalah *state* apakah dirinya akan menjadi *cluster head* atau tidak. Jika setelah 10 detik flagCH bernilai *True* maka *node* tersebut akan mengumumkan bahwa dirinya adalah *cluster head* putaran ini. Bagi *node-node* yang tidak akan menjadi *cluster head* hanya menunggu ada salah satu *node* yang menjadi *cluster head*.

4.4.7 Fungsi Steady Phase

Fungsi ini dimana *node* melakukan transmisi data ke *cluster head*. Fungsi ini akan dijalankan jika saat *setup phase* sudah terdapat *cluster head yang terpilih*.

```
function steady_phase():
    if flagCH = True:
        call radio.setPALevel(HIGH)
        call setTimer(0,1,0)

        msg <- new dataSt

        while T.Seconds <> 0:
            if radio_listening(self_addr,
               msg, call sizeof(msg)):
                call radio_send(sink_addr,
                   msg)

        call setTimer(0,0,1)

        while T.Seconds <>0 :
            call radio_send(broadcast_addr,
                change_ch_message)

    else:
        call radio.setPALevel(MIN)

        call T1.startTimer()

        while True:
            if radio_listening(
               broadcast_addr, msg):
                if msg = change_ch_message:
                    break

            if T1.Seconds mod 3 = 0:
```

```
                call radio_send(ch_now.addr
                    , sensordata)
```

Kode Sumber IV.7: Fungsi Steady Phase

4.4.8 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 semasa Arduino hidup.

```
function setup():
    call radio.begin()

    gmsg.addr <- self_addr

    random_number <- call getRandom()

    call setup_phase()

    while flag_ch_found = False and flagCH
       = False:
        call setup_phase()

    if flagCH = True:
        call tell_iam_ch()
```

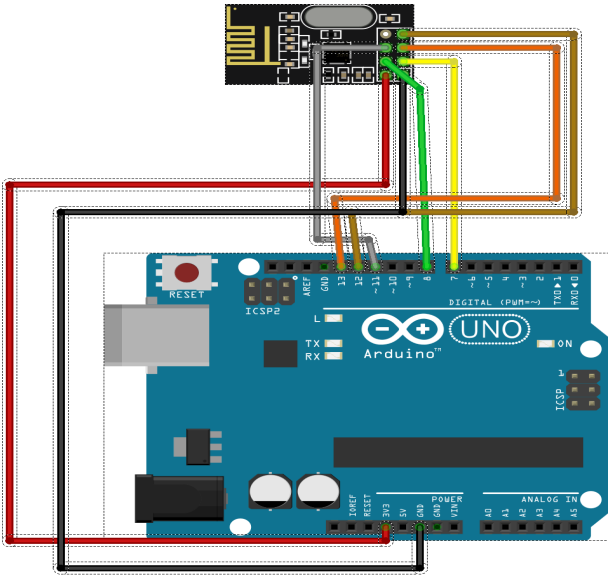
Kode Sumber IV.8: Fungsi Setup

Saat Arduino baru menyala dan fungsi ini di eksekusi maka Arduino akan megaktifkan radio nRF terlebih dahulu dan langsung memasuki *setup phase*. Ketika *setup phase* berakhir karena *cluster head* sudah ditemukan atau *timeout* Arduino akan

memeriksa apakah dirinya adalah *cluster head* putaran ini atau *cluster head* telah terpilih. Jika dirinya bukanlah *cluster head* terpilih atau belum ada *cluster head* yang terpilih maka Aruino akan kembali ke *setup phase* sampai salah satu kondisi terpenuhi.

4.5 Implementasi *Coordinator*

Pada bagian ini dijelaskan implementasi dari *coordinator*.*coordinator* hanya melakukan *listening* dari *cluster node*. Rangkaian *coordinator* mirip seperti rangkaian pada *cluster member* dimana Arduino dihubungkan dengan modul *wireless* nRF24L01 melalui *interface* SPI, namun tanpa memakai baterai. Rangkaian akan terlihat seperti pada Gambar 4.2.



Gambar 4.2: Rangkaian *coordinator*

4.6 Implementasi Code Program *Coordinator*

Pada bab ini akan di implementasikan code program *coordinator*. Dalam implementasi code program dalam 2 bagian yaitu bagian *setup* dan bagian *loop*. Setiap implementasi *code* program akan dibahas dalam subbab berikut.

4.6.1 Global Variabel dan Type

Subbab ini membahas variabel dan type data yang akan dibuat dalam menunjang fungsi-fungsi lain dalam program.

```
sink_addr <- 3003

radio <- new RF24(7, 8)

structure dataSt:
    addr
    msg
    id_packet
```

Kode Sumber IV.9: Global Variabel dan Type

4.6.2 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 semasa Arduino hidup.

```
function setup():
    data <- new dataSt
```

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

Kode Sumber IV.10: Fungsi Setup Coordinator

Fungsi ini hanya bertujuan untuk inisialisasi variabel dan nRF. Karena *coordinator* hanya menunggu data maka Arduino di set untuk melakukan *listening* sejak dia dihidupkan.

4.6.3 Fungsi Loop

Fungsi ini adalah fungsi yang akan selalu dijalankan selama arduino beroperasi. Fungsi loop merupakan fungsi utama dari Arduino. Semua program dan intruksi Arduino akan dijalankan dalam fungsi ini. Disini *coordinator* akan menunggu transmisi dan mencetak isi transmisi.

```
function loop():
    data <- new dataSt

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

        print data.addr
        print data.id_packet
```

Kode Sumber IV.11: Fungsi Loop Coorinator

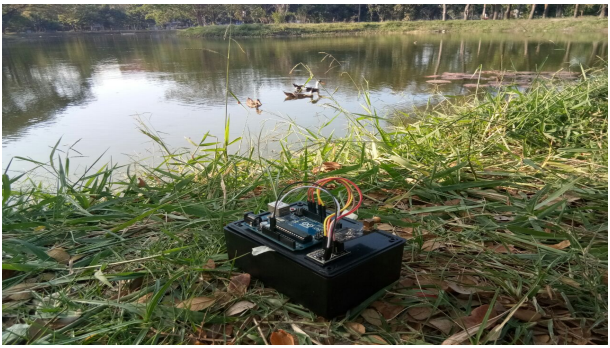
BAB V

HASIL UJI COBA DAN EVALUASI

Bab ini berisi penjelasan mengenai skenario uji coba dan evaluasi pemilihan *cluster head* dan pemilihan daya transmisi secara dinamis. Hasil uji coba didapatkan dari implementasi pada Bab 4 dengan skenario yang berbeda. Bab ini berisikan pembahasan mengenai lingkungan pengujian, data pengujian, dan uji kinerja.

5.1 Lingkungan pengujian

Setiap pengujian skenario dilakukan di kolam delapan yang bertempatan di area dalam ITS, menaruh tiap *node cluster* diatas tanah dan ditenagai oleh baterai, jarak antar cluster dan *coordinator* kurang lebih 50 meter menyeberangi kolam. Pengujian dilakukan siang hari dan *node cluster* ditaruh di tempat yang cukup teduh untuk menghindari terjadinya kerusakan akibat panas dari matahari. Untuk menghindari kerusakan yang diakibatkan debu maka tiap node di rangkai diatas sebuah circuit box.



Gambar 5.1: Lingkungan pengujian node 1

Gambar 5.2: Lingkungan pengujian node 2

Gambar 5.3: Lingkungan pengujian node 3

Seluruh node akan memakai baterai yang sama yaitu baterai Panasonic Rechargeable berbentuk kotak dengan voltase 9V dan kapasitas 220 mAh. Baterai ini dipilih karena bisa di isi ulang sehingga dapat digunakan berulang dan mampu mentenagai Arduino beserta modul nRF2401. 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 member* dalam penggunaan daya baterai.

Gambar 5.4: Baterai Panasonic 9V

Untuk alat pengukur voltase dan arus baterai penulis menggunakan multimeter digital. Multimeter digital dipilih karena mudah dalam penggunaan dan pembacaan hasil.

Gambar 5.5: Multimeter digital

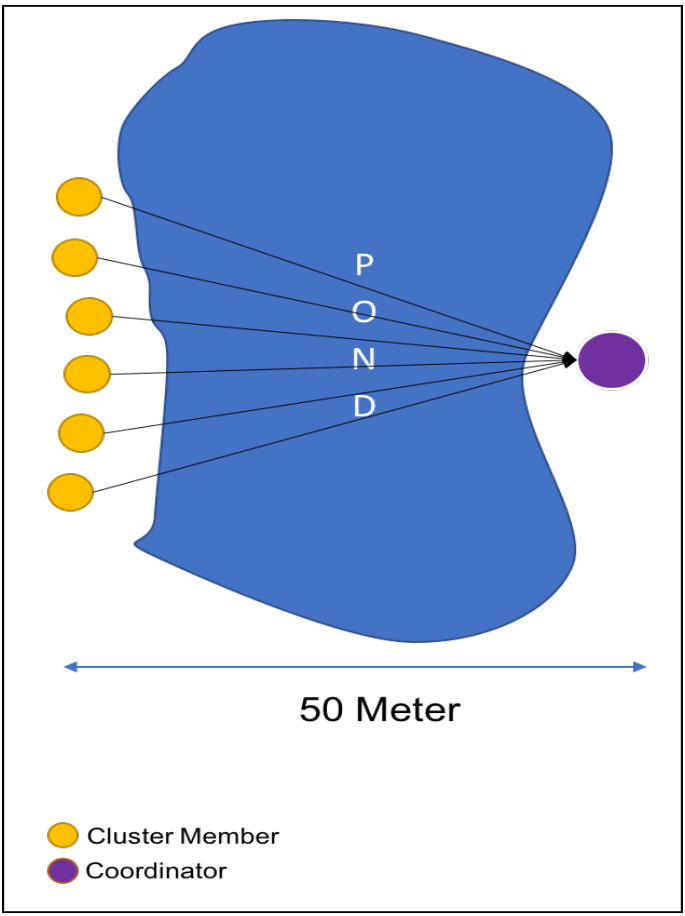
5.2 Skenario Uji Coba

Dalam uji coba akan dibuat 2 *cluster* yaitu *node cluster Direct transmission* dan cluster *Leach based cluster head selection* dengan pemilihan daya. *Node cluster* yang menggunakan *direct transmissien* semua node akan mengirim transmisi langsung ke *coordinator* tanpa melalui sebuah *cluster head*, sehingga tidak terdapat pemilihan *cluster head* dalam *cluster*. *Node cluster* yang menggunakan *Leach based cluster head selection* dengan pemilihan daya menggunakan metode seperti metode yang dibahas pada bab III dalam tugas akhir ini, pengiriman ke *coordinator* diwakilkan oleh sebuah *cluster head* dan terjadi proses pemilihan *cluster head*. Untuk *hardware* kedua *cluster* memakai *hardware* yang sama. Ilustrasi *cluster* dapat dilihat pada gambar 5.6 dan 5.7.

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 50 meter.

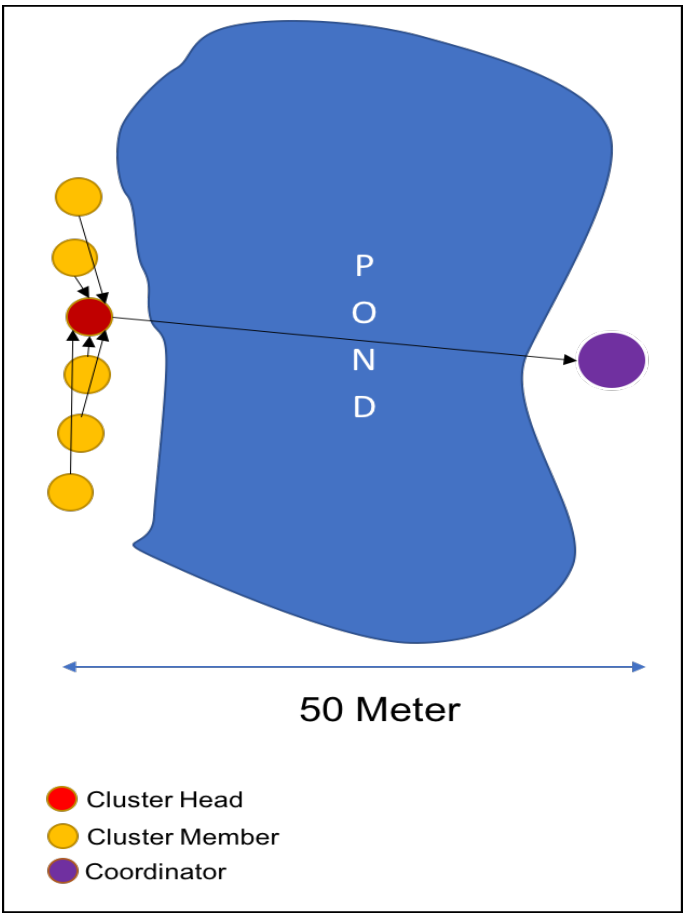
Modul nRF yang menggunakan daya transmisi tinggi akan menghabiskan baterai yang lebih banyak daripada modul nRF yang menggunakan daya transmisi rendah. Hal ini akan membuat perbedaan penggunaan energi antar *cluster* yang dapat kita ukur untuk menentukan selisih penurunan energinya.

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 5.6: *Cluster* Tanpa Pemilihan Cluster Head

Pada gambar 5.6 6 *cluster member* (lingkaran berwarna kuning) diletakan sejajar di pinggir kolam berseberangan dengan *coordinator* (lingkaran berwarna ungu) dengan jarak 50 meter. Garis panah merupakan arah transmisi, semua garis panah dari *cluster member* menunjuk ke *coordinator* yang menunjukan seluruh *cluster member* melakukan transmisi ke *coordinator*.



Gambar 5.7: *Cluster* dengan Pemilihan Cluster Head

Pada gambar 5.7 5 *cluster member* (lingkaran berwarna kuning) dan 1 *cluster head* (lingkaran berwarna merah) diletakan sejajar di pinggir kolam berseberangan dengan *coordinator* (lingkaran berwarna ungu) dengan jarak 50 meter. Garis panah menunjukan arah transmisi. Pada gambar *cluster member* mengirim transmisi ke *cluster head* lalu *cluster head* mengirim

transmisi ke *coordinator*.

Sebelum melakukan uji coba, perlu ditentukan skenario yang akan digunakan dalam uji coba. Melalui skenario ini, perangkat akan diuji apakah sudah berjalan dengan benar dan bagaimana performa pada masing-masing skenario. Dan membandingkan skenario manakah yang memiliki hasil lebih baik. Terdapat 3 macam skenario uji coba sebagai berikut:

1. Pengujian performa *cluster node* saat transmisi data ke koordinator tanpa menggunakan Pemilihan *Cluster Head* dan pemilihan daya dinamis. Dilihat dari *packet delivery ratio* dan penurunan daya baterai.
2. Pengujian performa *cluster node* saat transmisi data ke koordinator menggunakan Pemilihan *Cluster Head* dan pemilihan daya dinamis tanpa adanya *time slot*. Dilihat dari *packet delivery ratio* dan penurunan daya baterai.
3. Pengujian performa *cluster node* saat transmisi data ke koordinator menggunakan Pemilihan *Cluster Head* dan pemilihan daya dinamis dengan *time slot*. Dilihat dari *packet delivery ratio* dan penurunan daya baterai.

5.2.1 Skenario Uji Coba 1

Dalam skenario ini kita akan menghitung performa pada *cluster node* tanpa menggunakan Pemilihan *Cluster Head* dan pemilihan daya dinamis. Cara untuk mengetahui packet lost adalah memberikan nomor urut dan alamat node pengirim untuk setiap paket yang hendak dikirim dan *coordinator* akan menghitung berapa paket yang sampai kepadanya dan dari node mana *packet lost* tersebut berasal. Setiap *node* akan mengirimkan 20 paket ke *coordinator* sehingga ada total 120 paket yang di kirim ke *coordinator*. Jarak antara *node* dan *coordinator* dalam uji coba ini dalah 20 sampai 50 meter dengan

pernambahan jarak 10 meter 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 ujicoba terdapat pada tabel dibawah ini.

Tabel 5.1: Penghitungan PDR pada *cluster* node tanpa Pemilihan Cluster Head

Jarak (Meter)	**Paket dikirim**	**Paket terkirim**	**PDR**
20	120	120	100 %
30	120	120	100 %
40	120	116	96 %
50	120	113	94 %

Pada tabel 5.1 uji coba dilakukan dengan memberi jarak pada *cluster* dan *coordinator* sejauh 20, 30, 40, dan 50 meter. Tiap uji coba *cluster* akan mengirim paket sebanyak 120 paket ke *coordinator*, paket yang sampai akan dicatat dan dihitung PDR (*Packet Delivery Ratio*) dengan cara paket yang terkirim dibagi dengan paket yang terkirim dikali 100 persen. Pada tabel terlihat penurunan rasio PDR pada jarak 40 dan 50 meter. Dapat disimpulkan semakin jauh jarak antar *cluster* dan *coordinator* mempengaruhi performa pengiriman paket.

Tabel 5.2: Daya baterai sebelum uji coba pada *cluster* Direct Transmission

Node	**Voltase awal (V)**	**Arus awal (A)**	**V x I**
1	9.5 V	0.67 A	6.37 W
2	9.5 V	0.66 A	6.27 W
3	9.5 V	0.65 A	6.18 W
4	9.5 V	0.67 A	6.37 W
5	9.5 V	0.66 A	6.27 W

Tabel 5.2: Daya baterai sebelum uji coba pada *cluster* Direct Transmission

Node	**Voltase awal (V)**	**Arus awal (A)**	**V x I**
6	9.5 V	0.67 A	6.37 W
Rata-tara			6.30 W

Tabel 5.3: Daya baterai setelah uji coba pada *cluster* Direct Transmission

Node	**Voltase akhir**	**Arus akhir**	**V x I**
1	8.83 V	0.60 A	5.30 W
2	8.27 V	0.58 A	4.80 W
3	8.28 V	0.58 A	4.80 W
4	8.25 V	0.61 A	5.03 W
5	8.28 V	0.60 A	4.97 W
6	8.36 V	0.60 A	5.02 W
Rata-tara			4.99 W

Tabel 5.2 berisi data daya baterai tiap *node* sebelum dilakukan uji coba. Data yang dicatat adalah voltase (V) dan arus (A) tiap baterai tiap node dan daya baterai (W) yaitu perkalian dari V dan A, dilanjutkan dengan menghitung rata-rata daya baterai sistem. Arus dari baterai tiap *node* memang berbeda-beda walaupun menggunakan 1 model baterai yang sama.

Tabel 5.2 akan dibandingkan dengan tabel 5.3 untuk mendapatkan rata-rata penurunan daya baterai *cluster*. Tabel 5.3 sama seperti tabel 5.2 mencatat daya baterai tiap *node* namun data yang dicatat adalah daya baterai setelah uji coba skenario. Dari data hasil uji coba pada tabel 5.3 kita bisa lihat rata-rata daya baterai *cluster* adalah 4.99 W, lebih kecil dibandingkana rata-rata daya baterai sebelum uji coba yang dapat dilihat pada tabel 5.2. Maka dapat dihitung penurunan daya rata-rata *cluster* 6.30 W - 4.99 W yaitu 1.31 W.

5.2.2 Skenario Uji Coba 2

Dalam skenario ini kita akan menghitung performa pada *cluster node* menggunakan Pemilihan *Cluster Head* dan pemilihan daya dinamis tanpa *Time Slot*. Cara untuk mengetahui *packet lost* adalah memberikan nomor urut dan alamat *node* pengirim untuk setiap paket yang hendak dikirim dan *coordinator* akan menghitung berapa paket yang sampai kepadanya dan dari *node* mana *packet lost* tersebut berasal. Setiap *node* akan mengirimkan 20 paket ke *coordinator* sehingga ada total 120 paket yang di kirim ke *coordinator*. Jarak antara node dan *coordinator* dalam uji coba ini dalah 20 sampai 50 meter dengan pernambahan jarak 10 meter 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 ujicoba terdapat pada tabel di bawah ini.

Tabel 5.4: Penghitungan PDR pada *cluster* dengan pemilihan *cluster head* tanpa connection time slot

Jarak (Meter)	Paket dikirim	Paket Terkirim	PDR
20	120	119	99.2 %
30	120	119	99.2 %
40	120	117	96.7 %
50	120	116	95 %

Pada tabel 5.4 uji coba dilakukan dengan memberi jarak pada *cluster* dan *coordinator* sejauh 20, 30, 40, dan 50 meter. Tiap uji coba *cluster* akan mengirim paket sebanyak 120 paket ke *coordinator*, paket yang sampai akan dicatat dan dihitung PDR (*Packet Delivery Ratio*) dengan cara paket yang terkirim dibagi dengan paket yang terkirim dikali 100 persen. Pada tabel terlihat

dari jarak 20 meter PDR tidak mencapai 100 % seperti pada skenario 1. Penurunan PDR juga terjadi pada jarak 40 dan 50 meter walaupun lebih sedikit daripada skenario 1.

Tabel 5.5: Daya baterai sebelum uji coba pada *cluster* dengan pemilihan cluster head tanpa connection time slot

Node	**Voltase Awal (V)**	**Arus awal (A)**	**V x I**
1	9.5 V	0.67 A	6.37
2	9.5 V	0.66 A	6.27
3	9.5 V	0.66 A	6.27
4	9.5 V	0.65 A	6.18
5	9.5 V	0.66 A	6.27
6	9.5 V	0.67 A	6.37
Rata-tara			6.29

Tabel 5.6: Daya baterai setelah uji coba pada *cluster* dengan pemilihan cluster head tanpa connection time slot

Node	**Voltase akhir (V)**	**Arus akhir**	**V x I**
1	8.31 V	0.60 A	4.99
2	8.36 V	0.60 A	5.02
3	8.50 V	0.62 A	5.27
4	8.40 V	0.58 A	4.87
5	8.30 V	0.63 A	5.23
6	8.30 V	0.61 A	5.06
Rata-tara			5.07

Tabel 5.11 berisi data daya baterai tiap *node* sebelum dilakukan uji coba. Data yang dicatat adalah voltase (V) dan arus (A) tiap baterai tiap node dan daya baterai (W) yaitu perkalian

dari V dan A, dilanjutkan dengan menghitung rata-rata daya baterai sistem. Arus dari baterai tiap *node* memang berbeda-beda walaupun menggunakan 1 model baterai yang sama.

Tabel 5.11 akan dibandingkan dengan tabel 5.6 untuk mendapatkan rata-rata penurunan daya baterai *cluster*. Tabel 5.11 sama seperti tabel 5.6 mencatat daya baterai tiap *node* namun data yang dicatat adalah daya baterai setelah uji coba pada skenario ini. Dari data hasil uji coba pada tabel 5.6 kita bisa lihat rata-rata daya baterai *cluster* adalah 5.07 W, lebih kecil dibandingkana rata-rata daya baterai sebelum uji coba yang dapat dilihat pada tabel 5.2. Maka dapat dihitung penurunan daya rata-rata *cluster* 6.29 W - 5.07 W yaitu 1.22 W. Pada skenario 2 penurunan energi lebih sedikit dibanding dengan skenario 1.

5.2.3 Skenario Uji Coba 3

Dalam skenario ini kita akan menghitung performa pada *clustere* node menggunakan Pemilihan Cluster Head dan pemilihan daya dinamis dengan *Time Slot*. Cara untuk mengetahui *packet lost* adalah memberikan nomor urut dan alamat *node* pengirim untuk setiap paket yang hendak dikirim dan *coordinator* akan menghitung berapa paket yang sampai kepadanya dan dari *node* mana packet lost tersebut berasal. Setiap *node* akan mengirimkan 20 paket ke *coordinator* sehingga ada total 120 paket yang di kirim ke *coordinator*. Jarak antara node dan *coordinator* dalam uji coba ini dalah 20 sampai 50 meter dengan pernambahan jarak 10 meter 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 ujicoba terdapat pada tabel.

Tabel 5.7: Penghitungan PDR pada *cluster* dengan pemilihan cluster head dengan connection time slot

Jarak (Meter)	**Paket dikirim**	**Packet terkirim**	**PDR**
20	120	120	100 %
30	120	120	100 %
40	120	117	96.7 %
50	120	117	96.7 %

Pada tabel 5.7 uji coba dilakukan dengan memberi jarak pada *cluster* dan *coordinator* sejauh 20, 30, 40, dan 50 meter. Tiap uji coba *cluster* akan mengirim paket sebanyak 120 paket ke *coordinator*, paket yang sampai akan dicatat dan dihitung PDR (*Packet Delivery Ratio*) dengan cara paket yang terkirim dibagi dengan paket yang terkirim dikali 100 persen. Pada tabel terlihat dari jarak 20 dan 30 meter PDR mencapai 100 %, lebih baik dari skenario uji coba 2. Namun pada jarak 40 dan 50 meter PDR menurun.

Tabel 5.8: Daya baterai sebelum uji coba pada *cluster* dengan pemilihan cluster head dengan connection time slot

Node	**Voltase Awal (V)**	**Arus Awal (A)**	**V x I**
1	9.5 V	0.67 A	6.37
2	9.5 V	0.66 A	6.27
3	9.5 V	0.64 A	6.27
4	9.5 V	0.67 A	6.18
5	9.5 V	0.66 A	6.27
6	9.5 V	0.67 A	6.37
Rata-tara			6.29

Tabel 5.9: Daya baterai setelah uji coba pada *cluster* dengan pemilihan cluster head dengan connection time slot

Node	**Voltase Akhir (V)**	**Arus Akhir (A)**	**V x I**
1	8.31 V	0.62 A	5.15
2	8.37 V	0.60 A	5.02
3	8.40 V	0.60 A	5.04
4	8.40 V	0.59 A	4.96
5	8.40 V	0.63 A	5.29
6	8.50 V	0.61 A	5.19
Rata-tara			5.11

Tabel 5.8 berisi data daya baterai tiap *node* sebelum dilakukan uji coba. Data yang dicatat adalah voltase (V) dan arus (A) tiap baterai tiap node dan daya baterai (W) yaitu perkalian dari V dan A, dilanjutkan dengan menghitung rata-rata daya baterai sistem. Arus dari baterai tiap *node* memang berbeda-beda walaupun menggunakan 1 model baterai yang sama.

Tabel 5.8 akan dibandingkan dengan tabel 5.9 untuk mendapatkan rata-rata penurunan daya baterai *cluster*. Tabel 5.9 sama seperti tabel 5.8 mencatat daya baterai tiap *node* namun data yang dicatat adalah daya baterai setelah uji coba pada skenario ini. Dari data hasil uji coba pada tabel 5.9 kita bisa lihat rata-rata daya baterai *cluster* adalah 5.11 W, lebih kecil dibandingkana rata-rata daya baterai sebelum uji coba yang dapat dilihat pada tabel 5.2. Maka dapat dihitung penurunan daya rata-rata *cluster* 6.29 W - 5.11 W yaitu 1.18 W. Pada skenario 3 penurunan energi lebih sedikit dibanding dengan skenario 2 maupun okenario 1.

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* (PDR)

Berikut adalah tabel hasil uji coba keseluruhan untuk *Packet Delivery Ratio*:

Tabel 5.10: Hasil uji coba keseluruhan untuk *Packet Delivery Ratio*

Metode	**Rata - rata PDR**
Direct Transmission	97.50 %
Pemilihan *cluster head* tanpa Connection Time Slot	97.52 %
Pemilihan *cluster head* dengan Connection Time Slot	98.35 %

Dari tabel diatas dapat dilihat metode *pemilihan cluster head* dengan Connection Time Slot memiliki PDR paling tinggi, dikarenakan adanya penjadwalan transmisi antar node yang mengurangi tubrukan transmisi.

5.3.2 Penurunan Daya Baterai

Berikut adalah hasil uji coba secara keseluruhan untuk penurunan daya baterai:

Tabel 5.11: Hasil uji coba keseluruhan untuk *Penurunan baterai*

Metode	**Penurunan Daya Baterai**
Direct Transmission	20.88 %
Pemilihan cluster head tanpa Connection Time Slot	19.3 %
Pemilihan cluster head dengan Connection Time Slot	18.74 %

Dilihat dari tabel diatas metode pemilihan cluster head dengan Connection Time Slot menggunakan energi baterai paling sedikit yaitu menggunakan rata-rata 1 8.74 % dari keseluruhan daya baterainya. *Direct transmission* menggunakan daya baterai paling banyak, hal ini dikarenakan semua node menggunakan kekuatan transmisi tinggi untuk menjangkau *coordinator* sehingga *cluster* memakan banyak energi.

BAB VI

KESIMPULAN DAN SARAN

Bab ini berisikan kesimpulan yang dapat dapat diambil dari hasil uji coba yang telah dilakukan. Selain kesimpulan, terdapat juga saran yang ditujukan untuk pengembangan perangkat lunak selanjutnya.

6.1 Kesimpulan

Kesimpulan yang didapatkan berdasarkan hasil uji coba pemilihan *cluster head* dan pengaturan kekuatan transmisi secara dinamis adalah sebagai berikut:

1. Modul nRf24L01 bekerja menggunakan cara *slave to slave* tanpa ada menggunakan *master coordinator*. Hal ini membuat nRF24L01 dapat melakukan *broadcasting* dan transmisi secara bebas sehingga setiap *node* dapat berkoordinasi. Kekuatan transmisi modul nRF24L01 juga dapat diatur sehingga sistem dapat mengatur kekuatan transmisi saat dibutuhkan.
2. Setiap putaran sensing *cluster head* akan dipilih untuk menjadi wakil *cluster* dalam mengirimkan data ke *coordinator*. Cluster akan mengatur kekuatan transmisinya dimana cluster head menggunakan transmisi tinggi dan *cluster* member menggunakan transmisi rendah untuk menghemat baterai. Metode pemilihan *cluster head* dan pengaturan kekuatan transmisi secara dinamis mampu menghemat penggunaan baterai sebesar 1.58% dibanding menggunakan *Direct Transmision*.
3. *Time slot* diterapkan dengan mengirimkan masing-masing *cluster member* waktu jeda transmisi ke *cluster head* menghindari penuhnya modul nRF24L01 akibat datangnya transmisi bersamaan. Dari hasil uji coba skenario 3 time slot meningkatkan *Packet Delivery Ratio* sebesar 0.75 % dibanding uji coba skenario 2 yang tanpa menggunakan

time slot.

6.2 Saran

Saran yang diberikan terkait pengembangan pada Tugas Akhir ini adalah:

1. Menambah jumlah node yang dipakai untuk pengujian. Jumlah node yang dipakai manih terlalu sedikit sehingga membuat hasil uji coba belum akurat dan dapat terjadi tidak adanya *cluster head* terpilih.
2. Menambahkan modul pengukur daya baterai pada Arduino untuk pengukuran yang lebih akurat.
3. Menambahkan sebuah *Real Time Clock* agar Arduino dapat membuat *scheduling* yang akurat.

DAFTAR PUSTAKA

[1] "Wireless network," Jul. 2018, page Version ID: 849533435. [Daring]. Tersedia pada: https://en.wikipedia.org/w/index.php?title=Wireless_network&oldid=849533435. [Diakses: Page Version ID: 849533435].

[2] Nordic Semiconductor, "nRF2401 PRODUCT SPECIFICATION."].

[3] Wikipedia, "Wireless sensor network," Jul. 2018, page Version ID: 849272708. [Daring]. Tersedia pada: https://en.wikipedia.org/w/index.php?title=Wireless_sensor_network&oldid=849272708. [Diakses: Page Version ID: 849272708].

[4] Wendi Rabiner Heinzelman, Anantha Chandrakasan, dan Hari Balakrishnan, "Energy-Efficient Communication Protocol forWireless Microsensor Networks," *Massachusetts Institute of Technology*, hal. 10, 2000.

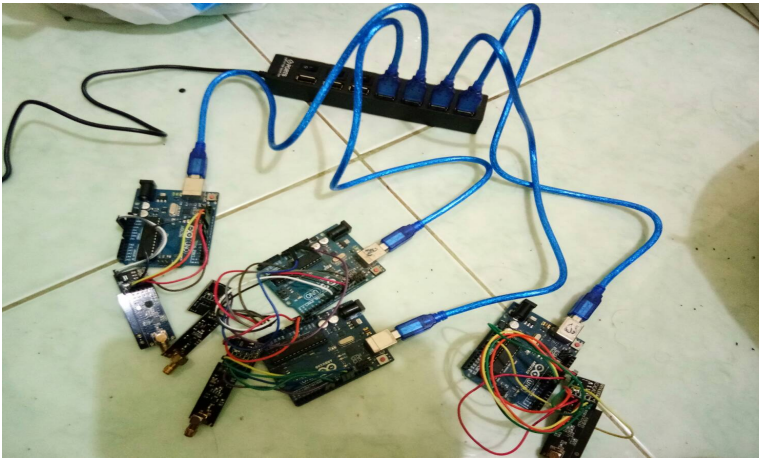
[5] Wikipedia, "Arduino," Jun. 2018, page Version ID: 847557047. [Daring]. Tersedia pada: https://en.wikipedia.org/w/index.php?title=Arduino&oldid=847557047. [Diakses: Page Version ID: 847557047].

[6] ——, "Atom (text editor)," Jun. 2018, page Version ID: 844627342. [Daring]. Tersedia pada: https://en.wikipedia.org/w/index.php?title=Atom_(text_editor)&oldid=844627342. [Diakses: Page Version ID: 844627342].

(Halaman ini sengaja dikosongkan)

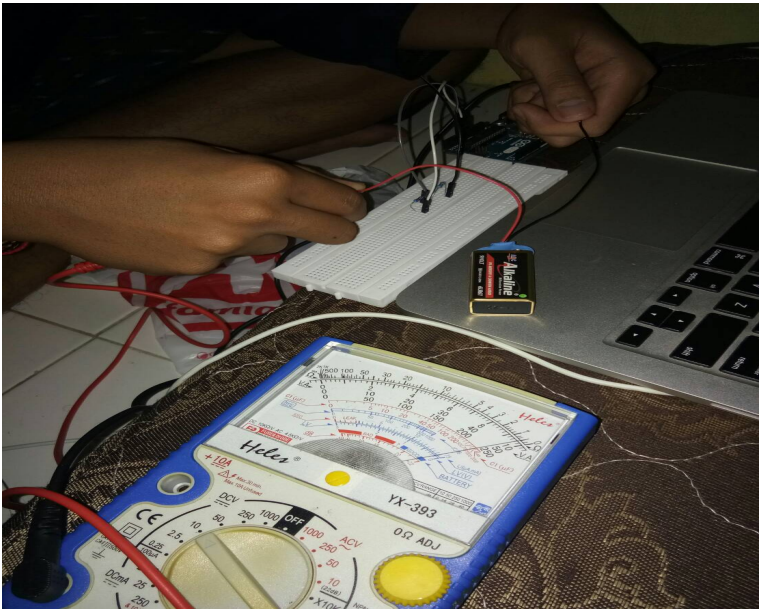
LAMPIRAN A

DOKUMENTASI PEMBUATAN *CLUSTER NODE*



Gambar 1.1: Pemrograman *cluster member*

Gambar 1.2: Modul Wireless nRF24L01



Gambar 1.3: Mengukur daya baterai dengan multimeter

LAMPIRAN B

KODE SUMBER

```
#include<Arduino.h>
#include <math.h>
#include <CountUpDownTimer.h>
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>


int n = 2;
int r = 1;
double p = 0.48;

RF24 radio(7, 8);

const long broadcast_addr = 1000;
const long self_addr = 2001;
const long sink_addr = 3003;
long CH_addr;

const char change_ch_message[] = "CNGCH";
//Flag saya CH atau bukan
bool flagCH = false;

//Flag CH sudah ada belum
bool flag_ch_found = false;

long packet_send = 0;
unsigned long packet_id_counter = 1;

int random_number;
```

```
struct data_CH{
    long CH_addr;
    long status;
};

struct dataSt{
    long addr;
    char msg[10];
    unsigned long id_packet;
};

data_CH ch_lain;
data_CH ch_now;

dataSt gmsg;
CountUpDownTimer T(DOWN);
CountUpDownTimer T1(UP, HIGH);

float findT()
{
    float phase1 = r * fmod(1 , p);
    float phase2 = 1 - p * phase1;
    float t = p / phase2;

    return t;
}

float getRandom()
{
    randomSeed(analogRead(A0));
    return random(1,100) / 100.0;
}
```

```
float getStatus()
{
  //TODO: masih dummy
    randomSeed(analogRead(A0));
    return random(1,100);

}

bool findCH()
{

    //Cek jumlah round
    if(r > n)
    {
        //Reset round
        r = 1;
    }

    float T = findT();
    float ran = getRandom();

    r++;

    if(ran < T)
    {
        return true;
    }
    else
    {
        return false;
    }

}
```

```
void setTimer(int h, int m, int s)
{
    T.SetTimer(h,m,s);
    T.StartTimer();
}

//
   ------------------------------------------------------------

//Blok fungsi untuk overloaded
   radio_listening
   ---------------------------------
bool radio_listening(long addr, data_CH *
   ret)
{
    radio.openReadingPipe(0, addr);
    radio.startListening();

    if(radio.available())
    {
        radio.read(ret, sizeof(data_CH));
        return true;
    }

    return false;
}

bool radio_listening(long addr, dataSt* msg
   , unsigned int packet_size)
{
    radio.openReadingPipe(0, addr);
    radio.startListening();
```

```
    if(radio.available())
    {
        radio.read(msg, sizeof(dataSt));
        return true;
    }

    return false;
}

bool radio_listening(long addr, char* msg,
   unsigned int packet_size)
{
    radio.openReadingPipe(0, addr);
    radio.startListening();

    if(radio.available())
    {
        radio.read(msg, packet_size);
        return true;
    }

    return false;
}

//End block fungsi radio_listening
```

```
//
```

```
//Block fungsi radio_send
-------------------------------------------------------

void radio_send(long addr, data_CH *msg)
{
    radio.stopListening();
    radio.openWritingPipe(addr);

    radio.write(msg, sizeof(data_CH));

    packet_send ++;
}

void radio_send(long addr, data_CH * msg,
   long delay_micro)
{
    radio.openWritingPipe(addr);
    radio.stopListening();
    delay(delay_micro);
    radio.write(msg, sizeof(data_CH));

    packet_send ++;
}

void radio_send(long addr, char *msg,
   unsigned int packet_size, long
   delay_micro)
{
    radio.openWritingPipe(addr);
    radio.stopListening();
    delay(delay_micro);
    radio.write(msg, sizeof(msg));
```

```
    packet_send ++;
}

void radio_send(long addr, char *msg,
   unsigned int packet_size)
{
    radio.openWritingPipe(addr);
    radio.stopListening();
    radio.write(msg, packet_size);

    packet_send ++;
}

void radio_send(long addr, dataSt* msg,
   long delay_micro)
{
    radio.openWritingPipe(addr);
    radio.stopListening();
    delay(delay_micro);
    radio.write(msg, sizeof(dataSt));

    packet_send ++;
}

//End block fungsi radio_send
   ------------------------------------------------------------



void reset_ch_flag()
{
    //Reset status CH sebelum melakukan
       findCH kembali
    flagCH = false;
```

```
    flag_ch_found = false;
}

void tell_i_am_ch()
{
    data_CH data;
    data.CH_addr = self_addr;
    data.status = -1;

    radio_send(broadcast_addr, &data);
}

void wait_for_ch()
{
    setTimer(0, 0, 30);
    ch_now.status = 0;
    while(!T.TimeCheck(0, 0, 0))
    {
        //reset ch_sekarang
        //TODO: Jadikan fungsi
        T.Timer();
        Serial.println("Mebunggu CH");
        if(radio_listening(broadcast_addr,
           &ch_lain))
        {
            // if(ch_lain.status > ch_now.
               status)
            // {
            //     ch_now.CH_addr = ch_lain
               .CH_addr;
            //     ch_now.status = ch_lain.
               status;
            //     flag_ch_found = true;
```

```
            // }
            if(ch_lain.status == -1)
            {
              ch_now.CH_addr = ch_lain.
                  CH_addr;
              ch_now.status = ch_lain.
                  status;
              flag_ch_found = true;
                break;
            }
        }
    }
}

void setup_phase()
{
    if(findCH())
    {
        //Jika terpilih jadi CH
        flagCH = true;
        Serial.println("jadiCH");
        data_CH data;
        data.CH_addr = self_addr;
        data.status = packet_send +
            random_number;



        Serial.print("Status: ");
        Serial.println(data.status);
        //set broadcast selama 6 detik
        setTimer(0,0,10);
```

```
//Broadcast CH dan listening
   kemungkinan ada CH lain
while(!T.TimeCheck(0,0,0))
{
    //Serial.println("sfsf");
    T.Timer();

    //Broadcast dan listening
       secara bergantian
    //Broadcast saya CH di detik
       genap
    if(T.ShowSeconds() % 2 ==0 && T
       .TimeHasChanged())
    {
      randomSeed(analogRead(A0));
      radio_send(broadcast_addr, &
         data,random(1,100));
    }

    //Listening kemungkinan ada CH
       lain di detik kelipatan 3
    if(T.ShowSeconds() % 3 == 0 &&
       T.TimeHasChanged())
    {
      if(radio_listening(
         broadcast_addr, &ch_lain))
      {
          Serial.println("dapat␣
             dari␣CH␣lain:␣");
          Serial.println(ch_lain.
             status);
```

```
                //cek jika dapat ada CH
                    lain dengan status
                    sendiri dan
                    membandingkan
                //TODO: status masih
                    random
                if(ch_lain.status > data.
                    status)
                {
                    //Set CH addr ke CH
                        lain
                    flagCH = false;
                    //flag_ch_found =
                        true;
                    Serial.print(ch_lain.
                        CH_addr);
                    Serial.println("␣jadi
                        ␣CH");
                    // ch_now.CH_addr =
                        ch_lain.CH_addr;
                    // ch_now.status =
                        ch_lain.status;
                    //CH_addr = ch_lain.
                        CH_addr;
                    break;
                }
                else
                {
                    Serial.println("Saya␣
                        jadi␣CH");
                    flagCH = true;
                }
            }
```

```
            }
        }
    }
    else{
        //Jika tidak jadi CH listening
            untuk broadcast CH

        //Setting Timer untuk lama
            listening menunggu broadcast
            dari CH
        wait_for_ch();
    }
}

void setup() {
  // put your setup code here, to run once:
  Serial.begin(57600);
  radio.begin();
  Serial.println("mulai");
  gmsg.addr = self_addr;

  Serial.println("setup phase");

  random_number = getStatus();

  setup_phase();

  while( (flag_ch_found == false) && (
     flagCH == false) )
  {
    wait_for_ch();
  }
```

```
  if(flagCH == true)
  {
      tell_i_am_ch();
      delay(100);
      tell_i_am_ch();
  }

}

void loop() {
    //T.Timer();
    Serial.println("Masuk loop");
    if(flagCH)
    {
        //Jadi CH
        radio.setPALevel(RF24_PA_HIGH);
        Serial.println("Menunggu data");
        setTimer(0, 1, 0);

         dataSt msg;

        while(!T.TimeCheck(0, 0, 0))
        {
            T.Timer();

            // if(T.ShowSeconds() % 5 == 0)
            // {
            //      tell_i_am_ch();
            // }

            if(radio_listening(self_addr, &
               msg, sizeof(msg)))
            {
```

```
                //Kirim me sink
                Serial.println(msg.
                    id_packet);
                radio_send(sink_addr, &msg,
                    0);
            }
        }

        //Broadcast ganti CH
        setTimer(0,0,3);

        while(!T.TimeCheck(0,0,0))
        {
            T.Timer();
            Serial.println("Ganti ch");
            char text[] = "CNGCH";
            radio_send(broadcast_addr, text
                , sizeof(text));
        }
    }
    else if(flag_ch_found)
    {
        //jadi node biasa
        radio.setPALevel(RF24_PA_MIN);
        //setTimer(0,15,0);
        T1.StopTimer();
        T1.StartTimer();

        char msg[25];
        while(true)
        {
            T1.Timer();
```

```
            if(radio_listening(
                broadcast_addr,msg,sizeof(
                msg)))
            {
                Serial.println("Mendengar")
                    ;
                Serial.println(msg);
                if(strcmp(msg,
                    change_ch_message) == 0)
                {
                    Serial.println("
                        Mendapatkan pesan 
                        cahnge CH");
                    break;
                }
            }

            if(T1.ShowSeconds() % 3 == 0 &&
                T1.TimeHasChanged())
            {
                Serial.println("mengirim 
                    data");
                gmsg.id_packet =
                    packet_id_counter;
                strcpy(gmsg.msg, "dummy");
                radio_send(ch_now.CH_addr,
                    &gmsg, getRandom());
                packet_id_counter++;
            }

        }
    }
```

```
//TODO: Ganti cara cari ch agar
    bersamaan

radio.setPALevel(RF24_PA_HIGH);
if(flagCH)
{
     reset_ch_flag();
     wait_for_ch();
}
else
{
  reset_ch_flag();

  setup_phase();

  if(flagCH == true)
  {
       tell_i_am_ch();
       delay(100);
       tell_i_am_ch();
  }
  else
  {
       wait_for_ch();
  }
}

while( (flag_ch_found == false) && (
    flagCH == false) )
{
  setup_phase();
  if(flagCH == true)
  {
```

```
            tell_i_am_ch();
            delay(100);
            tell_i_am_ch();
        }
        else
        {
            wait_for_ch();
        }

    }

}
```

Kode Sumber B.1: Kode Sumber Program

(Halaman ini sengaja dikosongkan)

BIODATA PENULIS

I Gede Putu Nobby Aswi Phala, akrab dipanggil Nobby lahir pada tanggal 16 April 1996 di Tabanan, Bali. Penulis merupakan seorang mahasiswa yang sedang menempuh studi di Jurusan Departemen Informatika Institut Teknologi Sepuluh Nopember. Memiliki hobi antara lain membaca novel, utak-atik elektonika, dan juga bermain game, terutama game Real Time Strategi. Selama menempuh pendidikan di kampus, penulis juga aktif dalam organisasi kemahasiswaan, antara lain Staff Departemen Riset dan Teknologi Himpunan Mahasiswa Teknik Computer-Informatika pada tahun ke-2. Pernah menjadi staff WebKestari Schematics tahun 2015 dan 2016. Selain itu penulis pernah menjadi asisten dosen di mata kuliah Pemrograman Jaringan, serta asisten praktikum pada mata kuliah Dasar Pemrograman.