IMPLEMENTASI ALOKASI PENGALAMATAN DINAMIS PADA WIRELESS SENSOR NETWORK DENGAN METODE TREECAST

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Fadhil Arief Fathoni
NIM: 115060800111109

PROGRAM STUDI INFORMATIKA/ILMU KOMPUTER
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2015

PENGESAHAN

IMPLEMENTASI ALOKASI PENGALAMATAN DINAMIS PADA WIRELESS SENSOR NETWORK DENGAN METODE TREECAST

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Fadhil Areif Fathoni
NIM. 115060800111109

Skripsi ini telah diuji dan dinyatakan lulus pada
30 Oktober 2015
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Sabriansyah Rizqika Akbar, S.T., M.Eng.	Eko Setiawan, S.T., M.T.
NIP. 19820809 201212 1 004	NIK. 201102 870610 1 001

Mengetahui,
Ketua Program Studi Informatika / Ilmu Komputer

Drs. Marji, M.T.
NIP. 19670801 199203 1 001

PERNYATAAN ORISINALITAS SKRIPSI

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

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

Malang, 15 Oktober 2015

Fadhil Arief Fathoni

NIM. 11506080011109

KATA PENGANTAR

Puji dan syukur Penulis panjatkan kehadirat Allah SWT, karena hanya dengan rahmat dan bimbingannya Penulis dapat menyelesaikan tugas akhir dengan judul "Implementasi Alokasi Pengalamatan Dinamis pada *Wireless Sensor Network* dengan Metode Treecast" dengan baik. Tanpa rahmat dan bimbingan dari Tuhan Yang Maha Esa, maka niscaya Penulis tidak akan dapat menyelesaikan tugas akhir ini dengan baik dan tepat waktu.

Penulisan dan penyusunan laporan skripsi ini dapat terlaksana dengan baik karena adanya bantuan secara langsung maupun tidak langsung dari pihak tertentu diantaranya:

1. Kedua orang tua penulis Lukman Hakim dan Titin Nurhanendah. yang telah memberi motivasi, kasih sayang serta dukungan moril dan materil.
2. Bapak Sabriansyah Rizqika Akbar, S.T., M.Eng., selaku dosen pembimbing I yang telah banyak memberikan ilmu dan saran untuk laporan skripsi ini.
3. Bapak Eko Setiawam, ST., M.T., selaku dosen pembimbing II yang juga banyak memberikan ilmu dan saran untuk laporan skripsi ini.
4. Andrini Alifia Imandari yang telah memberikan banyak dukungan, doa, motivasi, dan semangat selama ini.
5. Teman-teman TIF G 2011, terima kasih untuk segala pengalaman, pelajaran, dan dukungan selama bergabung bersama kalian.
6. Semua teman-teman PTIIK, khususnya Informatika/Ilmu Komputer 2011 terima kasih atas segala bantuan dan dukungannya selama ini.
7. Segenap dosen dan karyawan PTIIK Universitas Brawijaya yang telah membantu pelaksanaan skripsi ini.
8. Seluruh karyawan BPTIK dan Student Employee BPTIK, terima kasih untuk dukungan semangat, pengalaman, dan pengertiannya selama ini.
9. Semua pihak yang tidak dapat penulis sebutkan satu per satu yang terlibat baik secara langsung maupun tidak langsung demi terselesaikannya tugas akhir ini.

Semoga jasa dan amal baik mendapatkan balasan dari Allah SWT. Dengan segala kerendahan hati, penulis menyadari sepenuhnya bahwa skripsi ini masih jauh dari sempurna karena keterbatasan materi dan pengetahuan yang dimiliki penulis. Akhirnya semoga skripsi ini dapat bermanfaat dan berguna bagi pembaca terutama mahasiswa PTIIK Universitas Brawijaya.

Malang, Oktober 2015

Penulis
fadeal.gtu@gmail.com

ABSTRAK

Fadhil Arief Fathoni. 2015. Implementasi Alokasi Pengalamatan Dinamis pada Wireless Sensor Network dengan Metode Treecast. Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya, Malang. Dosen Pembimbing Sabriansyah R.A, ST., M.Eng. dan Eko Setiawan, S.T., M.T.

Wireless Sensor Network merupakan salah satu contoh dari pengembangan teknologi jaringan nirkabel dan teknologi *ad-hoc* yang dari sensor-sensor yang terdistribusi, otonom, murah, hemat energi, dan memiliki kemampuan komunikasi & komputasi yang terbatas. Terkadang dalam penerapanya sensor-sensor harus bergerak mengikuti objek yang diamatinya, sehingga menyebabkan sensor berada dalam luar jaringan maupun berpindah ke jaringan lainya. Hal ini menimbulkan masalah apabila setiap kali sensor berpindah jaringan harus dikonfigurasi ulang secara manual. Salah satu solusi untuk mengatasi masalah tersebut adalah dengan menerapkan paradigma *auto*-konfigurasi pada WSN untuk meningkatkan aspek skalabilitas pada WSN itu sendiri. Salah satu bentuk dari *auto*-konfigurasi adalah penerapan sistem alokasi pengalamatan secara dinamis. Pada penelitian ini diterapkan pengalamatan dinamis dengan metode Treecast. Skenario pengujian dibagi menjadi skenario pemberian alamat, pemberian alamat kepada node yang baru bergabung dengan WSN, dan pemberian alamat kepada node yang berpindah parent. Hasil dari penelitian ini menyatakan bahwa pengalokasian alamat secara dinamis dengan metode Treecast mampu memberikan alamat dengan rata-rata waktu yang dibutuhkan 251.3 ms, kemudian waktu yang dibutuhkan dalam pemberian alamat kepada node yang baru bergabung dengan WSN rata-rata 1209.5 ms, dan waktu yang dibutuhkan dalam pemberian alamat kepada node yang berpindah parent sekitar 1227.4 ms. Waktu yang dibutuhkan sebuah *node* dari *booting* sampai menjadi *parent* dan dapat memberikan alamat kepada *node* lainya adalah 1304 ms.

Kata Kunci: Wireless Sensor Network, Pengalamatan Dinamis, *Treecast*, *Auto*-konfigurasi.

ABSTRACT

Fadhil Arief Fathoni. 2015. Implementasi Alokasi Pengalamatan Dinamis pada Wireless Sensor Network dengan Metode Treecast. Information Technology and Computer Science Program, Brawijaya University, Malang. *Advisor:* Sabriansyah R.A, ST., M.Eng. dan Eko Setiawan, S.T., M.T.

Wireless Sensor Network (WSN) is one example of the development of wireless network technology and ad-hoc technology. WSN consists of distributed sensors which is autonomous, inexpensive, energy efficient, and have limited communication and computing capabilities. Sometimes in the application of WSN, sensors must move to follow the observed object, causing the sensors out from the network or switch to other networks. This poses a problem if every time sensor switch to other network the sensor need reconfigured manually. One solution to overcome this problem is to implement auto-configuration paradigm on WSN to improve the scalability aspects of the WSN itself. One form of auto-configuration is the application of dynamic addressing allocation. In this study applies the Treecast dynamic addressing method. Testing scenario is divided into several scenarios of addressing, giving the address of the new node joins the WSN, and giving the address of the node that switching parent. Results from this study stated that the dynamic addressing allocation method Treecast, is able to provide an address for node with average time 251.3 ms, and average time that needed for addressing to a new node that joins the WSN is 1209.5 ms, and average time that needed for addressing to node that switching parent is 1227.4 ms. The average time taken by the node from booting until become parents and are able to provide the address is 1304 ms.

Keywords: *Wireless Sensor Network, Treecast, Dynamic addressing, Auto-configuration.*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Wireless Sensor Network, selanjutnya disebut WSN, merupakan salah satu contoh dari pengembangan teknologi jaringan nirkabel dan teknologi *ad-hoc.* WSN adalah sebuah jaringan yang tidak bergantung pada adanya infrastruktur yang tetap, dan terdiri dari sensor-sensor yang terdistribusi dan bersifat otonom, murah, hemat energi, dan memiliki kemampuan komunikasi serta komputasi yang terbatas. WSN seringkali digunakan untuk memonitor suatu kondisi fisik maupun lingkungan, contohnya seperti temperature, suara, dan tekanan, kemudian mengirim data hasil monitoring tersebut melalui jaringan yang ada hingga sampai ke tempat tujuan yang biasanya berperan untuk mengolah data dari sensor-sensor tersebut.

Pada masa kini WSN diterapkan di banyak bidang, seperti militer, pengawasan dan kontrol di bidang industri, kesehatan, dan banyak lagi bidang lainya. Dalam perkembanganya, seringkali sensor-sensor yang ada pada WSN ini bersifat dinamis, dikarenakan objek yang dimonitornya terkadang tidak hanya sebuah objek statis melainkan juga objek bergerak. Sehingga seringkali sensor-sensor pada WSN harus berada dalam luar jaringan maupun berpindah dari satu jaringan WSN ke jaringan WSN lain.

Dengan perkembangan tersebut, tentunya akan menimbulkan permasalahan tersendiri apabila *node-node* sensor yang ada pada WSN harus dikonfigurasi secara manual. Hal ini akan mengurangi tingkat skalabilitas pada WSN dan tentunya akan menghabiskan sumberdaya untuk sekedar melakukan pengaturan dan konfigurasi ulang sebuah *node* sensor pada WSN ketika terjadi perpindahan dari satu jaringan WSN ke jaringan lainya. Selain itu dalam melakukan *maintenance* pada WSN, seringkali terjadi penggantian sensor-sensor yang ada pada WSN. Dikarenakan adanya kerusakan, habisnya sumber tenaga dari sebuah *node* sensor, maupun pembaharuan pada sensor-sensor tersebut. Apabila kita harus melakukan konfigurasi satu persatu dari *node-node* baru tersebut tentunya akan kurang efisien dan terkadang menimbulkan kesulitan tersendiri. Dikarenakan akan memakan waktu untuk melakukan konfigurasi, dan terkadang letak dari *node-node* berada di tempat yang sulit dijangkau [1].

Salah satu solusi untuk mengatasi masalah tersebut adalah dengan cara menerapkan paradigma *auto*-konfigurasi pada WSN. Salah satu bentuk dari *auto*-konfigurasi adalah penerapan sistem alokasi pengalamatan secara dinamis. Dengan sistem ini pemberian alamat kepada *node-node* dalam WSN akan ditangani oleh sistem pengalamatan dinamis ini. Sehingga nantinya tidak memerlukan campur tangan manusia dalam melakukan konfigurasi alamat jaringan terhadap *node* sensor WSN yang baru saja bergabung dengan WSN tersebut. Dengan demikian akan menghemat waktu untuk melakukan pengaturan alamat jaringan pada *node-node* sensor yang berjumlah banyak.

Penelitian yang membahas menganai penerapan alokasi pengalamatan dinamis pada WSN telah dilakukan oleh Santashil Pal Chaudhuri dkk. [2] yang menerapkan metode *Treecast* dalam pemberian alamat secara dinamis pada WSN. Sistem yang dibangun bertujuan untuk memberikan alamat kepada *node* sensor secara dinamis, dan menjaga agar komunikasi sensor tetap dapat dilakukan meski adanya kegagalan pada salah satu *node* di WSN.

1.2 Rumusan Masalah

Berdasarkan uraian latar belakang di atas, maka dapat dirumuskan permasalahan pada penelitian ini yaitu sebagai berikut:

1. Bagaimana membangun sistem pengalokasian alamat secara dinamis pada *Wireless Sensor Network* (WSN) yang dapat memberikan alamat jaringan kepada *node-node* sensor di WSN?
2. Bagaimana performa pemberian alamat kepada *node-node* sensor di WSN?

1.3 Batasan Masalah

Adapun batasan permasalahan pada penelitian ini adalah:

1. Sistem yang diusulkan diterapkan pada *node* yang berbasis Arduino uno dan *Radio Frequency* (RF).
2. Sistem yang diusulkan hanya dapat memberikan alamat kepada *node-node* sensor dan mencegah terjadinya duplikasi alamat.
3. Metode pemberian alamat kepada *node-node* sensor dari sistem yang diusulkan menggunakan metode *Treecast.*

1.4 Tujuan

Tujuan penelitian ini adalah membangun sebuah sistem pengalokasian alamat secara dinamis pada *Wireless Sensor Networks* (WSN). Dimana sistem tersebut akan memberikan alamat secara dinamis pada node-node sensor yang tergabung dalam network WSN tersebut. Sehingga node-node sensor mendapatkan alamat dan dapat berkomunikasi dengan node lainya.

1.5 Manfaat

Manfaat yang bisa didapatkan dari penelitian ini adalah dapat memudahkan dalam mengaplikasikan dan mengembangkan jaringan *Wireless Sensor Networks* (WSN), selain itu manfaat lain adalah menerapkan paradigma *auto configuration* pada WSN, sehingga meningkatkan efisiensi waktu dan *scalability* dalam melakukan konfigurasi pada *node-node* yang terdapat pada WSN

1.6 Sistematika Penulisan

Sistematika pembahasan dari penyusunan penelitian yang direncanakan adalah sebagai berikut:

BAB I PENDAHULUAN

Pendahuluan terdiri dari latar belakang, identifikasi dan pembatasan masalah, rumusan masalah, tujuan dan manfaat serta sistematika pembahasan dari proyek akhir ini. *Wireless Sensor Network* (WSN) adalah jaringan yang seringkali tidak bergantung pada adanya infrastruktur yang tetap dan terkadang bersifat dinamis. Dengan sifat-sifat tersebut maka WSN sering kali digunakan untuk memonitor objek-objek yang diam dan bergerak. Sehingga seringkali sensor-sensor pada WSN harus berada dalam luar jaringan maupun berpindah dari satu jaringan ke jaringan lainya. Selain itu dalam melakukan pengembangan jaringan WSN terkadang ada penambahan *node* sensor pada jaringan WSN. Apabila *node-node* ini dikonfigurasi secara manual tentunya akan kurang efisien, karena perlu untuk melakukan konfigurasi terhadap *node-node* baru maupun melakukan perubahan konfigurasi ketika *node-node* sensor berpindah dari satu jaringan WSN ke jaringan lainya. Untuk mengatasi permasalahan tersebut perlu dibuat sistem untuk menerapkan *auto*-konfigurasi pada WSN. Salah bentuk dari *auto*-konfigurasi adalah penerapan alokasi pengalamatan secara dinamis. Dengan Sistem ini pemberian alamat kepada *node-node* dalam WSN akan ditangani oleh sistem pengalamatan dinamis ini. Sehingga nantinya tidak memerlukan campur tangan manusia dalam melakukan konfigurasi alamat jaringan terhadap *node* sensor WSN.

BAB II KAJIAN PUSTAKA DAN DASAR TEORI

Pada Bab II dilakukan pengkajian pustaka terhadap teori-teori yang berkaitan dan menunjang dalam penyelesaian penelitian ini. Sistem pengalokasian alamat secara dinamis pada WSN menjadi fokus pada penelitian ini. *Arduino* diginakan sebagai platform dasar pengembangan sistem ini, *Radio Frequency* digunakan sebagai media komunikasi yang menghubungkan antar *node-node* sensor yang berbasis *Arduino*. Untuk membuat sistem ini diperlukan metode pemberian alamat kepada *node-node* sensor, metode *Treecast* digunakan sebagai metode pemberian alamat kepada *node-node* tersebut. Untuk mewujudkan sistem diperlukan metode yang dapat diimplementasikan pada penelitian ini, dan mendapatkan lingkungan jaringan dengan lingkup permasalahan. Teori-teori yang diambil berasal dari jurnal, buku, dan sumber referensi lainnya yang berhubungan dengan topik yang akan diteliti.

BAB III METODE PENELITIAN

Langkah-langkah dalam membangun arsitektur sistem pengalokasian alamat secara dinamis pada WSN dijelaskan pada bab III. Penjelasan langkah-langkah seperti perancangaan, implementasi, pengujian, serta analisis kebutuhan sistem dibahas secara umum.

BAB IV PERANCANGAN DAN IMPLEMENTASI

Pada bab IV dijelaskan mengenai perancangan sistem serta penerapan dari rancangan sistem secara terperinci dengan langkah-langkah pengerjaan serta menampilkan gambar-gambar dari implementasi yang dilakukan. Implementasi dilakukan berdasarkan sistematika yang dibuat pada bab sebelumnya.

BAB V PENGUJIAN DAN ANALISIS

Pada bab V memuat proses dan data hasil pengujian dan analisis terhadap implementasi arsitektur sistem pengalokasian alamat secara dinamis pada WSN yang telah dibangun. Hasil pengujian didapatkan dari pengujian yang telah dilakukan sesuai dengan skenario pengujian yang telah ditentukan. Dari hasil tersebut dapat dilakukan analisis terhadap kinerja sistem, serta mengetahui bagaimana sistem dapat mencapai tujuan.

BAB VI PENUTUP

Kesimpulan dari pelaksanaan penelitian ini dibuat berdasarkan hasil pengujian dan analisis terhadap perancangan dan implementasi arsitektur sistem pengalokasian alamat secara dinamis pada WSN yang dibangun dan dirangkum pada bab penutup. Untuk pengembangan dari hasil dalam penelitian ini, maka diberikan saran-saran untuk perbaikan dan penyempurnaan dari penelitian ini.

BAB II KAJIAN PUSTAKA DAN DASAR TEORI

2.1 Kajian Pustaka

Beberapa penelitian sebelumnya yang menjadikan landasan dari sistem yang dibuat yaitu, *A Survey of Addressing Algorithms for Wireless Sensor Networks*. Dalam jurnal ini telah dilakukan pembahasan terhadap beberapa algoritma pemberian alamat dalam *Wireless Sensor Network*. Berbagai algoritma yang dibahas dalam jurnal ini diantaranya adalah:

a. Energy-efficient Node Addres Naming (ENAN)
b. Low-Energy Address Allocation Scheme (LEADS)
c. Centralized Stateful Address Configuration (CSAC)
d. Hybrid Address Configuration (HAC)
e. Router-Based Address Configuration (RBAC)
f. Structured Addressing Scheme (SAC)
g. DADless Distributed base Station (DDBS)
h. Cell-based Distributed Addressig Technique (C-DAT)
i. TreeCast
j. Random, Ephemeral Transaction Identifiers (RETRI)
k. Preemptive Distributed Address Assignment Mechanism (PDAAM)

Pada jurnal tersebut [3] dilakukan pembahasan terhadap setiap algoritma tersebut tentang bagaimana proses pemberian alamat dari setiap 11 metode yang disebutkan diatas. Hasil dari pembahasan tersebut dapat dilihat dalam tabel 2.1 berikut:

Tabel 2.1 Perbandingan Metode Pemberian Alamat

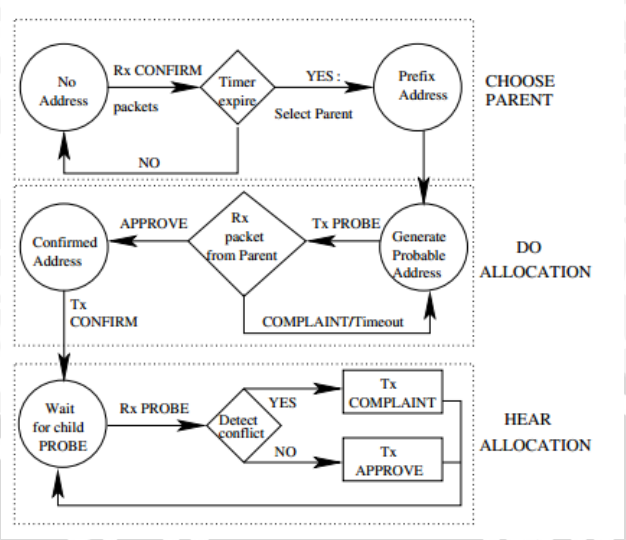
Addressing Protocol	Type	Allocation table	Address Space	Uniqueness	Address Reusing	Partition and merge
Muneeb et al. [15]	Stateful	Distributed cluster-based	Local 4-bit, Global 4n-bit	Local and Global	√	N/A
LEADS[38]	Stateful	Distributed disjoint	N/A	Global	N/A	√
CSAC[27]	Stateful	Centralized	16-bit	Global	N/A	N/A
HAC[27]	Stateful	Distributed disjoint	16-bit	Global	N/A	N/A
RBAC[27]	Stateful	Distributed disjoint	16-bit	Global	N/A	N/A
Ghulam et al. [23]	Stateful	Distributed disjoint	n-dimensional hyper-cube	Global	N/A	N/A
DDBS[42]	Stateful	Distributed cluster-based	Minimum number of bits	Local	√	N/A
C-DAT[40]	Stateful	Distributed cluster-based	16-bit	Local and Global	N/A	N/A
TreeCast[4]	Stateless	×	b^k	Global	√	√
RETRI[14]	Stateless	×	N/A	Local	√	N/A
PDAAM[22]	Hybrid	Distributed disjoint	16-bit	Global	N/A	√

Pada penelitian ini penulis mencoba untuk melakukan implementasi alokasi pengalamatan dinamis dengan metode *Treecast,* berdasarkan penelitian diatas metode *Treecast* dipilih untuk mengatasi permasalahan yang ada pada WSN yang telah dijelaskan pada bab sebelumnya karena keunggulanya dibanding metode-metode yang dijelaskan pada jurnal tersebut [3]. Metode *Treecast* memiliki keunggulan dimana algoritma yang diterapkan lebih sederhana daripada metode lainya. Metode *Treecast* menerapkan alamat yang bersifat global dan tidak memerlukan clustering dalam penerapanya, sehingga memungkinkan kode program yang ditulis lebih kecil daripada metode lainya. Selain itu, metode *Treecast* bersifat *stateless*, yang berarti tidak menyimpan informasi mengenai *node* sensor lainya. Sehingga apabila terjadi kesalahan ataupun *malfunction* pada salah satu *node* maka tidak akan mempengaruhi *node* lainya.

2.2 Treecast

Sistem ini menggunakan metode *Treecast* [2] dimana metode ini bertujuan memberikan alamat pada *node-node* sensor dengan bentuk topologi jaringan layaknya sebuah pohon atau *Tree*. Bentuk *Tree* ini dibentuk mulai dari *sink node* dari sebuah jaringan WSN. *Sink node* tersebut akan menjadi akar atau *root* dari struktur *tree* yang dibentuk. Yang kemudian *node-node* sensor yang terhubung secara langsung dengan *sink node* akan memilih sebuah *n*-bit alamat yang akan digunakan oleh dirinya sendiri, *node-node* inilah yang kemudian akan menjadi *node-node* level 1.

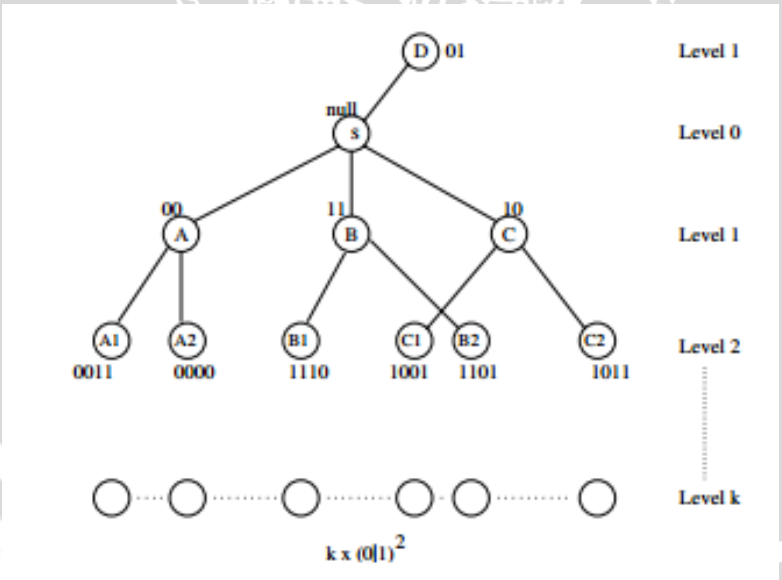
Setelah sensor yang berada pada level 1 ini memiliki alamat untuk digunakan pada dirinya sendiri, *node-node* sensor akan melakukan *broadcast* kepada *node-node* disekitarnya untuk memberitahukan alamat yang digunakanya. Kemudian *node-node* pada level 2 yang belum memiliki alamat akan mendapatkan pesan *broadcast* berisi alamat dari *node-node* level 1 tersebut. Selanjutnya *node-node* pada level 2 akan memilih salah satu pesan berisi alamat yang didapat dari *node-node* level 1 dan memilih *node* yang mengirim pesan tersebut sebagai orang tua atau *parent,* sehingga *node* level 2 tersebut akan menjadi anak atau *child* dari *node* level 1 yang dipilih.



Gambar 2.1 Gambaran Proses Algoritma *Treecast*

Sumber: [2, p. 3]

Kemudian *node-node* pada level 2 akan meng-*generate* sebuah *n*-bit alamat yang kemudian akan digabungkan dengan alamat yang digunakan oleh *parent*-nya dan kemudian digunakan untuk dirinya sendiri. Setelah *node-node* level 2 memiliki alamat, *node-node* tersebut akan mengirimkan pesan broadcast kepada node disekitarnya, begitu selanjutnya hingga *node* level selanjutnya. Gambaran mekanisme metode *Treecast* dapat dilihat sebagai berikut.



Gambar 2.2 Gambaran Topologi Algoritma *Treecast*

Sumber: [2, p. 4]

2.3 Wireless Sensor Network

Wireless sensor network atau disingkat dengan (WSN) adalah suatu peralatan *sistem embedded* yang didalamnya terdapat satu atau lebih sensor dan dilengkapi

dengan peralatan sistem komunikasi. Sensor disini digunakan untuk menangkap informasi sesuai dengan karakteristik informasi yang diinginkan.

Sebuah jaringan sensor merupakan sebuah insfratuktur yang terdiri dari *sensing* (*masuring*), komputasi dan elemen komunikasi yang memberikan administrator untuk melakukan instrumentsasi, mengamati, dan bereaksi tehadap fenomena dalam lingkungan tertentu. Administrator yang dimaksud adalah entitas sipil, komersial, ataupun industrial. Efisiensi daya di WSN umumnya dicapai dalam tiga cara yaitu :

Operasi *Low-duty-cycle*

1. Proses local atau di sebuah jaringan untuk operasi mengurangi volume data karenanya transmisi waktu)
2. Jaringan multihop untuk mengurangi kebutuhan transmisi jarak jauh.

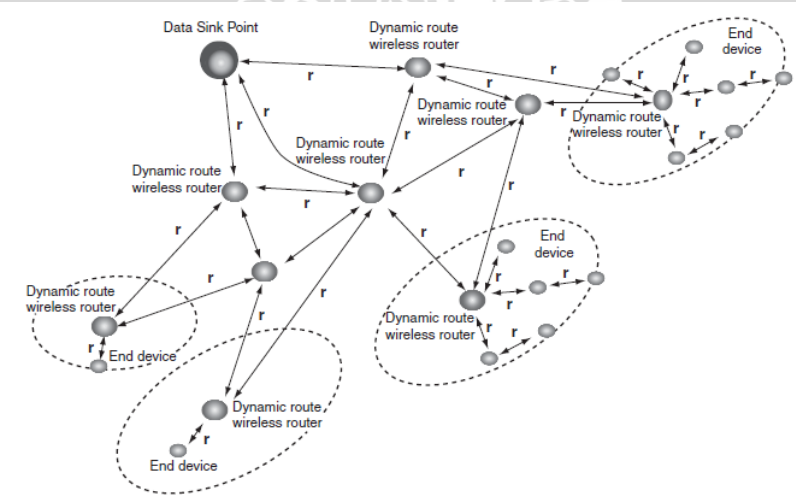
Pada saat ini ada berbagai jenis jaringan sensor nirkabel. *Wireless Sensor Network* saat ini dilengkapi dengan transceiver radio atau perangkat komunikasi nirkabel dan sumber energi yang biasanya berupa baterai. Ada empat komponen dasar dalam jaringan sensor:

1. Perakitan sensor secara distribusi atau local
2. Jaringan Interkoneksi
3. Titik pusat dari informasi clustering
4. Satu set sumber daya komputasi pada titik pusat (atau di luar) untuk menangani korelasi data, acara tren query, dan data mining.

Terdapat 2 kategori dalam WSN, yaitu:

1) WSN Kategori 1 (C1WSNs_

Pada kategori 1, jaringan menggunakan topologi mesh dengan jaringan radio multihop di kalangan atau diantara node pada WSN, dan memanfaatkan routing dinamis pada *wireless* dan kabel jaringan.



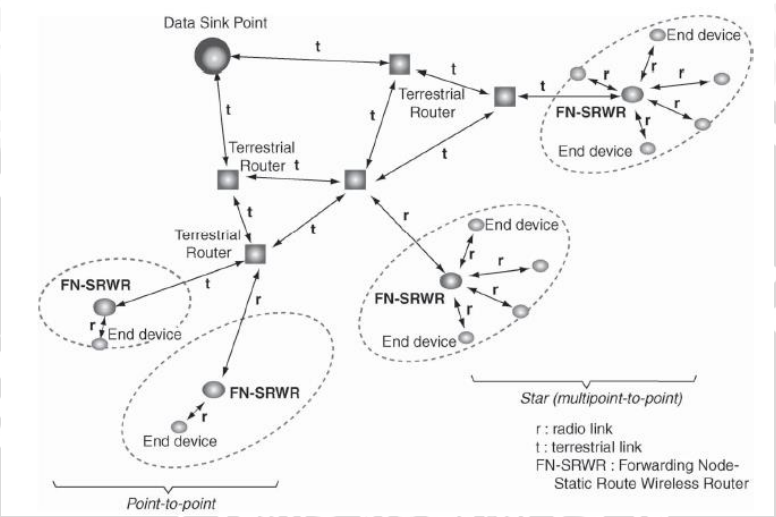
Gambar 2.3WSN Kategori 1

Sumber: [4]

Pada gambar 2.3 C1WSNs (WSN Kategori 1) merupakan jaringan dimana perangkat akhir (*end device*) yaitu sensor, diijinkan untuk lebih dari satu radio hop yang jauh dari *forwarding node*.

2) WSN Kategori 2 (C2WSNs)

Pada kategori 2, jaringan menggunakan topologi star (point-to-point) atau multipoint-to-point) dengan jaringan radio single-hop, memanfaatkan routing statis melalui jaringan nirkabel.



Gambar 2.4 WSN Kategori 2

Sumber: [4]

Pada gambar 2.4 C2WSNs (WSN Kategori 2) merupakan jaringan dimana perangkat akhir (*end device*) yaitu sensor, hanya terdapat satu radio hop yang jauh dari *forwarding node* [4].

2.4 Network Address

Network Address atau dalam bahasa Indonesia disebut dengan alamat jaringan, merupakan sebuah identitas bagi sebuah alat yang digunakan dalam sebuah jaringan. *Network Address* dalam jaringan pada umumnya bertujuan untuk meberikan informasi topologi dalam menentukan jalur. *Network Address* juga digunakan sebagai identitas layaknya sebuah nama, untuk menspesifikasikan sebuah ujung jalur komunikasi (*end point*), seperti sebuah jaringan telepon dimana setiap rumah memiliki sebuah nomor telepon yang berbeda [3].

2.5 Mikrokontroller

Mikrokontroller adalah sebuah *processor* dengan *memory* dan banyak komponen lainya yang terintegrasi menjadi satu dalam sebuah *chip*. Mikrokontroller saat ini digunakan dalam banyak aspek kehidupan, seperti peralatan rumah tangga (*microwave*, mesin cuci, mesin pembuat kopi), telekomunikasi (telepon gengam), industri otomotif (*fuel injection*), dan banyak bidang lainya. Mikrokontroller sudah mempunyai seluruh komponen yang

dibutuhkan untuk berjalan secara mandiri, dan telah di desain sedemikian rupa untuk digunakan sebagai alat *monitoring* dan atau mengontrol sebuah tugas. Salah satu jenis mikrokontroller yang beredar sekarang adalah Arduino yang menggunakan mikrokontroller Atmega328P [5].

2.5.1 Arduino

Arduino merupakan sebuah *platform* dari *physical computing* yang bersifat *open source*. *Physical Computing* adalah sebuah sistem atau perangkat fisik yang dibuat menggunakan *software* dan *hardware* yang bersifat interaktif, yaitu dapat menerima rangsangan dari lingkungan dan merespon balik. Physical computing adalah sebuah konsep untuk memahami hubungan antara lingkungan yang sifat alaminya adalah analog dengan dunia digital. Pada prakteknya konsep ini diaplikasikan dalam desain-desain alat atau projek-projek yang menggunakan sensor dan *microcontroller* untuk menerjemahkan input analog ke dalam sistem *software* untuk mengontrol gerakan alat-alat elektro-mekanik seperti lampu, motor, dan sebagainya [6].

Arduino tidak hanya sebuah alat pengembangan, malainkan kombinasi dari *hardware,* Bahasa pemrograman dan *Integrated Development Environment* (IDE) yang canggih. IDE adalah sebuah *software* yang berperan untuk menulis program, meng-*compile* menjadi kode biner dan meng-*upload* ke dalam *memory microcontroller*.

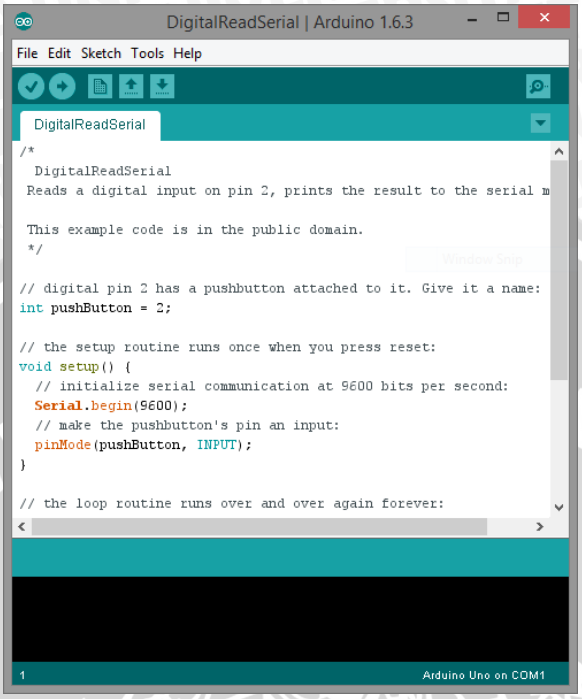
Kelebihan Arduino dari *platform hardware microcontroller* lain adalah sebagai berikut:

a. IDE Arduino merupakan *multiplatform* yang dapat dijalankan di berbagai sistem operasi, seperti Windows, Macintosh, dan Linux.
b. IDE Arduino dibuat berdasarkan pada IDE *processing* yang sederhana sehingga mudah digunakan
c. Pemrograman Arduino menggunakan kabel yang terhubung dengan *port* USB, bukan *port* serial. Fitur ini berguna karena banyak komputer yang sekarang ini tidak memiliki *port* serial.
d. Arduino adalah *hardware* dan *software open source* artinya dapat digunakan tanpa harus membayar.

2.5.2 Lingkungan Pemrograman Arduino

Lingkungan pemrograman disebut juga dengan *Intergrated Development Environment* (IDE), mempunyai kelebihan diantaranya mudah dalam penggunaan IDE-nya karena kesederhanaanya, selain itu pemrograman Arduino menggunakan kabel yang terhubung dengan *port* USB, karena banyak komputer yang sekarang ini tidak memiliki *port* serial, berbeda halnya dengan sistem minimum lainya yang

membutuhkan *downloader* tersendiri untuk memasukkan program ke dalam *microcontroller*.



Gambar 2.5 Tampilan IDE Arduino

Sumber: [7]

Gambar 2.5 merupakan tampilan dari *software* IDE Arduino yang digunakan dalam mengambangkan pemrograman berbasis Arduino.

2.5.3 Arduino Pro Mini

Arduino Pro Mini adalah board berbasis mikrokontroler ATmega328. Board ini memiliki 14 digital input/output pin (6 pin dapat digunakan sebagai output PWM), 8 pin analog, 16 MHz osilator Kristal serta tombol reset. Pin-pin ini berisi semua yang diperlukan untuk mendukung mikrokontroler. Untuk menuliskan program ke dalam Arduino Promini digunakan FTDI. Bentuk dari Arduino Promini ditunjukkan dalam Gambar 2.6.

Gambar 2.6 Board Arduino Pro Mini

Sumber : [7]

Arduino pro mini memiliki fitur- fitur sebagai berikut:

1. Serial: Pin yang digunakan untuk komunikasi Serial adalah Pin 0 sebagai TX dan pin 1 sebagai RX
2. I^2C: pin yang digunakan untuk komunikasi I^2C adalah pin SDA(A4) dan SDL (A5).
3. Pin Analog: A0, A1, A2, A3, A4, dan A5.
4. Pin Digital: ArduinoPro mini memiliki 14 pin digital yaitu pin 0 sampai pin 13.
5. SPI: Pin yang digunakan adalah **10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).**
6. PWM: Arduino pro mini sudah memiliki fitur PWM yaitu menggunakan pin 3, 5, 6, 9, 10, dan 11 dengan format 8 bit.
7. Interrupts external: pin yang digunakan adalah pin 2 dan pin 3.

Spesifikasi Arduino Pro mini dapat dilihat ditabel sebagai berikut:

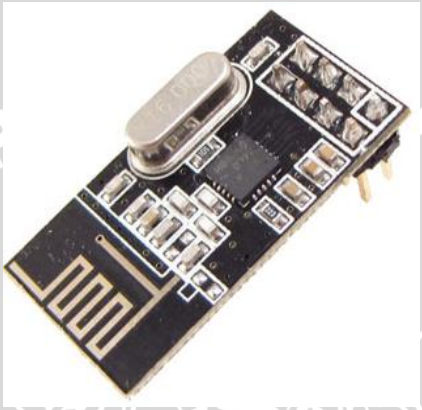
Tabel 2.2 Spesifikasi Arduino Pro Mini

Microcontroller	Atmega328
Operating Voltage	5V (*depending on model*)
Input Voltage	5 - 12 V (5V model)
Digital I/O Pins	14 (*of which 6 provide* PWM *output*)
Analog Input Pins	8
DC Current per I/O Pin	40 mA
Flash Memory	16 KB (*of which 2 KB used by bootloader*)
SRAM	1 KB
EEPROM	512 bytes
Clock Speed	16 MHz (5V model)

2.6 Modul Komunikasi Nirkabel pada Wireless Sensor Network

Dalam penerapan *Wireless Sensor Network* (WSN), selain adanya mikrokomputer sebagai otak dari *node-node* pada WSN dan sensor sebagai alat *sensing* terhadap objek yang diamati, terdapat juga modul komunikasi yang bertugas sebagai media mengiriman antar *node* yang terdapat dalam WSN. Media pengiriman yang digunakan dalam penerapan WSN ada berbagai macam salah satunya adalah *Radio Frequency* (RF).

nRF24L01 adalah sebuah modul komunikasi jarak jauh dengan single-chip RF-*transceiver* yang ditujukan untuk aplikasi pada gelombang 2.4 GHz ISM band.



Gambar 2.7 Modul nRF24L01

Sumber: [8]

Gambar 2.7 merupakan bentuk fisik dari modul nRF24L01, Modul nRF24L01 menggunakan antar muka SPI (*Serial Pheripheral Interface*) untuk komunikasi dengan tegangan kerja 5V DC.

a. Spesifikasi pada modul nRF24L01

1) *Worldwide* 2.4GHz ISM *band operation*
2) Jangkauan data di udara yaitu 250kbps, 1Mbps dan 2Mbps
3) *Ultra low power operation*
4) 11.3mA TX at 0dBm *output power*
5) 13.5mA RX at 2 Mbps *air data rate*
6) *Power* turun pada 900Na
7) 32 µA in standby-I
8) *On chip voltage regulator*
9) 1.9 to 3.6V *supply range*
10) *Enhanced ShockBurst™*
11) *Automatic packet handling*
12) *Auto packet transaction handling*

13) 6 *data pipe MultiCeiver™*

14) *Drop-in compatibility with* nRF24L01

15) *On-air compatible in* 250kbps dan 1Mbps dengan nRF2401A, nRF2402, nRF24E1 dan nRF24E2

16) *Low cost* BOM

17) ±60ppm 16MHz crystal

18) 5V *tolerant inputs*

19) *Compact* 20-pin 4x4mm QFN *package*

b. Imnplementasi Modul nRF24L01

1) *Wireless* PC *peripherals*
2) Mouse, keyboards dan remot
3) 3-in-1 desktop bundles
4) *Advanced Media center remote controls*
5) *VoIP headsets*
6) Pengontrol game
7) *Sports watches and sensors*
8) Remot kontrol RF untuk pengguna elektronika
9) Rumah dan *automasi* komersil
10) Jaringan sensor daya rendah
11) Active RFID
12) *Asset tracking sistems*

c. Fitur-fitur pada modul nRF24L01

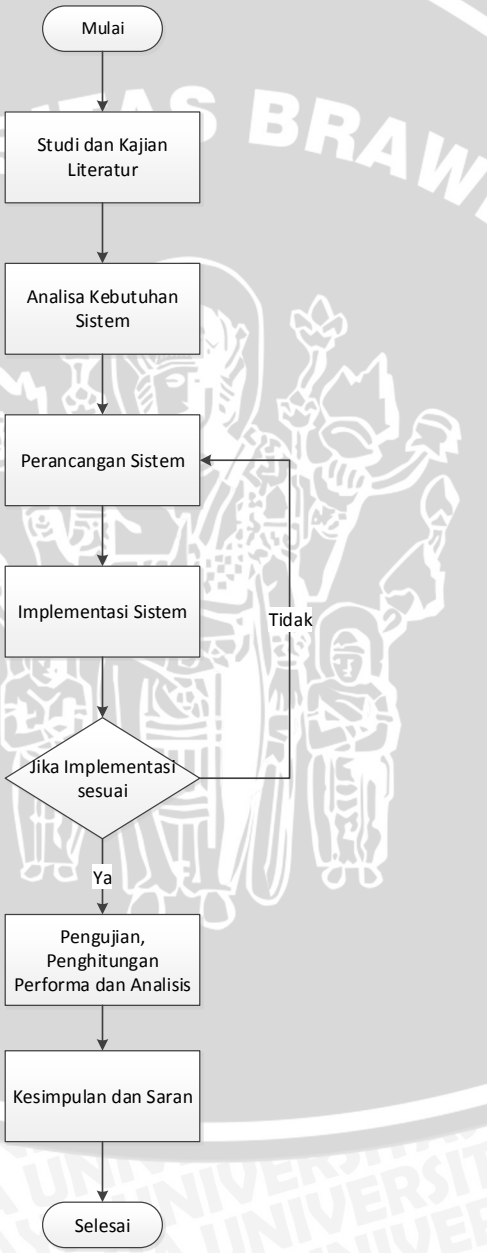
1) Radio:
   Worldwide 2.4GHz ISM *band operation*
   126 RF *channels*
   Common RX *and* TX *interface*
   GFSK *modulasi*
   250kbps, 1 and 2Mbps *air data rate*
   1MHz *non-overlapping channel spacing at* 1Mbps
   2MHz *non-overlapping channel spacing at* 2Mbps
2) Transmitter:
   Programmable output power: 0, -6, -12 or -18dBm
   11.3Ma at 0dBm *output power*
3) Receiver:
   Fast AGC for improved dynamic range
   Integrated channel filters
   13.5Ma at 2Mbps

-82dBm *sensitivity* at 2Mbps
-85dBm *sensitivity* at 1Mbps
-94dBm *sensitivity* at 250kbps

4) RF Synthesizer:
   Fully integrated synthesizer
   No external loop filer, VCO varactor diode or resonator
   Accepts low cost ±60ppm 16MHz crystal
5) *Enhanced ShockBurst™*:
   1 to 32 *bytes dynamic payload length*
   Automatic packet handling
   Auto packet transaction handling
   6 *data pipe MultiCeiver™* untuk 1:6 jaringan *star*
6) *Power Management*:
   Integrated voltage regulator
   1.9 to 3.6V *supply range*
   Idle modes with fast start-up times for advanced power management
   26Ma *Standby-I mode*, 900Na *power down mode*
   Max 1.5ms *start-up from power down mode*
   Max 130us *start-up from standby-I mode*
7) *Host Interface*:
   4-pin SPI
   Maksimal 10Mbps
   32 bytes TX and RX FIFOs
   5V *tolerant inputs*
8) *Compact* 20-pin 4x4mm QFN *packag*

BAB III METODE PENELITIAN

Bab ini berisi metode yang digunakan pada penelitian yang menjelaskan tentang tujuan dan cara dari tiap-tiap langkah yang dilakukan dalam penelitian. Metode penelitian menjelaskan tujuan dan cara dari tiap-tiap langkah yang dilakukan, meliputi studi literatur, analisis kebutuhan sistem, perancangan sistem, implementasi sistem, pengujian dan analisis, kesimpulan dan saran.



Gambar 3.1 Flowchart Metode Penelitian

3.1 Studi Literatur

Studi literatur menjelaskan seluruh dasar teori yang mendukung dalam penelitian perancangan modul slot parkir menggunakan *wireless sensor network* dengan perangkat arduino dan nRF24L01, adapun yang dijadikan bahan dalam studi literatur adalah dasar-dasar teori untuk dapat merancang sistem meliputi :

a. Arduino Promini
b. *Network Address*
c. nRF24L01
d. *Wireless Sensor Network*
e. Metode Pengalokasian Alamat *Treecast*

3.2 Analisis Kebutuhan Sistem

Analisis kebutuhan dalam merancang sistem tersebut terdiri dari kebutuhan fungsional dan non-fungsional. Kebutuhan fungsional yang terkait dengan sistem antara lain:

a. Sistem dapat memberikan informasi alamat *node* yang telah memiliki alamat kepada *node* yang belum memiliki alamat untuk digunakan sebagai acuan alamat yang digunakan.
b. Sistem dapat membuat alamat berdasarkan alamat *node* yang diberikan dari *node* sebelumnya.
c. Sistem dapat melakukan pengecekan terhadap alamat yang dibuat oleh sebuah *node* apakah alamat tersebut telah terpakai atau tidak (*duplicate address detection*).
d. Sistem dapat melakukan pengalokasian alamat baru apabila *node* yang menjadi acuan dalam membuat alamat terjadi masalah atau mati, dengan mencari *node* lainya untuk dijadikan sebagai acuan baru.

Analisis kebutuhan non-fungsional diperlukan untuk mengetahui kebutuhan sistem yang terkait dengan perangkat lunak maupun perangkat keras. Kebutuhan non-fungsional yang terkait dengan sistem antara lain:

a. Kebutuhan perangkat keras, antara lain :
    1. 2 perangkat Arduino Pro Mini
    2. 2 perangkat nRF24L01
    3. 1 unit computer
    4. 2 Perangkat FTDI

b. Kebutuhan perangkat lunak antara lain :

1. Arduino IDE

2.3 Perancangan Sistem

Berdasarkan proses analisis kebutuhan sistem yang telah dilakukan sebelumnya, dilakukanlah perancangan sistem. Perancangan sistem ini bertujuan untuk merancang sistem pengalokasian alamat secara dinamis pada *Wireless Sensor Network* (WSN) dengan kemampuan sistem sesuai dengan hasil dari analisis perancangan sistem sebelumnya. Perancangan sistem ini dilakukan dengan membuat diagram-diagram yang menunjukan alur kerja sistem yang bertujuan memenuhi kebutuhan-kebutuhan fungsional yang telah ditentukan sebelumnya.

3.4 Implementasi

Pada tahap ini menjelaskan tentang tahapan-tahapan implementasi berdasarkan hasil analisis kebutuhan sistem dan perancangan sistem. Pada tahap ini juga menjelaskan tentang lingkungan implementasi baik dari segi spesifikasi hardware maupun spesifikasi software.

Sistem ini nantinya akan dapat memberikan informasi konfigurasi alamat yang digunakan kepada *node-node* yang ada dalam W*ireless Sensor Network* (WSN), sistem juga dapat membuat alamat berdasarkan konfigurasi alamat yang telah diberikan oleh *node* sebelumnya, sistem juga dapat melakukan pengecekan terhadap alamat yang dibuat oleh *node-node* agar tidak terjadi duplikasi alamat yang digunakan oleh masing-masing *node*.

3.5 Pengujian dan Analisis

Pengujian terhadap sistem dilakukan dengan mengacu pada perancangan sistem yang meliputi:

- Pengujian pemberian alamat kepada *node* saat *booting* bersamaan dengan *node* lainya beserta waktu eksekusinya. Serta waktu yang dibutuhkan sebuah *node* dari mulai menyala sampai dapat menjadi *parent* bagi *node* lainya.
- Pengujian pemberian alamat kepada *node* yang baru ditambahkan dalam jaringan beserta waktu eksekusinya.
- Pengujian pemberian alamat kepada *node* yang kehilangan *node parent*-nya beserta waktu eksekusinya.

Dari data yang sudah didapat dari pengujian sistem, dilakukan analisa untuk mengetahui hasil yang akan digunakan untuk menarik kesimpulan dari penilitian yang dilakukan. Analisa hasil digunakan untuk mengetahui kelayakan dari sistem yang sudah dibuat.

3.6 Kesimpulan dan Saran

Pada tahap kesimpulan merupakan tahap terakhir dari penelitian yang telah memiliki hasil analisa data yang sudah dilakukan dan ditarik kesimpulan tentang sistem yang sudah dirancang.

BAB IV PERANCANGAN DAN IMPLEMENTASI

Bab ini berisi perancangan dan implementasi untuk penelitian. Perancangan membahas tentang rancangan yang ada pada penelitian. Dan implementasi membahas tentang pengimplementasian penelitian pada sistem yang ada di lingkungan.

4.1 Perancangan

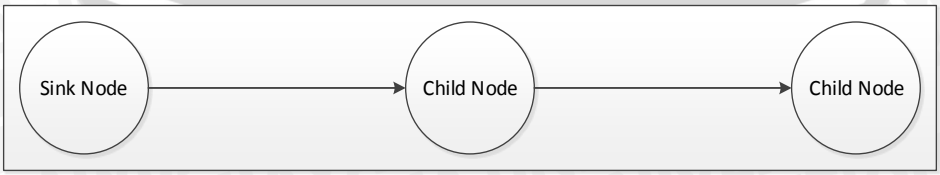
Tahap perancangan berisi tentang analisis perangkat keras dan perangkat lunak yang digunakan untuk membuat sistem tersebut. Ada beberapa tahap diantaranya sebagai berikut:

4.1.1 Lingkungan Sistem

Sistem alokasi pengalamatan dinamis pada *wireless sensor network* dengan perangkat Arduino Promini dan nRF24L01 membutuhkan perangkat lunak, perangkat keras dan analisis sistem.

1. Kebutuhan perangkat lunak yang digunakan pada sistem ini adalah :
   a) Compiler : Arduino IDE
   b) Simulator : ISIS Proteus 7
2. Kebutuhan perangkat keras sebagai berikut :
   a) Arduino Promini
   b) nRF24L01
   c) FTDI untuk konektivitas USB
3. Topologi Sistem

   Topologi sistem yang dibuat terdiri dari 3 *node*. Dimana terdapat *node* yang berperan sebagai *sink node* dan ada yang berperan sebagai *child node*. Topologi yang dibuat tampak pada gambar 4.1.



Gambar 4.1 Ilustrasi Topologi Sistem

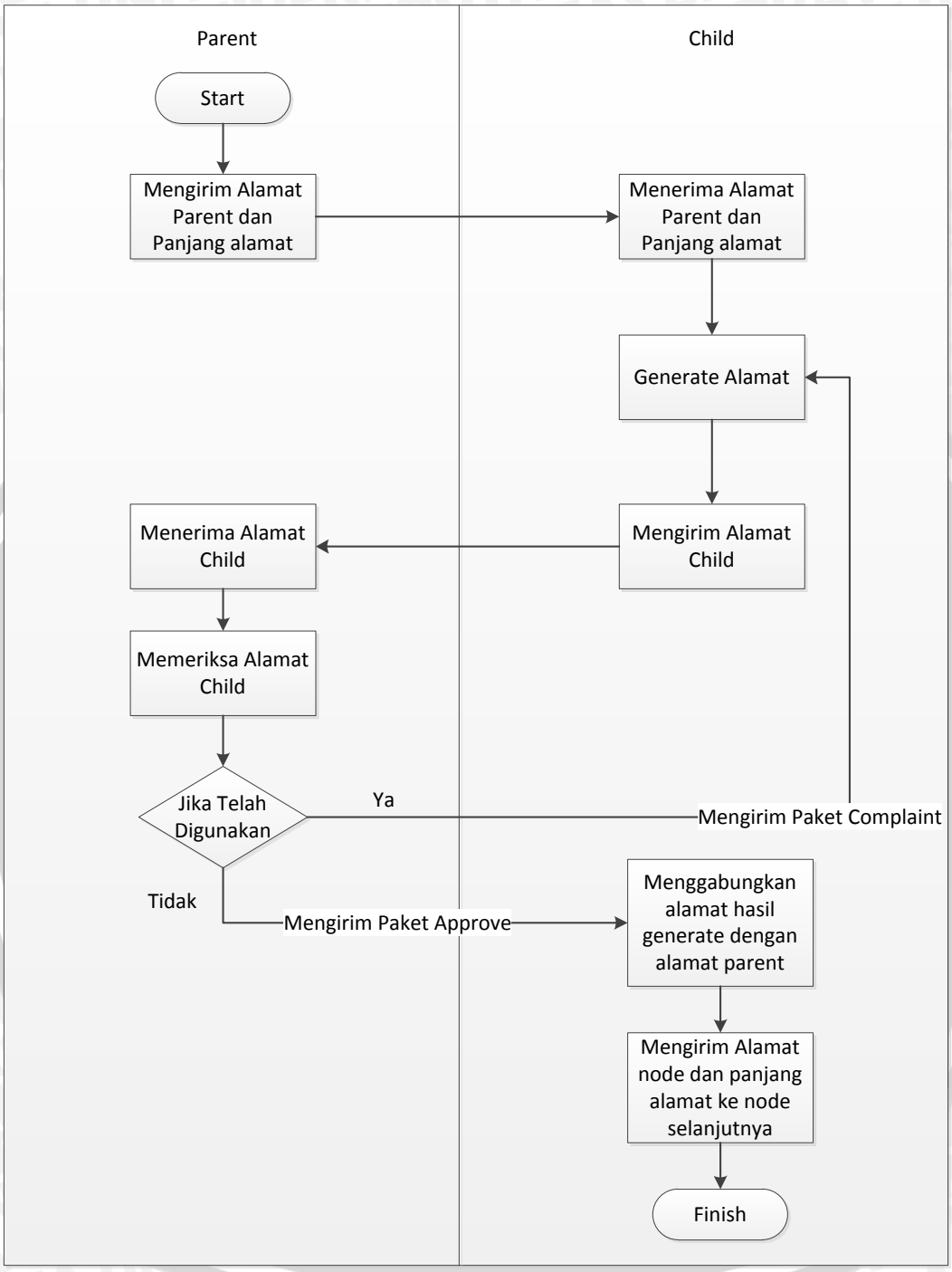
4. Analisis Kinerja Sistem

Secara keseluruhan sistem alokasi pengalamatan dinamis pada W*ireless Sensor Network* (WSN) dengan metode *Treecast* ini akan berjalan dengan *node-node* yang ada terbagi menjadi 2 peran, yaitu Orang Tua (*parent*) dan anak (*child*). Selain itu *node-node* yang ada juga ada yang berperan sebagai *sink node* yaitu sebuah node yang menjadi titik pusat (*root)* dalam sebuah *Wireless Sensor Network* (WSN). *Sink node* biasanya dalam WSN juga berfungsi sebagai tempat berkumpulnya data yang telah dikumpulkan oleh *node-node* sensor yang ada dalam WSN.

Proses jalanya sistem berawal dari *sink node* yang merupakan titik pusat (*root*) dalam sebuah jaringan WSN. Disini peran dari *sink node* adalah untuk menjadi orang tua (*parent*) bagi *node-node* yang ada pada jangkauan transmisi gelombang radionya. Sebuah node dikatakan *parent* apabila telah memiliki alamat jaringan yang telah dikonfigurasi. Selanjutnya *parent* akan mengirimkan pesan yang bersifat menyeluruh (*broadcast*) kepada *node-node* disekitarnya yang belum memiliki alamat. Pesan yang dikirim oleh *parent* ini berisi alamat dari *parent* itu sendiri (kita asumsikan dengan ***xx***) serta panjang alamat (*Length of Address*) yang digunakan. Dalam hal ini kita asumsikan panjang alamat yang digunakan *parent* adalah sepanjang 2 bit.

Setelah itu *node-node* yang menerima pesan dari *parent* tersebut akan menjadi *child*. Sebuah *child* yang telah menerima pesan dari *parent* akan membaca alamat yang digunakan oleh *parent* dan panjang alamat yang digunakan. Berbekal kedua informasi tersebut, *child* akan meng-*generate* secara acak sebuah alamat dengan panjang yang sama dengan alamat dari *parent*. Dengan demikian *child* telah memiliki alamat sepanjang 2 bit (kita asumsikan alamat hasil *generate* adalah **yy**).

Dari alamat hasi *generate* tersebut, akan dikirimkan kembali ke *parent*, yang kemudian oleh *parent* akan diperiksa apakah alamat tersebut sudah digunakan atau belum. Apabila telah digunakan maka parent akan mengirimkan pesan *COMPLAINT* kepada *child* yang meng-*generate* alamat tersebut. Kemudian *child* akan mengulangi proses tersebut dan meng-*generate* alamat baru dan dikirimkan lagi ke *parent.* Apabila alamat yang diperiksa oleh *parent* belum digunakan, maka *parent* akan mengirimkan pesan *APPROVE* kepada *child*, dan setelah itu *child* akan menggabungkan alamat *parent* dan alamat yang di setujui *parent* (dalam hal ini **xx** dan **yy** menjadi **xxyy**) kemudian akan digunakan sebagai alamat *child* tersebut. Setelah itu *child* akan bertindak sebagai *parent*, mengirimkan alamat dan panjang alamatnya secara *broadcast* kepada *node* lainya dan begitu seterusnya proses tersebut berjalan hingga node terakhir.

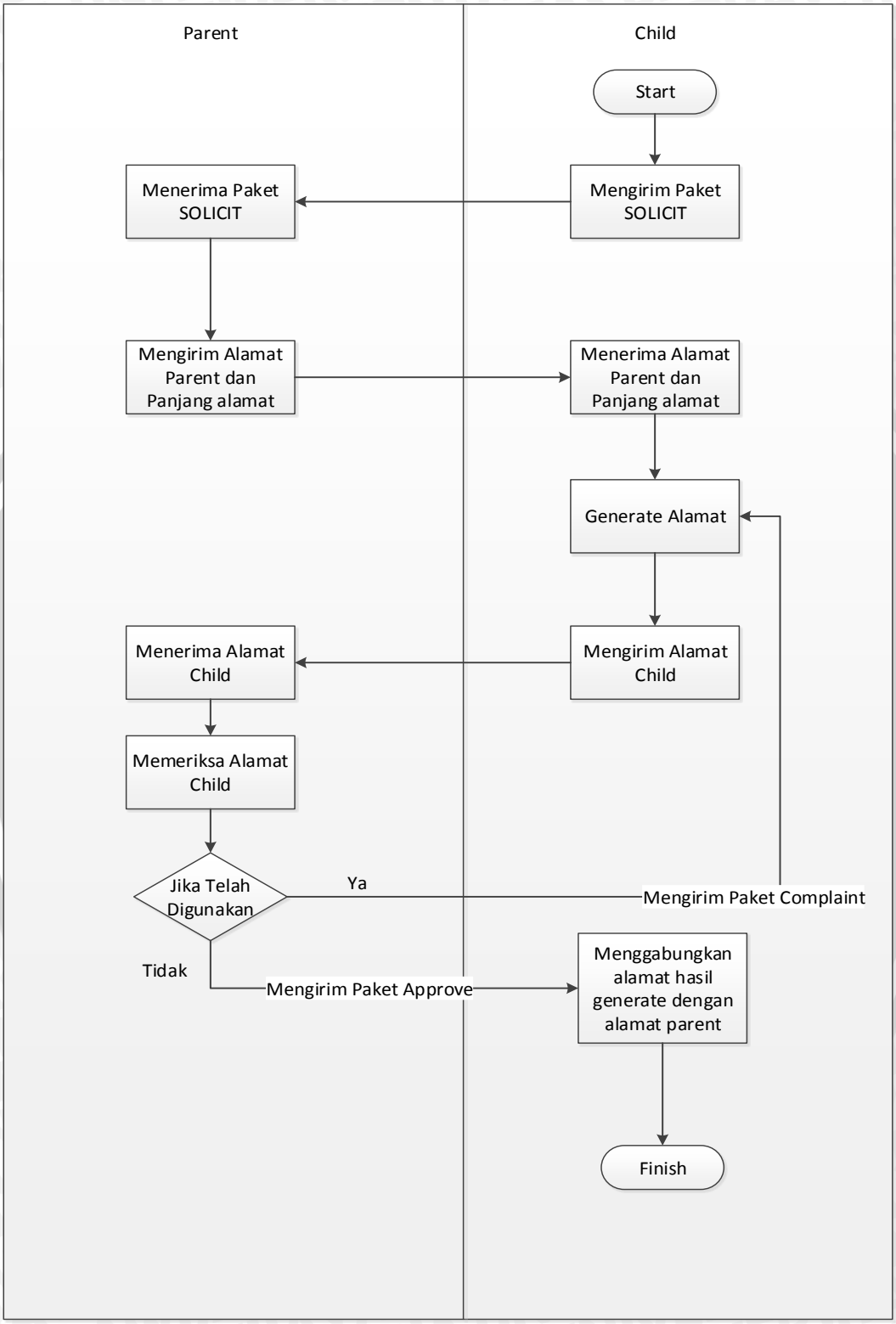


Gambar 4.2 Rancangan Alur Kerja Sistem dalam Pemberian Alamat

Gambar 4.2 merupakan rancangan alur kerja sistem dalam memberikan alamat dari satu node ke node lainya. Dimana proses akan dimulai dari *sink node* kemudian berlanjut ke *node* dibawahnya dan berlanjut ke *node* selanjutnya. Selain memberikan alamat, sistem juga harus dapat memberikan alamat kepada *node-node* yang baru bergabung dengan jaringan WSN tersebut. Hal ini diperlukan karena dalam pengaplikasian WSN sendiri

terkadang ada penambahan *node* baik karena kebutuhan maupun karena berkembangnya topologi pada WSN. Proses ini akan diawali oleh *node* baru yang bertindak sebagai *child*. Ketika *child* ini baru bergabung dengan jaringan WSN, *child* akan mengirimkan paket *SOLICIT* yang bersifat *broadcast* kepada *node-node* yang ada dalam jangkauanya.

Kemudian node-node disekitarnya akan mengirimkan alamtnya dan panjang alamat yang digunakan. Setelah itu *child* akan menerima alamat dan panjang alamat tersebut dan melakukan *generate* alamat berdasarkan panjang alamat yang diterima. Setelah itu alamat hasi *generate* akan dikirimkan ke *parent*, dimana *parent* akan memeriksa alamat tersebut apakah sudah digunakan atau belum. Apabila telah digunakan maka *parent* akan mengirim paket *COMPLAINT* dan *child* akan mengulangi proses *generate* alamat dan mengirimnya lagi ke *parent.* Namun apabila belum digunakan maka *parent* akan mengirim paket *APPROVE* dan *child* akan menggabungkan alamat hasil *generate* dengan alamat dari *parent*. Dengan proses inilah *node-node* yang baru bergabung dengan jaringan WSN akan mendapatkan alamatnya.



Gambar 4.3 Rancangan Alur Kerja Sistem dalam Pemberian Alamat pada Node Baru dalam jaringan

Tabel 4.1 Tabel Analisis Kebutuhan pada Sink Node

No	Analisis Kebutuhan
1	Mengirim alamat dan panjang alamat ke *node* selanjutnya
2	Menerima kiriman data alamat *node child*
3	Memeriksa alamat *child* apakah telah terpakai atau belum
4	Menerima paket data *SOLICIT* dari *node* yang baru bergabung dengan jaringan
5	Menerima paket *CHECK* dari *child*
6	Mengirim paket *PROBE* bahwa *parent alive* ke *child*
7	Mengirim paket *COMPLAINT* / *APPROVE* ke child

Tabel 4.2 Tabel Analisis Kebutuhan pada Node Sensor

No	Analisis Kebutuhan
1	Mengirim alamat dan panjang alamat ke *node* selanjutnya
2	Menerima kiriman data alamat *node child*
3	Memeriksa alamat *child* apakah telah terpakai atau belum
4	Menerima paket data *SOLICIT* dari *node* yang baru menyala
5	Menerima paket *CHECK* dari *child*
6	Mengirim paket *PROBE* bahwa *parent alive* ke *child*
7	Menerima kiriman data alamat dan panjang alamat dari *parent*
8	Menerima kiriman paket *COMPLAINT* / *APPROVE* dari parent
9	Mengirim paket *SOLICIT* ketika bergabung dengan jaringan
10	Mengirim paket *CHECK* ke *parent* untuk memeriksa kondisi *parent*
11	Mengirim alamat hasi *generate* ke *parent* untuk diperiksa oleh *parent*

4.1.2 Paket Data

Perancangan paket data merupakan bagian dari perancangan perangkat lunak. Pada bagian ini dilakukan perancangan terhadap paket data yang dipakai setiap kali pengiriman antar *node*, baik itu dari sensor *node* maupun *sink node*. Paket data yang digunakan dalam sistem alokasi pengalamatan dinamis ini merupakan sebuah *struct* yang terdiri dari beberapa tipe data yang berbeda.

1 Byte	1 Byte	1 Byte	1 Byte	1 Byte	2 Byte
Id	srcAddr	dstAddr	lengthAddr	tag	msg
Id	0-255	0-255	0-255	0-255	True / False

Gambar 4.4 Rancangan Format Paket Komunikasi Data

Pada gambar 4.4 adalah bagan frame data yang dipakai pada setiap pengiriman baik itu dari *sink node* maupun dari sensor *node*. Untuk fungsi dari masing-masing format data akan diterangkan pada tabel 4.3

Tabel 4.3 Keterangan Format Komunikasi Data

No	Jenis	Deskripsi
1	Id	Id pesan (000 = CHECK, 001 = FULL, 100 = SOLICIT, 101 = PROBE, 110 = NEW PARENT, 111 = CONFIRM)
2	srcAddr	Alamat asal paket
3	dstAddr	Alamat tujuan paket
4	lengthAddr	Panjang alamat yang digunakan
5	tag	Tag sebagai penanda paket
6	msg	Pesan Boolean APPROVE / COMPLAINT

Dari tabel diatas terlihat fungsi dari masing-masing format data yang dikirimkan dalam komunikasi antar *node* dalam sistem.

1. Id adalah *identifier* dari jenis paket yang dikirim, dimana jenis paket yang dikirim terbagi menjadi 6 macam yaitu :

   a) 000 (CHECK)

   Paket CHECK adalah paket yang dikirimkan ketika memeriksa apakah parent alive atau tidak

   b) 001 (FULL)

   Paket FULL adalah paket yang dikirimkan untuk memberitahu *child* bahwa kapasistas alamat *parent* telah penuh, ketika *child* mengirimkan alamat hasi *generate* untuk diperiksa oleh *parent*, sehingga *child* harus mencari *parent* baru.

   c) 100 (SOLICIT)

   Paket SOLICIT adalah paket yang dikirimkan untuk mencari parent baru, baik itu ketika *node* baru saja bergabung dengan jaringan, ketika *child* telah kehilangan parent, maupun ketika *child* gagal meminta alamat ke *parent* karena kuota alamat *parent* yang sudah penuh (FULL).

   d) 101 (PROBE)

   Paket PROBE adalah paket yang dikirimkan ketika *child* mengirimkan alamat hasil *generate* kepada *parent* untuk diperiksa apakah telah digunakan atau belum, selain itu digunakan juga untuk paket yang dikirimkan parent ke *child* untuk menyatakan alamat telah digunakan (COMPLAINT) atau alamat belum digunakan (APPROVE).

   e) 110 (NEW PARENT)

   Paket NEW PARENT adalah paket yang dikirimkan oleh *parent* kepada *child* untuk membalas paket SOLICIT yang telah dikirimkan sebelumnya oleh

child untuk mencari *parent* baru. Paket ini berisi informasi alamat *parent* dan panjang alamat yang digunakan.

f) 111 (CONFIRM)

Paket CONFIRM adalah paket yang dikirimkan oleh *node* apabila telah memiliki alamat yang disetujui oleh *parent*-nya. Tujuan dari paket ini adalah untuk memberitahu *node-node* selanjutnya bahwa *node* yang mengirim paket ini sudah siap menjadi *parent* dengan mengirimkan alamatnya dan panjang alamat yang digunakan.

2. Source Address (srcAddr) merupakan *field* yang berisi alamat asal paket tersebut dikirim. *Field* ini digunakan dalam beberapa kondisi, seperti ketika *child* mengecek kondisi *parent* apakah *alive* atau tidak, maka diperlukan informasi darimana paket ini berasal untuk mengarahkan *parent* menuju *node* mana nantinya balasan paket akan dikrim. Namun dalam beberapa kondisi lainya *field* dapat tidak digunakan karena *node* yang mengirim paket belum memiliki alamat.
3. Destination Address (dstAddr) merupakan *field* yang berisi alamat tujuan paket tersebut dikirim. Field ini digunakan dalam setiap pengiriman paket data, namun terkadang dalam paket yang bersifat *broadcast* seperti paket CONFIRM dan SOLICIT, *field* ini tidak digunakan.
4. Length Address (lengthAddr) merupakan *field* yang berisi panjang alamat dalam bit yang digunakan dalam sistem ini. Berdasarkan *field* ini lah *child* melakukan *generate* alamat sepanjang *n*-bit yang tertera dalam *field* ini. *Field* ini digunakan dalam paket CONFIRM dan NEW PARENT.
5. Tag (tag) merupakan field yang berisi *random* bilangan antara 1 sampai 255. Angka ini digunakan sebagai penanda paket ketika *child* mengirim paket PROBE kepada *parent* untuk mengirim alamat hasil *generate*
6. Message (msg) merupakan *field* yang dapat berisi pesan COMPLAINT / APPROVE. *Field* ini digunakan ketika mengirim paket PROBE untuk memberi konfirmasi apakah alamat yang diajukan oleh *child* sudah digunakan atau belum.

4.1.3 Media Radio Frequency dan Komunikasi Node

Komunikasi yang dilakukan antar node bersifat *wireless* dengan menggunakan *radio frequency*, jadi pada sub bab ini menjelaskan metode komunikasi yang dirancang. Komunikasi yang terjadi antara *parent* dan *child* bersifat *broadcast*. Hal ini dikarenakan setiap *node* yang bertindak sebagai *child* belum memiliki alamat. Kemudian proses berjalanya sistem alokasi pengalamatan dinamis akan berjalan bertahap dari *node* yang ada pada level teratas pada skema *Tree* menuju level terbawah. Hal ini dikarenan pada jaringan WSN jarak jangkauan

modul radio yang digunakan untuk komunikasi terbatas. Jadi digunakan metode ini untuk memberikan alamat kepada seluruh *node* yang ada dalam jaringan.

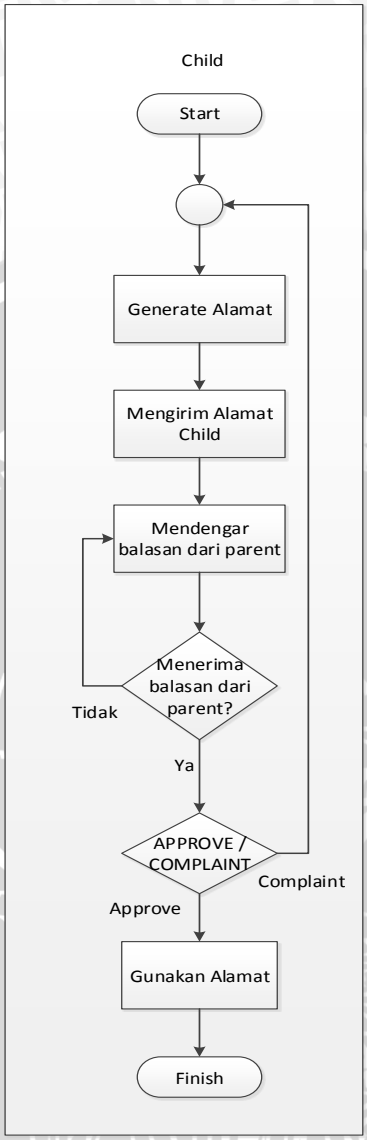
Modul radio yang digunakan adalah nRF24L01, modul ini memiliki 126 RF chanel dapat melakukan *auto packet handling*, modul ini memiliki 5 data pipeline, memiliki 32 *dynamic payload* jadi dapat mengirim 32 byte data dalam satu paket pengiriman. Dalam sistem ini *radio frequency* yang digunakan berada pada *frequency* 2.4GHz ISM Band dengan konfigurasi *channel* 20. Kemudian konfigurasi *payload* dalam setiap pengiriman dari radio frequency ini disesuaikan dengan besarnya paket data yang dikirim. Kemudian untuk membuat agar komunikasi antara *node* bersifat *broadcast* maka *transmiter address* dan *receiver address* harus disamakan antar satu node dengan node lainya.

Komunikasi yang ada pada sistem dilakukan secara *broadcast* pada *node-node* yang ada dalam jangkauan *sink node*. Pada lingkup inilah pertama kali proses pemberian alamat dijalankan. Ketika *node-node* yang berada dalam lingkup *sink node* telah memiliki alamat, barulah kemudian proses pengalokasian alamat berlanjut ke level berikutnya dengan menggunakan komunikasi *broadcast*. Dimana *node-node* yang berada pada level sebelumnya bertindak sebagai *parent*, sedangkan *node-node* pada level dibawahnya menjadi *child*.

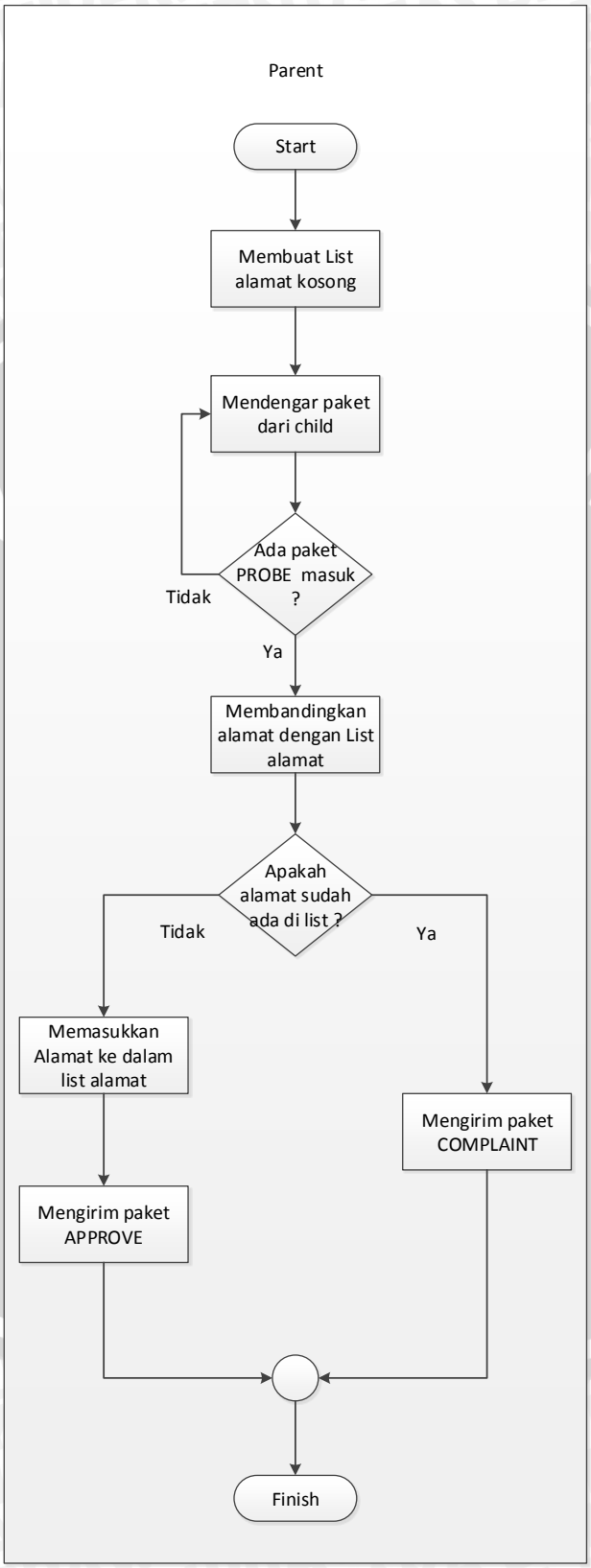
4.1.4 Algoritma Duplicate Address Detection

Pada sub bab ini akan dijelaskan tentang *duplicate address detection* yang berfungsi untuk mendeteksi adanya penggunaan alamat yang sama pada *node*. Hal ini diperlukan agar tidak ada alamat yang digunakan oleh lebih dari satu *node* sehingga menjamin komunikasi sampai pada node yang dituju. *Duplicate address detection* pada sistem ini berjalan pada tahap ketika sebuah *node child* telah melakukan *generate* alamat berdasarkan panjang alamat yang digunakan. Kemudian *child* akana mengirimkan alamat tersebut kepada *parent* untuk diperiksa apakah alamat tersebut telah digunakan atau tidak. *Parent* memiliki list alamat yang berisi status, apakah alamat tersebut telah digunakan sebuah *node* atau belum.

Sehingga ketika *parent* memeriksa sebuah alamat yang diajukan *child,* maka parent akan melihat alamat tersebut pada list yang dimilikinya. Apabila alamat tersebut telah digunakan, maka parent akan mengirimkan paket *COMPLAINT* sebagai pemberitahuan pada *child* bahwa alamat tersebut telah digunakan di *node lain*. Namun apabila alamat tersebut ternyata belum digunakan, maka *parent* akan mengirimkan paket *APPROVE* sebagai bentuk persetujuan kepada *child* untuk menggunakan alamat tersebut.



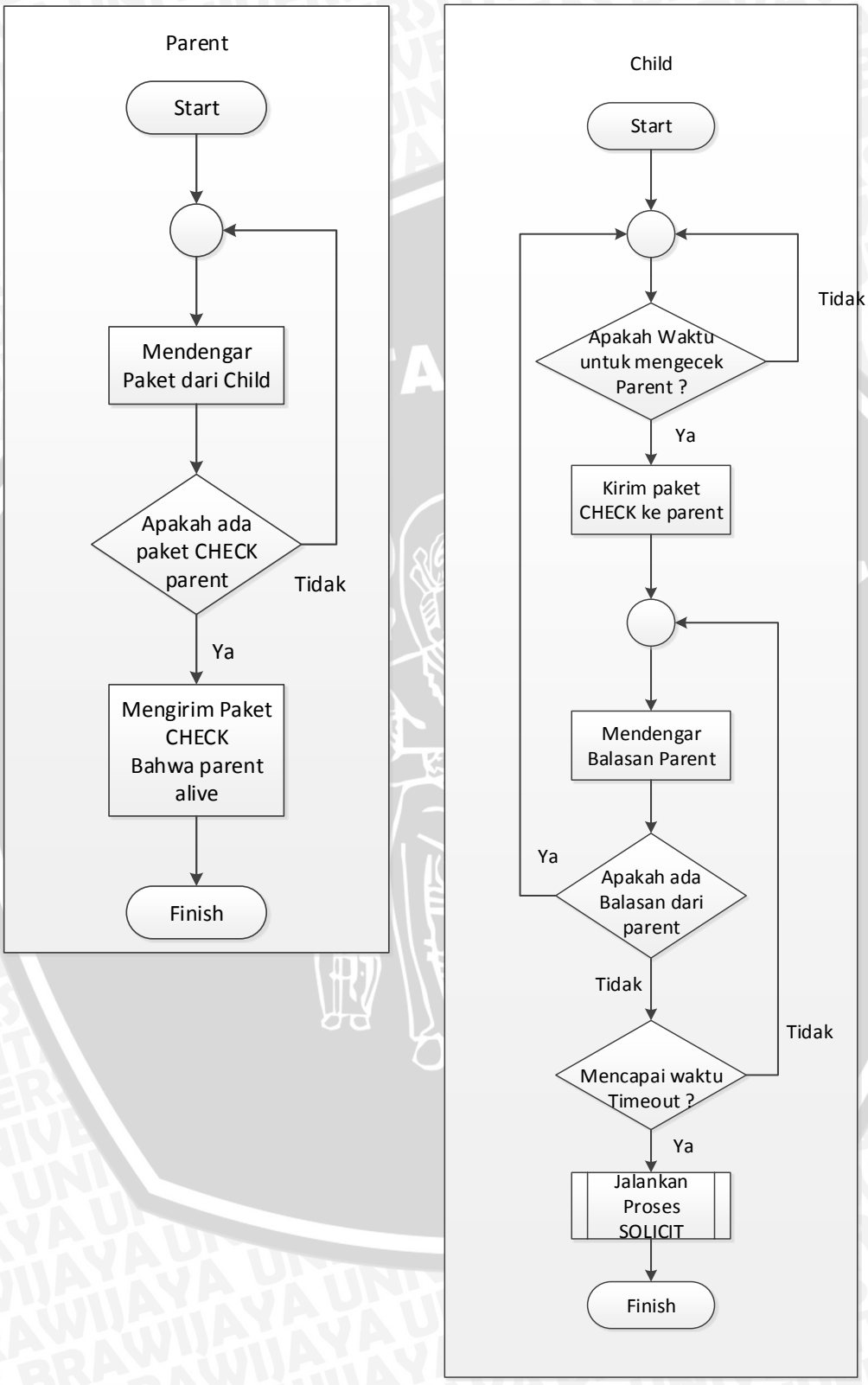
Gambar 4.5 Rancangan Proses Pengecekan Alamat pada Node Child



Gambar 4.6 Rancangan Proses Pengecekan Alamat pada Node Parent

4.1.5 Algortima Pengecekan Parent

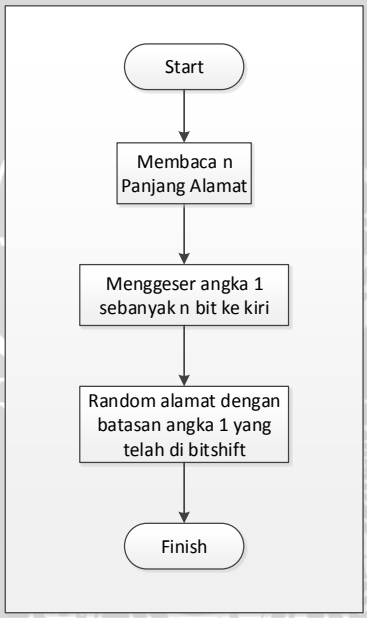
Pada sub bab ini akan dijelaskan tentang proses pengecekan *parent*, yang dimana akan dilakukan oleh *child*. Proses ini diperlukan untuk menjaga berlangsungnya komunikasi antar *node* dimana dalam metode *Treecast* ini komunikasi antar node bergantung pada *parent* dari node tersebut. Proses ini terjadi ketika sebuah *node child* telah mendapatkan alamat yang disetujui oleh *parent*-nya. Dalam kurun waktu tertentu secara periodik, *child* akan memeriksa komunikasinya dengan *parent.* Dengan demikian *node child* akan mengetahui apakah *parent* masih hidup atau tidak. *Child* akan mengirimkan paket *CHECK* kepada *parent*. Apabila parent masih hidup (*alive*), maka *parent* akan membalas paket tersebut kepada *child*. Namun apabila *parent* tidak membalas paket tersebut (*parent* mati atau mengalami kerusakan) maka *child* akan menganggapnya sebagai *timeout* dan akan mencari parent baru dengan proses yang sama layaknya *node* yang baru bergabung dengan jaringan WSN. Proses ini digambarkan pada gambar 4.7 dibawah ini.



Gambar 4.7 Rancangan Proses Pengecekan Parent pada Sistem

4.1.6 Algoritma Proses Generate Alamat

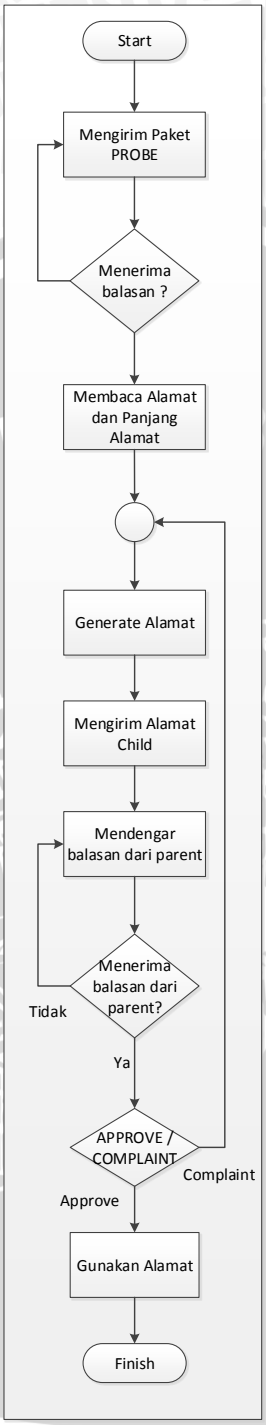
Pada sub bab ini akan dijelaskan tentang proses *generate* alamat yang dilakukan oleh *child* ketika melakukan proses alokasi alamat kepada *parent*. Ketika *child* telah mendapatkan alamat parent dan panjang alamat yang digunakan, maka *child* dapat melakukan *generate* alamat. Proses generate ini diawali dengan membaca panjang alamat dari *parent* kemudian melakukan *bit shift* atau disebut juga menggeser bit angka 1 sebanyak panjang alamat. Semisal panjang alamat yang digunakan adalah 2 bit maka angka 1 akan digeser ke kiri sebanyak 2 bit, menjadi 100 sehingga bernilai 4. Angka 4 ini lah yang akan dijadikan batasan untuk melakukan *generate* alamat secara *random*. Sehingga dengan fungsi random(4) maka hasilnya akan berada antara 0 – 3 begitu seterusnya.



Gambar 4.8 Rancangan Proses Generate Alamat pada Sistem

4.1.7 Algoritma Process Solicitation

Pada sub bab ini akan dijelaskan tentang proses *solicitation* pada *node child*. Proses ini akan dijalankan apabila dalam pengecekan *parent* ternyata *parent* tidak dapat menjawab dikarenakan *node parent* mati ataupun mengalami kerusakan. Proses ini akan diawali dengan *child* yang mengirimkan paket *SOLICIT* kepada *node-node* yang ada disekitarnya. Kemudian *node-node* disekitarnya membalas paket tersebut dengan mengirimkan alamatnya dan panjang alamat yang digunakan dalam sistem. Dari informasi yang diterima tersebut maka *child* akan melakukan proses *generate* alamat dan memeriksa alamat tersebut seperti halnya ketika melakukan proses pengalokasian alamat pertama kalinya.



Gambar 4.9 Rancangan Proses Solicitation pada Sistem

4.2 Implementasi

Sub bab ini membahas mengenai tahapan implementasi alokasi pengalamatan dinamis pada *Wireless Sensor Network* (WSN) berdasarkan hasil

yang telah didapat pada analisis kebutuhan dan proses perancangan perangkat lunak. Implementasi terdiri dari penjelasan lingkungan implementasi, batasan implementasi, dan implementasi sistem.

4.2.1 Lingkungan Implementasi

Lingkungan implementasi dari implementasi alokasi pengalamatan dinamis pada *Wireless Sensor Network* (WSN) ini menggunakan perangkat arduino dan nRF24L01, meliputi spesifikasi lingkungan perangkat keras (*hardware*) dan spesifikasi lingkungan perangkat lunak (*software*) dan bahasa pemrograman yang digunakan.

a. Spesifikasi lingkungan perangkat keras (*hardware*)

   Lingkungan perangkat keras yang digunakan dalam mengimpentasikan alokasi pengalamatan dinamis pada *Wireless Sensor Network* (WSN) dengan perangkat arduino dan nRF24L01 yaitu arduino pro mini dan nRF24L01.

b. Spesifikasi lingkungan perangkat lunak (*software*)

   Lingkungan perangkat lunak yang digunakan dalam mengimplementasikan alokasi pengalamatan dinamis pada *Wireless Sensor Network* (WSN) menggunakan arduino yaitu Arduino IDE sebagai *compiler* serta perancangan program yang akan dimasukkan ke dalam arduino, dan ISIS 7 Proteus sebagai aplikasi untuk menjalankan simulasi program sebelum dimasukkan ke dalam board arduino pro mini.

c. Bahasa pemrograman

   Bahasa pemrograman yang digunakan dalam mengimplementasikan alokasi pengalamatan dinamis pada *Wireless Sensor Network* (WSN) dengan perangkat arduino dan nRF24L01 yaitu bahasa pemrograman C pada Arduino. Dengan bantuan beberapa library pihak ketiga seperti mirf, sebagai libray untuk mengakses radio frekwensi nRF24L01

4.2.2 Batasan-batasan implementasi

Batasan implementasi ini dibuat dengan tujuan agar dalam pengujian sistem in dapat berjalan dengan baik. Beberapa batasan dalam implementasi alokasi pengalamatan dinamis pada Wireless Sensor Network (WSN) adalah sebagai berikut:

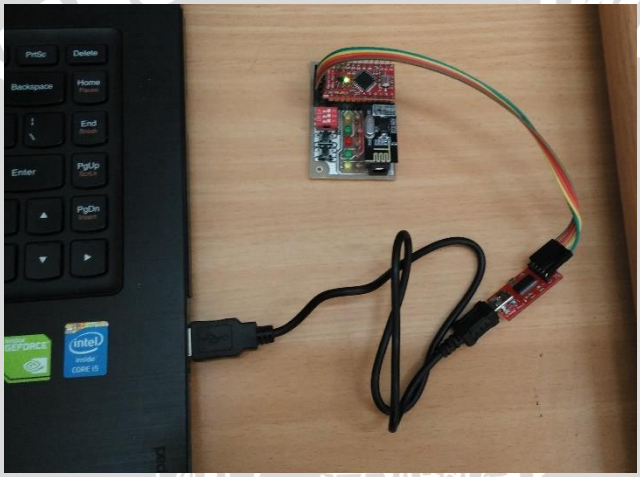
1. Panjang alamat yang digunakan dalam implementasi sepanjang 2 bit, yang ditampung dalam sebuah variabel byte. Sehingga panjang maksimal alamat yang digunakan sebesar 8 bit (8 bit = 1 Byte).
2. Jumlah *node* yang digunakan untuk implementasi sistem alokasi pengalamatan dinamis sejumlah 3 node, dimana dalam implementasi

terdapat node yang bertindak sebagai *sink node* dan ada yang bertindak sebagai sensor *node.*

3. Alamat yang digunakan oleh *sink node* ditentukan terlebih dahulu sebelumnya, dimana alamat yang bisa digunakan oleh *sink node* bit 01, 10, dan 11.

4.2.3 Implementasi Perangkat Lunak

Implementasi perangkat lunak merupakan tahap pemberian logika untuk perangkat keras agar dapat dioperasikan sesuai perintah yang diinginkan. Pada tahapan ini digunakan FTDI sebagai jembatan antara komputer dan modul arduino untuk melakukan *upload* program yang telah dibuat.



Gambar 4.10 Komunikasi Arduino dengan Komputer

Gambar 4.10 merupakan komunikasi arduino dengan komputer menggunakan modul FTDI, FTDI digunakan untuk mengunggah program yang dibuat pada arduino IDE ke perangkat arduino pro mini.

4.2.3.1 Implementasi Algoritma pada Sink Node

Implementasi algoritma pada *sink node* merupakan hasil konversi dari bab perancangan menjadi *source code* yang berupa kode arduino untuk menjalankan fungsi sebagai *sink node* tersebut.

Pada sub bab ini ditunjukkan konversi dari diagram alir seluruh fungsi program pada *sink node*, berupa beberapa bagian *source code* yang menjadi fungsi pokok yang mendukung jalanya *sink node* ini, implementasi dari perangkat lunak ini seluruhnya sama dengan diagram alir yang ada pada sub bab perancangan. Berikut potongan dari masing-masing fungsi utama yang sudah dibuat menggunakan bahasa pemrograman C pada arduino.

```
1.  const byte payload = 10;     //besar payload yang dikirim
2.  union
3.  {
4.          byte ch[payload];
5.          struct
6.          {
7.                  byte id;                //id paket
8.                  byte srcAddr;       //alamat asal
9.                  byte dstAddr;       //alamat tujuan
10.                 byte lengthAddr;  //panjang alamat
11.                 byte tag;               //paket tag
12.                 boolean msg = 0;  //boolean (false = COMPLAINT /true = APPROVE)
13.         };
14. } probe;
15. ……..
16. void setup()
17. {
18.         ……..
19. }
20. void loop()
21. {
22.         ……..
23. }
```

Gambar 4.11 Inisialisasi Paket Data

Gambar 4.11 merupakan potongan kode program yang digunakan untuk melakukan inisialisasi paket data yang digunakan. Sesuai dengan perancangan paket data pada sub bab sebelumnya, paket data yang digunakan terdiri dari sebuah variabel integer yang digunakan sebagai id paket, sebagai penanda jenis paket yang dikirimkan. Kemudian 2 variabel Byte yang berfungsi sebagai penanda alamat tujuan dan alamat asal paket tersebut dikirim. Kemudian 1 variabel Byte untuk menunjukkan panjang alamat yang digunakan oleh sistem, dan terakhir 1 variabel Boolean untuk digunakan sebagai pesan *COMPLAINT* / *APPROVE* dalam proses pengalokasian alamat dari *parent* ke *child*. Kemudian seluruh variabel tersebut dibungkus menjadi sebuah *struct* lalu digabungkan dengan sebuah variabel yang berisi besar *payload* yang dikirimkan, kemudian disatukan dalam bentuk sebuah *union* yang bernama *probe*.

```
1.  void setup() {
2.    ……….
3.    Mirf.spi = &MirfHardwareSpi;
4.    Mirf.init();
5.    Mirf.setRADDR((byte *)"broadcast1");
6.    Mirf.setTADDR((byte *)"broadcast1");
7.    Mirf.payload = sizeof(probe);
8.    Mirf.channel = 20;
9.    Mirf.config();
10.   ……..
11. }
```

Gambar 4.12 Konfigurasi pada Modul Radio Frequency

Gambar 4.12 merupakan potongan kode program yang digunakan untuk melakukan konfigurasi pada modul *radio frequency* nRF24L01. Pada sub bab perancangan sebelumnya telah disebutkan bahwa konfigurasi *channel* yang digunakan adalah *channel* 20. Kemudian untuk mengirimkan paket data secara *broadcast* kepada *node* lainya, maka *receiver address* dan *transmitter address* harus diset dengan nama yang sama yaitu broadcast1. Untuk besarnya *payload* dari data yang dikirimkan melalui *radio frequency* disesuaikan dengan besarnya paket data yang telah di inisialisasi sebelumnya yaitu sebesar paket data probe.

```
1.  void sendConfirm()
2.  {
3.        probe.srcAddr = nodeAddr;
4.        probe.dstAddr = 0;
5.        probe.lengthAddr = prefixLength;
6.        probe.id = 111;
7.        probe.msg = true;
8.        if ((isMax()) == 0) {
9.           Mirf.send(probe.ch);
10.          while (Mirf.isSending()) {
11.          }
12.          Serial.println("Finished sending CONFIRM packet");
```

```
13.     }
14. }
15. void setup()
16. {
17.     ……..
        sendConfirm();
    }
```

Gambar 4.13 Mengirim Paket CONFIRM

Gambar 4.13 adalah potongan kode program yang digunakan untuk mengirim paket *CONFIRM*. Paket *CONFIRM* dikirimkan dari sebuah *node* apabila node tersebut telah memiliki alamat. Namun dalam *sink node* yang menjadi titik awal dimulainya sistem alokasi pengalamatan dinamis, maka *sink node* akan langsung mengirimkan paket *CONFIRM* ketika menyala, karena alamatnya telah diset sebelumnya.

```
1.  Void loop(){
2.      if (!Mirf.isSending() && Mirf.dataReady()) {
3.          Serial.println("Got packet");
4.          if ( Mirf.dataReady() )
5.          {
6.              Mirf.getData(probe.ch);
7.              ……..
8.          }
9.      }
10. }
```

Gambar 4.14 Menerima Paket Data

Gambar 4.14 adalah potongan kode program yang digunakan untuk menerima dan membaca paket data yang masuk. Ketika *node* sedang tidak mengirim dan ada data yang masuk maka *node* akan menerima data tersebut dan meelakukan pembacaan. Dimana nanti pada step selanjutnya data yang diterima akan dibaca id nya untuk menentukan bagaimana data tersebut diproses.

```
1.  void loop () {
2.      if (!Mirf.isSending() && Mirf.dataReady()) {
3.          ……..
4.          {
5.              Mirf.getData(probe.ch);
```

```
6.      ……..
7.      if (probe.dstAddr == nodeAddr) {
8.          if (probe.id == 101) {
9.              if (arrayIsfull == index) {
10.                 probe.id = 001;
11.                 probe.dstAddr = probe.srcAddr;
12.                 probe.srcAddr = nodeAddr;
13.                 probe.msg = false;
14.                 Serial.println ("mengirim paket probe FULL ");
15.                 Mirf.send(probe.ch);
16.                 while (Mirf.isSending()) {}
17.                 Serial.println("Finished sending");
18.             }else if ((arrayAddr[probe.srcAddr] ==
                probe.srcAddr) && (arrayIsfull != index)) {
19                  probe.id = 101;
20                  probe.dstAddr = probe.srcAddr;
21.                 probe.srcAddr = nodeAddr;
22.                 probe.msg = false;
23.                 Serial.println ("mengirim paket probe
                    COMPLAINT ");
24.                 Mirf.send(probe.ch);
25.                 while (Mirf.isSending()) {}
26.                 Serial.println("Finished sending");
27.             } else {
28.                 arrayAddr[probe.srcAddr] = probe.srcAddr;
29.                 arrayIsfull = arrayIsfull + 1;
30.                 probe.id = 101;
31.                 probe.dstAddr = probe.srcAddr;
32.                 probe.srcAddr = nodeAddr;
33.                 probe.tag = probe.tag;
34.                 probe.msg = true;
35.                 Serial.println ("mengirim paket probe
                    APPROVE ");
```

```
36.                         Mirf.send(probe.ch);
37.                         while (Mirf.isSending()) {}
38.                         Serial.println("Finished sending");
39.                     }
40.                 }
41.             }
42.         }
43.     }
44. }
```

Gambar 4.15 Fungsi Memeriksa Alamat Child

Gambar 4.15 adalah potongan kode program yang digunakan untuk menerima paket data dari *child* yang berupa alamat hasil *generate*, yang dikirimkan ke *parent* untuk diperiksa apakah alamat tersebut telah digunakan atau belum. Setelah paket data masuk ke *parent* akan diperiksa id nya, apabila id nya 101 (paket *PROBE*) yang berisi alamat hasil *generate* oleh *child* maka akan diproses oleh *parent*. Kemudian *parent* akan memeriksa *array* yang berisi list alamat yang dimilikinya, apakah sudah penuh atau belum dengan cara memeriksa sebuah variabel yang menandakan jumlah alamat yang telah di setujui oleh *parent*. Apabila nilai dari variabel tersebut sama dengan jumlah alamat yang tersedia di *parent*, maka parent akan membalas paket *PROBE* tersebut dengan paket dengan id 001 (paket *FULL*) yang berarti alokasi alamat pada *parent* telah penuh.

Kemudian apabila alokasi alamat *parent* belum penuh maka parent akan memeriksa alamat yang dikirimkan oleh *child* sesuai dengan index array dari alamat tersebut contoh, alamat dengan biner 11 atau dalam desimal adalah 3 akan ditempatkan di *parent* pada array[3] (index array ke 3), begitu seterusnya. Apabila *parent* menemukan alamat tersebut sudah terisi pada index array dengan angka yang sama, maka berarti alamat tersebut sudah dialokasikan oleh *parent* untuk *child* lainya. Kemudian *parent* akan mengirimkan paket *PROBE* dengan pesan boolean false, yang menujukkan pesan *COMPLAINT* terhadap alamat yang diajukan oleh *child*.

Kemudian apabila ketika memeriksa alamat yang diajukan *child* alamat tersebut belum digunakan maka parent akan memasukkan alamat tersebut ke index array yang sesuai nilainya dengan alamat tersebut. Kemudian variabel penujuk jumlah alamat yang telah dialokasikan akan bertambah 1, lalu parent akan mengirimkan paket *PROBE* dengan pesan boolean true, yang menunjukkan pesan *APPROVE* terhadap alamat yang diajukan oleh *child* sehingga alamat tersebut dapat digunakan oleh *child*.

```
1.  void loop () {
2.      if (!Mirf.isSending() && Mirf.dataReady()) {
3.          ……..
4.          {
5.              Mirf.getData(probe.ch);
6.              ……
7.              if (probe.dstAddr == nodeAddr) {
8.                  if (probe.id == 000) {
9.                   Serial.println ("mengirim paket probe parent alive ");
10.                  probe.id = 000; //CHECK
11.                  probe.dstAddr = probe.srcAddr;
12.                  probe.srcAddr = nodeAddr;
13.                  probe.msg = true;
14.                  Mirf.send(probe.ch);
15.                  while (Mirf.isSending()) {}
16.                  Serial.println("Finished sending probe parent ALIVE");
17.                 }
18.             }
19.         }
20.     }
21. }
```

Gambar 4.16 Fungsi Menjawab Paket Check Parent

Gambar 4.16 adalah potongan kode program yang digunakan untuk memberitahu *child* bahwa *parent* masih *alive* (hidup). Fungsi dipanggil ketika *child* secara periodik dalam waktu tertentu mengirimkan paket dengan id 000 (*CHECK*) yang betujuan untuk memeriksa apakah *parent* masih hidup atau tidak. Ketika *parent* menerima sebuah paket maka akan diperiksa id nya, apabila id nya 000 (id paket *CHECK*) maka parent akan mengirimkan paket dengan id paket 000 kepada child yang mengirimkan paket *CHECK* tersebut dengan mengisi alamat tujuan dengan alamat dari *child*. Sehingga *child* dapat mengetahui bahwa *parent* masih hidup.

```
1.  Void loop(){
2.      if (!Mirf.isSending() && Mirf.dataReady()) {
3.          ……..
```

```
4.          {
5.              Mirf.getData(probe.ch);
6.              ……….
7.              if ((probe.id == 100) && (arrayIsfull != index)&&(!isMax())) {
8.                  Serial.println ("Mengirim Paket Probe Parent Baru ");
9.                  probe.id = 110;
10.                 probe.dstAddr = probe.srcAddr;
11.                 probe.srcAddr = nodeAddr;
12.                 probe.lengthAddr = prefixLength;
13.                 probe.msg = true;
14.                 Mirf.send(probe.ch);
15.                 while (Mirf.isSending()) {}
16.                 Serial.println("Finished sending");
17.
18.             }
19.         }
20.     }
21. }
```

Gambar 4.17 Fungsi Menjawab Paket Solicitation Child

Gambar 4.17 merupakan potongan kode program yang digunakan untuk proses *solicitation* saat sebuah *child* kehilangan *parent*-nya. Dengan demikian *child* tersebut akan berusaha untuk mencari *parent* baru dengan proses *solicitation*. Proses ini dimulai dengan *child* mengirimkan paket dengan id 100 (*SOLICIT*) yang ditujukan pada node disekitarnya, kemudian calon *parent* yang mendapatkan paket ini akan memeriksa apakah alokasi alamat nya telah penuh apa belum, apabila sudah penuh, maka *parent* akan mengabaikan paket ini. Apabila alokasi alamat yang dimiliki *parent* belum penuh maka *parent* akan mengirimkan paket dengan id 110 (*NEWPARENT*) kepada *node* yang mengirimkan paket *solicitation* sebelumnya. Paket tersebut berisi alamat *parent* dan panjang alamat yang digunakan oleh sistem, sehingga *node* tersebut dapat melakukan *generate* alamat untuk mencari *parent* baru.

4.2.3.2 Implementasi Algoritma pada Sensor Node

Implementasi algortima selain pada *sink node* juga diterapkan pada sensor *node*. Apabila pada *sink node* implementasi algortima yang diterapkan agar *sink node* bertindak sebagai *parent* dalam sistem ini. Namun untuk sensor *node,* implementasi algortima yang diterapkan bertujuan agar sensor *node* tidak hanya

menjadi *child* namun juga bisa bertindak sebagai *parent* dalam sistem ini layaknya *sink node*. Karena sensor *node* juga dapat bertindak sebagai *parent* maka beberapa fungsi yang telah disebutkan di *sink node* juga akan diterapkan pada sensor *node*.

Pada sub bab ini ditunjukkan konversi dari diagram alir seluruh fungsi program pada sensor *node*, kecuali fungsi yang sebelumnya telah dibahas pada *sink node*, karena sensor node juga memiliki fungsi-fungis yang ada pada *sink node*. Implementasi dari perangkat lunak ini seluruhnya sama dengan diagram alir yang ada pada sub bab perancangan. Berikut potongan dari masing-masing fungsi utama yang sudah dibuat menggunakan bahasa pemrograman C pada arduino.

```
1.  byte genAddr(int bitLength) {
2.      int power = 1 << bitLength;
3.      byte generate = random(power);
4.      Serial.print ("alamat hasil generate : ");
5.      Serial.println (generate, BIN);
6.      return generate;
7.  }
```

Gambar 4.18 Fungsi Generate Alamat pada Child

Gambar 4.18 merupakan potongan kode program yang digunakan untuk melakukan *generate* alamat pada child. Ketika *child* telah menerima panjang alamat dari *parent* maka *child* akan memanggil fungsi ini. Fungsi ini berjalan dengan 1 parameter yaitu panjang alamat dalam bit, yang kemudian akan dijadikan acuan untuk menggeser bilangan 1 sebanyak n bit panjang alamat. Kemudian hasilnya digunakan untuk menentukan batasan bilangan dalam proses me-*random* alamat. Setelah itu alamat hasi *random* akan dikembalikan ke fungsi yang memanggilnya.

```
1.  void noAddress() {
2.    byte data[Mirf.payload];
3.    if (!Mirf.isSending() && Mirf.dataReady()) {
4.      Serial.println("Got packet");
5.      If ( Mirf.dataReady() )
6.      {
7.        Mirf.getData(probe.ch);
8.        if (probe.id==111) {
9.          ……..
```

```
10.             incomingAddr = probe.srcAddr;
11.             ……..
12.             incomingLengthAddr = probe.lengthAddr;
13.             ……..
14.             nodeAddr = doAllocation(incomingAddr, incomingLengthAddr);
15.             if (nodeAddr == 0) {
16.               solicitOtherparent();
17.             }
18.             ……..
19              haveAddress = 1;
20              counter = 0;
21.             sendConfirm();
22.           }
23.         }
24.     }
25. }
```

Gambar 4.19 Fungsi Menerima Paket CONFIRM pada Child

```
1.  byte doAllocation(byte incomingAddr, int incomingLengthAddr) {
2.     byte tempAddr, addr;
3.     Serial.println ("Doing Address Alocation");
4.     prefixAddr = incomingAddr;
5.     while (1) {
6.        Serial.println("Generate Alamat....");
7.        Serial.println(" ");
8.        tempAddr = genAddr(incomingLengthAddr);
9.        Serial.println("Mengirim Probe ke Parent");
10.       Serial.println(" ");
11.       probe.id = 101;
12.       probe.srcAddr = tempAddr;
13.       probe.dstAddr = incomingAddr;
14.       byte randomTag = random(255);
15.       probe.tag = randomTag;
16.       Mirf.send(probe.ch);
```

```
17.         while (Mirf.isSending()) {
18.         }
19          Serial.println("Finished sending probe");
20          delay(3000);
21.         do {
22.           if (!Mirf.isSending() && Mirf.dataReady()) {
23.             Serial.println("Got packet");
24.             Mirf.getData(probe.ch);
25.           }
26.         } while ((probe.id != 101) && (probe.id != 001));
27.         Serial.print ("MSG Probe Dari Parent : ");
28.         if ((probe.msg == false) && (probe.id == 001)) {
29.           Serial.println("FULL");
30.           addr = 0;
31.           return addr;
32.         } else if ((probe.msg == false) && (probe.id == 101)) {
33.           Serial.println("COMPLAINT");
34.         } else {
35.           Serial.println("APPROVE");
36.           addr = (prefixAddr << 2) | tempAddr;
37.           Serial.print("Alamt Node Yang Telah Disetujui : ");
38.           Serial.println(addr, BIN);
39.           return addr;
40.       }
41.     }
42.   }
```

Gambar 4.20 Fungsi Pengalokasian Alamat pada Child

Proses sebuah *node* untuk mendapatkan alamat dimulai dari ketika *child* telah menerima paket *CONFIRM* dari *parent*. Kemudian setelah menerima paket tersebut *child* akan membaca isi data berupa alamat parent dan panjang alamat dari paket tersebut dan memasukanya ke variabel incomingAddr dan incomingLengthAddr seperti terlihat pada gambar potongan program 4.19. Kemudian kedua variabel tersebut dikirimkan ke fungsi doAllocation yang berfungsi melakukan proses alokasi alamat pada *child* seperti pada gambar 4.20.

Fungsi ini akan menyimpan alamat dari *parent* sebagai alamat *prefix* kemudian melakukan proses generate alamat dengan menjalangkan fungsi genAddr dengan mengirimkan panjang alamat yang digunakan, untuk dijadikan acuan dalam melakukan *generate* alamat secara *random*. Setelah fungsi *generate* alamat telah berjalan dan menghasilkan sebuah alamat, *child* akan mengirimkan alamat tersebut dengan sebuah paket *PROBE* dengan paket id 101. Setelah itu *child* akan menunggu balasan dari parent apakah alama tersebut dapat digunakan atau tidak. Apabila *parent* menjawab dengan paket *COMPLAINT* maka proses perulangan tidak akan berhenti dan *child* akan terus melakukan *generate* alamat sampai alamat yang di *generate* di setujui. Namun apabila *parent* menjawab dengan paket *APPROVE* maka alamat tersebut akan digabungkan dengan alamat *prefix* dan dimasukkan ke variabel nodeAddr yang menjadi penampung alamat *node child*. Dan apabila *parent* menjawab dengan pake *FULL* sebagai tanda alokasi alamat *parent* telah penuh maka *child* akan melakukan proses *solicitation* untuk mencari *parent* lainya.

```
1.  int counter = 0;
2.  void loop() {
3.
4.    if (haveAddress == 0) {
5.      ……..
6.    }
7.    } else {
8.      byte data[Mirf.payload];
9.      if (!Mirf.isSending() && Mirf.dataReady()) {
10.       ……
11.     }
12.     if (counter == 5) {
13.       Serial.println("Initiate Parent Status Check");
14.       byte newParent;
15.       probe.id = 000;
16.       probe.dstAddr = prefixAddr;
17.       probe.srcAddr = nodeAddr;
18.       probe.lengthAddr = prefixLength;
19        probe.msg = true;
20        unsigned long time = millis();
21.       Mirf.send(probe.ch);
```

```
22.          while (Mirf.isSending()) {
23.          }
24.          Serial.println("Finished sending CHECK packet");
25.          while (1) {
26.              delay(1000);
27.              Serial.println("Waiting for parent status..");
28.              if (Mirf.dataReady()) {
29.                  Mirf.getData(probe.ch);
30.                  if ((probe.id == 000) && (probe.dstAddr == nodeAddr) &&
                     (probe.srcAddr == prefixAddr)) {
31.                      Serial.println("Parent Alive..");
32.                      counter = 0;
33.                      break;
34.                  }
35.              }
36.              if (( millis() - time ) > 5000 ) {
37.                  Serial.println("Parent Dead..");
38.                  do{
39.                      newParent = solicit();
40.                  }while(newParent == 0);
41.                  counter = 0;
42.                  break;
43.              }
44.          }
45.      }
46.    ..............
47.  }
48.  delay(2000);
49. }
```

Gambar 4.21 Fungsi Pengecekan Parent pada Child

Gambar 4.21 merupakan potongan kode program yang digunakan untuk melakukan pengecekan terhadap *parent* apakah hidup atau tidak. Proses ini dijalankan secara periodik dalam waktu tertentu. Fungsi ini berjalan dalam sebuah fungsi loop yang dijalankan terus menerus. Kemudian dengan

memanfaatkan sebuah variabel yang menghitung berapa kali fungsi *loop* ini berjalan, fungsi pengecekan di atur untuk melakukan pengecekan terhadap *parent* setiap 5 kali fungsi *looping* ini dijalankan, dengan asumsi *delay* yang diset adalah 2000 ms untuk setiap kali eksekusi fungsi *loop*, maka kurang lebih fungsi ini akan berjalan tiap 10 detik. Pengecekan akan dimulai dengan mengirimkan sebuah paket *CHECK* dengan paket id 000, paket ini dikirimkan menuju *parent*. Apabila *parenti* membalas paket ini sebelum waktu timeout, maka *child* akan meneruskan eksekusi fungsi loop dan mereset variabel counter menjadi 0. Namun apabila dalam kurun waktu lebih dari 5000 ms *child* tidak menerima jawaban dari paket *CHECK*, maka *child* akan memulai proses *solicitation* untuk mencari *parent* baru dan mendapatkan alamat baru. Apabila proses *solicitation* ini gagal maka akan terus diulang sampai *child* ini mendapat alamat baru dari *parent* yang baru.

```
1.  byte solicit() {
2.      Serial.println("Initiate SOLICITATION sequence");
3.      probe.id = 100;
4.      probe.dstAddr = 0;
5.      probe.srcAddr = nodeAddr;
6.      probe.lengthAddr = prefixLength;
7.      probe.msg = true;
8.      Mirf.send(probe.ch);
9.      while (Mirf.isSending()) {
10.     }
11.     Serial.println("Finished sending SOLICITATION packet");
12.     unsigned long timeNewparent = millis();
13.     while (1) {
14.         delay(1000);
15.         Serial.println("Waiting for new parent..");
16.         if((millis()-timeNewparent) > 10000){
17.             Serial.println("Cant Found New Parent ");
18.             return 0;
19.         }
20.         Mirf.getData(probe.ch);
21.         if ((probe.id == 110) && (probe.dstAddr == nodeAddr)) {
22.             Serial.println("New Parent Found !! ");
```

```
23.             incomingAddr = probe.srcAddr;
24.             incomingLengthAddr = probe.lengthAddr;
25.             nodeAddr = doAllocation(incomingAddr, incomingLengthAddr);
26.             Serial.print("Alamat Node Baru : ");
27.             Serial.println(nodeAddr, BIN);
28.             break;
29.         }
30.     }
31. }
```

Gambar 4.22 Fungsi Solicitation pada Child

Gambar 4.22 merupakan potongan kode program yang digunakan untuk melakukan proses *solicitation* pada *child*. Proses ini berjalan ketika sebuah *node child* telah kelhilangan *parent* nya setelah melalui proses pengecekan *parent*. Atau ketika melakukan pemeriksaan alamat kepada *parent*, mendapat balasan paket *FULL* yang berarti alokasi alamat *parent* telah penuh sehingga harus mencari *parent* baru. Proses ini dimulai dengan mengirimkan paket *SOLICIT* dengan paket id 100 secara *broadcast* ke *node* disekitarnya. Dari paket tersebut, *node-node* disekitarnya yang masih memeliki alokasi alamat yang bisa digunakan akan menjawab dengan paket *NEWPARENT* dengan paket id 110, apabila menerima balasan paket selain paket id 110, maka *child* akan mengabaikanya sampai akhirnya *timeout* dan fungsi *solicitation* akan mengembalikan nilai 0 sebagai tanda proses *solicitation* gagal. Apabila menerima paket dengan paket id 110 maka fungsi akan memproses paket tersebut dengan membaca alamat *node* calon *parent* yang dikirimkan beserta panjang alamat yang digunakan sistem. Setelah itu fungsi solicitation akan memanggil fungsi doAllocation sebagai fungsi yang melakukan pengalokasian alamat untuk *node* dengan parameter alamat dari calon *parent* dan panjang alamat yang digunakan. Setelah fungsi doAllocation berhasil mendapatkan alamat, makan *node child* telah memiliki alamat baru dengan *parent* baru.

BAB V PENGUJIAN DAN ANALISIS

Pada bab ini dilakukan proses pengujian dan analisis dari sistem yang telah dibuat. Tujuan dari dilakukanya pengujian ini adalah untuk mengetahui bahwa semua kebutuhan fungsional maupun non-fungsionalyang dirancang sebelumnya telah terpenuhi. Hal-hal yang diuji pada sistem ini yaitu mengenai fungsional dari perangkat lunak yang digunakan pada sistem tersebut.

Pengujian sistem dilakukan dengan beberapa perubahan program sesuai dengan pengujian yang dibutuhkan. Pada setiap pengujian dipersiapkan sekenario dari pengujian.

Analisis sistem dilakukan dengan membandingkan data pada pengujian dibandingkan dengan hipotesa dan ditarik kesimpulan dari setiap pengujian yang ada pada setiap sub bab.

5.1 Pengujian

Pengujian dalam penelitian ini terdiri dari beberapa proses uji coba sistem, dikarenakan untuk mengetahui apakah semua sub-sistem dapat bekerja sesuai dengan yang diinginkan. Pengujian yang dilakukan adalah pengujian mengenai fungsional dari sistem alokasi pengalamatan secara dinamis pada WSN. Pengujian fungsional adalah pengujian mengenai fungsional dari perangkat lunak pada sistem ini.

Berdasarkan metode penelitian dan perancangan pada penelitian maka dilakukan pengujian pada sistem untuk dapat menganalisis hasil yang didapatkan. Penjelasan hasil pengujian dan analisis sistem sebagai berikut.

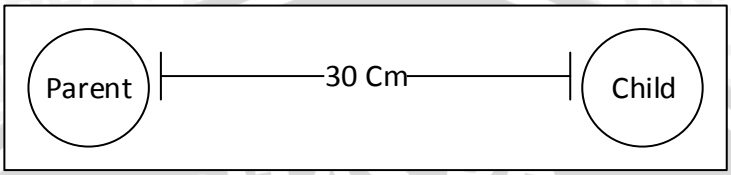
5.1.1 Pemberian Alamat Kepada Node dan Waktu Untuk Menjadi Parent

Pengujian pada bagian ini akan dilakukan terhadap fungsi sistem dalam memberikan alamat dari sebuah *node* kepada *node* lainya ketika jaringan WSN mulai menyala pertama kalinya. Ketika sebuah WSN menyala untuk pertama kalinya maka seluruh *node* yang ada pada WSN akan menyala secara bersamaan dan mendapatkan alamat dari *sink node*.

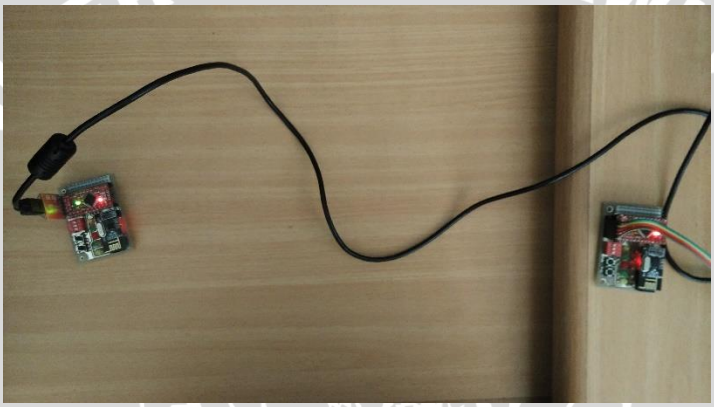
5.1.1.1 Skenario Pengujian

Pengujian pemberian alamat terhadap *node*, dilakukan dengan menggunakan 2 node yang berjarak sekitar 30 cm. satu *node* akan bertindak sebagai *parent* dan satu *node* lainya bertindak sebagai *child*. Kedua *node* tersebut akan dinyalakan, kemudian tombol *reset* pada masing-masing Arduino akan ditekan secara bersamaan sehingga seolah-olah kedua *node* tersebut menyala secara bersamaan. Setelah itu dilakukan pengamatan melalui *serial monitor* untuk melihat proses pada *node* dalam mendapatkan alamat serta diamati berapa alamat yang didapatkan dari *parent*. Selain itu di ukur juga waktu yang

dibutuhkan untuk melakukan proses pengalamatan yang diukur mulai dari saat *child* mendapatkan paket *CONFIRM* dari parent, sampai *node child* berhasil mendapatkan alamat yang disetujui oleh *parent.* Selain itu diukur juga waktu dari awal *node child* menyala sampai *node child* mengirimkan paket *CONFIRM* sebagai ukuran berapa waktu yang dibutuhkan untuk sebuah *node* menjadi *parent* bagi *node* berikutnya.



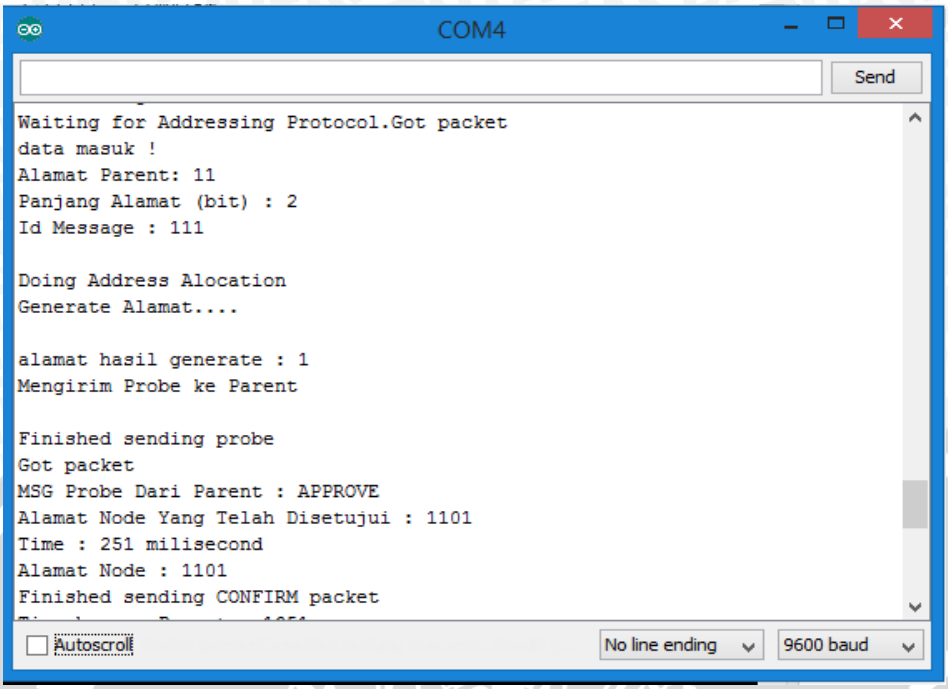
Gambar 5.1 Gambaran Skenario Pengujian 2 Node



Gambar 5.2 Penggunaan 2 Node pada Pengujian

5.1.1.2 Hasil Pengujian

Hasil dari pengujian pemberian alamat dari *parent* kepada *child* berupa alamat yang didapatkan oleh *node child* dan waktu yang dibutuhkan untuk mendapatkan alamat tersebut dari *parent*.



Gambar 5.3 Pengujian Pemberian Alamat kepada Node

Gambar 5.3 merupakan salah satu hasil pengujian pemberian alamat terhadap *child node* yang dimana mendapatkan alamat 1101 dengan waktu 251 millisecond dan waktu yang diutuhkan untuk node tersebut menjadi parent adalah 1287 millisecond. Hasil pengujian dari pemberian alamat ini dapat dilihat pada tabel 5.1 untuk lebih jelasnya.

Tabel 5.1 Hasil Pengujian Pemberian Alamat

No Percobaan	Alamat yang didapat (Binary)	Waktu (ms)	Waktu menjadi parent (ms)
1	1101	251	1287
2	1101	251	1263
3	1100	251	1294
4	1111	252	1303
5	1101	251	1304
6	1111	252	1328
7	1110	252	1314
8	1100	251	1306
9	1101	251	1321
10	1100	251	1320
	Rata-rata	251.3	1304

5.1.1.3 Analisis

Pada pengujian ini alamat yang telah diset sebelumnya pada *parent node* adalah 11. Kemudian *parent* mengirimkan sebuah *CONFIRM* paket yang berisi alamat *parent* dan panjang alamat yang digunakan sebesar 2 bit. Alamat yang di *generate* oleh *child* secara *random* yang kemudian disetujui oleh *parent*.

Waktu yang ada pada tabel 5.1 menujukkan bahwa untuk memperoleh sebuah alamat dari *parent* dibutuhkan sekitar 251 millisecond. Dari tabel tersebut juga terlihat bahwa dengan alamat berapapun yang di-*generate* oleh *child*, waktu yang dibutuhkan agar alamat tersebut dapat digunakan oleh *child* berkisar di 251 millisecond. Selain itu ditunjukkan juga bahwa waktu yang dibutuhkan untuk sebuah *node* dari mulai menyala sampai menjadi parent bagi node lainya adalah berkisar 1263 sampai 1328 millisecond dalam 10 kali percobaan.

Dari pengujian ini dapat disimpulkan bahwa waktu yang dibtuhkan oleh sebuah *node* untuk mendapatkan alamat dari *parent* sekitar 251.3 millisecond dari 10 kali percobaan. Waktu ini dihasilkan berdasarkan perhitungan waktu mulai dari saat *child* menerima paket *CONFIRM* sampai alamat yang diajukan oleh *child* disetujui oleh *parent*-nya. Sedangkan waktu yang dibutuhkan untuk sebuah *node* dapat menjadi *parent* bagi *node* lainya mulai dari node menyala adalah 1304 millisecond dari 10 kali percobaan.

5.1.2 Pemberian Alamat Kepada Node Baru dalam Jaringan

Pengujian pada bagian ini akan dilakukan terhadap fungsi sistem dalam memberikan alamat dari sebuah *node* kepada *node* lainya yang baru bergabung dengan jaringan. Hal ini terjadi ketika adanya penambahan *node* dalam WSN, baik itu karena penggantian *node* atau adanya *node* yang bergerak dan berpindah dari sebuah jaringan WSN ke jaringan WSN lainya.

5.1.2.1 Skenario Pengujian

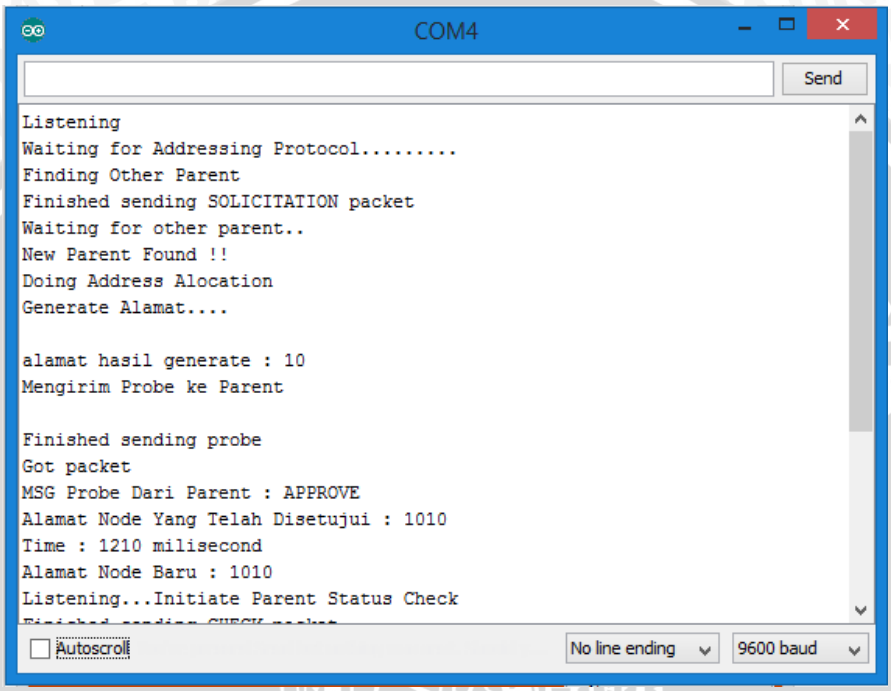
Pengujian pemberian alamat kepada *node* yang baru masuk kedalam jaringan WSN, dilakukan dengan menggunakan skenario yang sama dengan pengujian pada sub bab sebelumnya dimana terdapat 2 *node* yang berjarak sekitar 30 cm antara satu dan lainya. Node pertama bertindak sebagai *parent* kemudian *node* kedua bertindak sebagai *child*. Pada pengujian ini sebelumnya alamat pada *node parent* telah di mendapatkan terlebih dahulu alamatnya melalui proses pemberian alamat.

Kemudian *node parent* akan masuk ke mode *listening* setelah mengirimkan paket *CONFIRM*, untuk mendengarkan adanya *node-node* yang baru bergabung dengan jaringan. Kemudian setelah sekitar 5 detik *node child* akan dinyalakan untuk meminta sebuah alamat kepada *node parent* yang telah masuk ke mode *listening* dengan proses *solicitation*. Kemudian akan dilakukan pengamatan melalui serial monitor untuk mengetahui apakah *parent node* berhasil memberikan alamat kepada *child node* yang baru bergabung dan akan

dihitung berapa waktu yang dibutuhkan untuk memberikan alamat dari sebuah *parent node* kepada *child node* yang baru bergabung dengan jaringan WSN.

5.1.2.2 Hasil Pengujian

Hasil dari pengujian pemberian alamat dari *parent* kepada *child* yang baru bergabung ke dalam jaringan berupa alamat yang didapatkan oleh *node child* dan waktu yang dibutuhkan untuk mendapatkan alamat tersebut dari *parent*.



Gambar 5.4 Pengujian Pemberian Alamat Kepada Node Baru dalam Jaringan

Gambar 5.4 merupakan salah satu hasil pengujian pemberian alamat terhadap *child node* yang baru bergabung dalam jaringan, dimana mendapatkan alamat 1010 dengan waktu 1210 milisecond. Hasil pengujian dari pemberian alamat ini dapat dilihat pada tabel 5.2 untuk lebih jelasnya.

Tabel 5.2 Hasil Pengujian Pemberian Alamat Kepada Node Baru dalam Jaringan

No Percobaan	Alamat yang didapat (Binary)	Waktu (ms)
1	1001	1209
2	1000	1209
3	1000	1209
4	1010	1210
5	1000	1209
6	1011	1210
7	1010	1210
8	1010	1210
9	1001	1209
10	1010	1210
	Rata-rata	1209.5

5.1.2.3 Analisis

Pada pengujian ini alamat yang telah diset sebelumnya pada *parent node* adalah 10. Kemudian *parent* yang telah mengirimkan paket *CONFIRM* memasuki mode listening untuk mendengarkan adanya paket *solicitation*. Kemudian *child* yang menyala 5 detik kemudian melakukan proses meminta alamat dengan didauhului paket *SOLICIT*.

Waktu yang ada pada tabel 5.2 menujukkan bahwa untuk memperoleh sebuah alamat dari *parent* dengan proses *solicitation* dibutuhkan sekitar 1209 millisecond. Dari pengujian ini dapat disimpulkan bahwa waktu yang dibtuhkan oleh sebuah *node* yang baru bergabung ke dalam jaringan, untuk mendapatkan alamat dari *parent* yang ada di sekitarnya dengan proses *solicitation* rata-rata sekitar 1209.5 millisecond. Waktu ini dihasilkan berdasarkan perhitungan waktu mulai dari saat *child* mengirim paket *SOLICIT* sampai alamat yang diajukan oleh *child* disetujui oleh *parent*-nya.

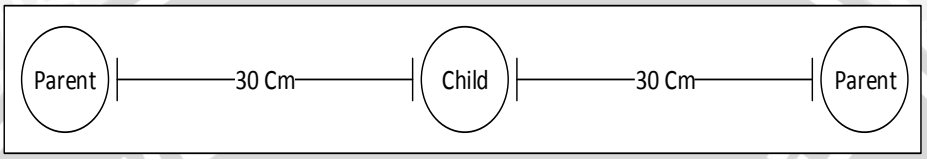
5.1.3 Pemberian Alamat Kepada Node yang Kehilangan Parent

Pengujian pada bagian ini akan dilakukan terhadap fungsi sistem dalam memberikan alamat terhadap *node* yang kehilangan *parent*. Terkadang dalam jaringan WSN adanya *down* pada *node* yang disebabkan kerusakan maupun kehabisan daya. Sehingga untuk *node* yang menjadi *child* dari *node* tersebut harus mencari *parent* baru untuk mendapatkan alamat.

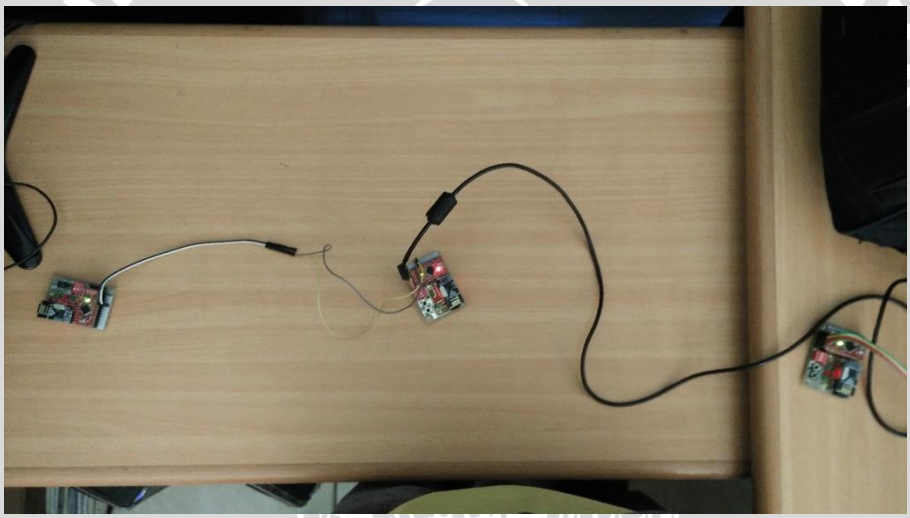
5.1.3.1 Skenario Pengujian

Pengujian pemberian alamat kepada *node* yang telah kehilangan *node parent* nya dilakukan dengan menggunakan 3 *node*. Dimana 2 *node* berperan

sebagai *parent* kemudian 1 *node* berperan sebagai *child*. Sebelumnya kedua *node parent* telah ditentukan alamat nya yaitu 10 dan 11. Kemudian ketiga *node* tersebut dinyalakan secara bersamaan, hingga *node child* mendapatkan alamat dari salah satu *node parent* yang ada. Ketika sudah mendapatkan alamat dari sebuah *node parent*, *node parent* yang menjadi parent bagi *node child* tersebut kemudian dimatikan untuk mensimulasikan sebuah *node* yang kehilangan *parent*. Kemudian dengan serial monitor diamati apakah *node child* tersebut mendapatkan alamat baru dari *node parent* lainya beserta waktu yang dibutuhkan untuk mendapatkan alamat tersebut.



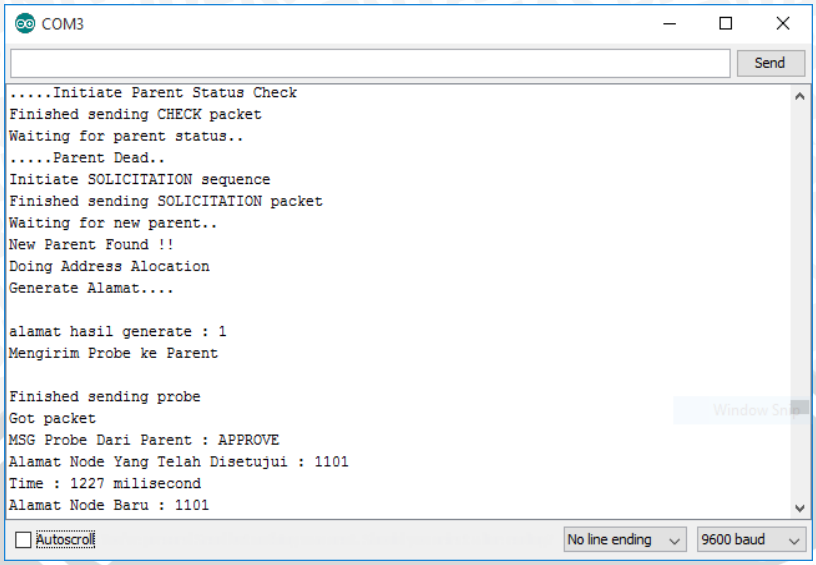
Gambar 5.5 Gambaran Skenario Pengujian dengan 3 Node



Gambar 5.6 Pengujian dengan Menggunakan 3 Node

5.1.3.2 Hasil Pengujian

Hasil dari pengujian pemberian alamat kepada *node child* yang kehilangan *parent*-nya berupa alamat yang didapatkan saat menjadi *child* dari *parent* pertama dan alamat saat menjadi *child* dari *parent* kedua beserta waktu perpindahan dari satu *parent* ke *parent* lainya.



Gambar 5.7 Pengujian Pemberian Alamat Kepada Node yang Kehilangan Parent

Gambar 5.7 merupakan salah satu hasil pengujian pemberian alamat terhadap *child node* yang kehilangan *parent*-nya, dimana *child* mendapatkan alamat 1101 dengan waktu 1227 milisecond. Hasil pengujian dari pemberian alamat ini dapat dilihat pada tabel 5.3 untuk lebih jelasnya

Tabel 5.3 Hasil Pengujian Pemberian Alamat Kepada Node yang Kehilangan Parent

No Percobaan	Alamat awal yang didapat (Binary)	Alamat Baru yang didapat (Binary)	Waktu (ms)
1	1010	1100	1227
2	1011	1101	1227
3	1100	1011	1228
4	1001	1100	1227
5	1000	1110	1228
6	1110	1010	1227
7	1111	1000	1227
8	1110	1010	1228
9	1011	1100	1227
10	1100	1011	1228
		Rata-rata	1227.4

5.1.3.3 Analisis

Pada pengujian ini alamat pada kedua *parent* telah diset sebelumnya. Kemudian *node parent* yang telah mmberikan alamat kepada *child* dimatikan. Kemudian *child* yang telah kehilangan *parent*-nya memulai sebuah proses *solicitation* dengan mengirimkan paket *SOLICIT* kemudian *parent* yang baru menjawab dengan mengirimkan alamatnya dan panjang alamat yang digunakan. Setelah itu *node child* mendapatkan alamat yang disetujui oleh *parent* yang baru.

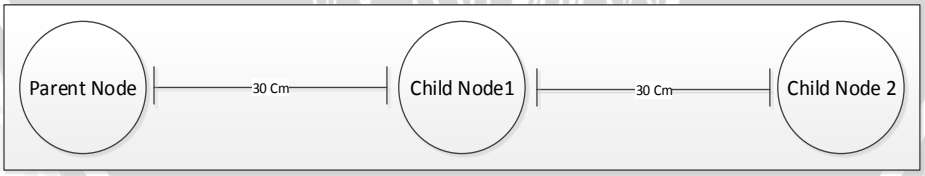
Waktu yang ada pada tabel 5.3 menujukkan bahwa waktu yang dibutuhkan oleh node child untuk berpindah dari sebuah node parent ke node parent lainya berkisar antara 1227 sampai 1228 millisecond. Dari pengujian ini dapat disimpulkan bahwa waktu yang dibutuhkan oleh sebuah *node child* yang baru kehilangan parent-nya kemudian mencari parent baru dan mendapatkan alamat dengan proses *solicitation* rata-rata sekitar 1227.4 millisecond dari 10 kali percobaan. Waktu ini dihasilkan berdasarkan perhitungan waktu mulai dari saat *child* telah kehilangan *parent*-nya sampai alamat yang diajukan oleh *child* disetujui oleh *parent* baru dan mendapatkan alamat baru tersebut.

5.1.4 Pemberian Alamat Kepada Child dan Child pada Level Bawahnya

Pengujian pada bagian ini akan dilakukan terhadap fungsi sistem dalam memberikan alamat terhadap *child node* kemudian *child node* yang telah mendapatkan alamat akan memberikan alamat kepada *child node* yang berada pada level dibawahnya.

5.1.4.1 Skenario Pengujian

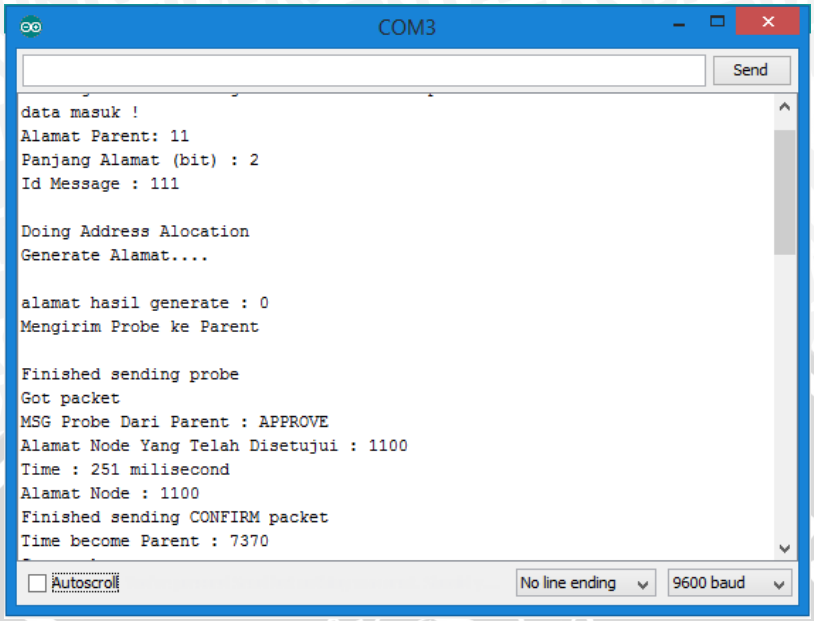
Pengujian pemberian alamat kepada 2 *child node* yang berbeda level menggunakan 3 *node* dimana jarak masing-masing *node* adalah kurang lebih 30 centimeter. 1 *node* berperan sebagai *parent* dan kemudian kedua *node* lainya berperan sebagai *child*. Sebelumnya *node parent* telah ditentukan alamatnya yaitu 11. Kemudian *node parent* dan *node child* 1 dinyalakan secara bersamaan, namun *node child* yang kedua dinyalakan setelah 2 detik Hal ini dimaksudkan untuk menunggu agar *node child* yang pertama mendapatkan alamat terlebih dahulu baru kemudian *child* yang ke dua mendapat paket *CONFIRM* dari *child* yang pertama. Kemudian dengan serial monitor diamati alamat yang didapatkan di kedua *node child* tersebut.



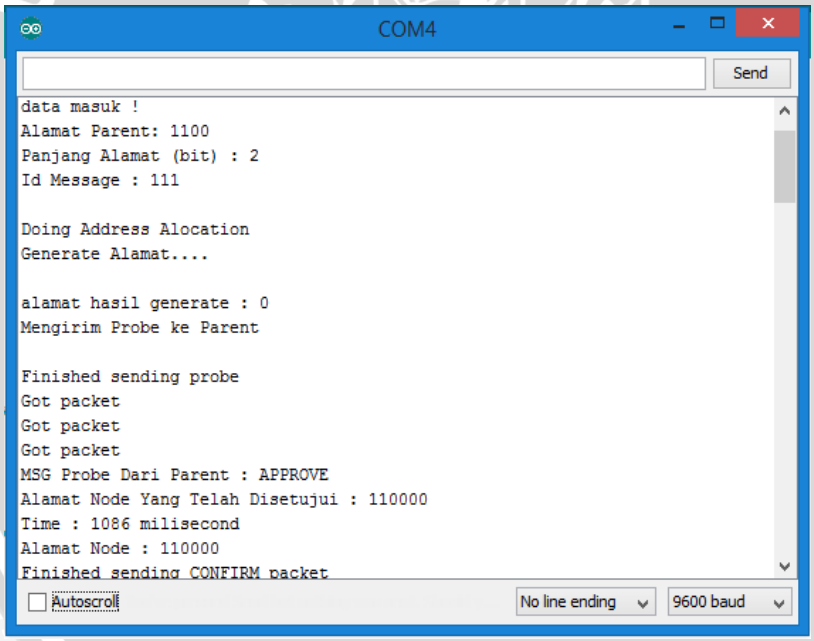
Gambar 5.8 Gambaran Skenario Pengujian dengan 1 Parent dan 2 Child

5.1.4.2 Hasil Pengujian

Hasil dari pengujian pemberian alamat kepada 2 *node child* yang berbeda level nya berupa alamat yang didapatkan pada *child* pertama dan alamat yang didapatkan oleh *child* kedua.



Gambar 5.9 Pengujian Pemberian Alamat pada Child 1



Gambar 5.10 Pengujian Pemberian Alamat pada Child 2

Pada gambar 5.9 Merupakan salah satu hasil pengujian yang dilakukan pada *child* 1 dimana alamat yang didapatkan adalah 1100. Kemudian pada gambar 5.10 Merupakan hasil pengujian selanjutnya pada *child* 2 dimana alamat yang didapatkan adalah 110000. Hasil keseluruhan dapat dilihat pada tabel 5. Untuk lebih jelasnya.

Tabel 5.4 Hasil Pengujian Pemberian Alamat Kepada Child dan Child pada Level Dibawahnya

Pengujian Pemberian Alamat Kepada Child dan Child pada Level Dibawahnya		
No Percobaan	**Alamat Child 1**	**Alamat Child 2**
1	1101	110101
2	1101	110110
3	1111	111101
4	1100	110011
5	1101	110110
6	1100	110010
7	1101	110101
8	1111	111111
9	1110	111000
10	1100	110000

5.1.4.3 Analisis

Pada pengujian ini alamat *parent* sebelumnya telah di tentukan yaitu 11. Kemudian selanjutnya proses pemberian alamat dilakukan pada *child* 1 dimana alamat yang didapatnya adalah 1100 yaitu gabungan dari alamat *parent* 11 dan alamat hasil *generate* yang telah disetujui yaitu 00. Setelah *child* 1 mendapatkan alamat, *child* 1 mengirimkan paket *CONFIRM* kepada *child* 2. *Child* 2 yang menerima paket tersebut kemudian melakukan *generate* alamat dimana alamat yang didapatnya adalah 110000 yaitu gabungan dari alamat *child* 1 1100 dan alamat hasil *generate* yang telah disetujui yaitu 00.

5.2 Analisis Keseluruhan Sistem

Pada sub bab ini dilakukan analisis dari data yang sudah didapat dalam sub bab pengujian, data di analisis dan dicari kesimpulan untuk mendapat hasil pengimplementasian yang tepat.

Secara global sistem yang dirancang sudah dapat menjalankan fungsi-fungsinya dengan baik, dari data pengujian pada keempat sub bab (5.1.1 – 5.1.4) didapat kesimpulan, sistem mampu memberikan alamat dari *node* satu ke *node* lainya dalam berbagai kondisi perubahan dalam topologi jaringan WSN. Hal ini dibuktikan dengan pengujian pada sub bab 5.1.1 dimana saat topologi WSN tetap kemudian sistem WSN dinyalakan, *sink node* yang berperan sebagai *parent* dapat memberikan alamat kepada *node* lainya yang bertindak sebagai *child*. Kemudian pengujian pada sub bab 5.1.2 dimana saat ada penambahan node dalam topologi WSN, node yang baru bergabung dengan jaringan WSN bisa mendapatkan alamat dari node yang ada disekitarnya. Kemudian pengujian pada sub bab 5.1.3 dimana saat terjadi perubahan pada topologi WSN dengan matinya salah satu *node*

parent, node child yang berkaitan dengan *node parent* yang mati tersebut dapat mencari *parent* baru dan mendapatkan alamat baru dari *parent* tersebut. Pada sub bab 5.1.4 sistem juga dapat memberikan alamat kepada 2 *child* yang berbeda *level* dan memberikan alamat sesuai dengan hirarki pengalamatan yang berbentuk *tree*.

Waktu dalam pengujian di setiap sub bab menunjukkan performa sistem dalam memberikan alamat dalam setiap kondisi perubahan topologi pada WSN. Dimana sistem memberikan alamat kepada sebuah *node* ketika jaringan mulai *up* dengan waktu rata-rata 251.3 millisecond, dengan waktu yang dibutuhkan setiap *node* agar bisa memberikan alamat ke node lainya rata-rata 1304 millisecond. Kemudian waktu yang dibutuhkan untuk memberikan alamat ketika adanya penambahan *node* dalam jaringan WSN rata-rata 1209.5 millisecond. Sementara waktu yang dibutuhkan untuk memberikan alamat kepada sebuah node *child* ketika terjadi *parent node*-nya mati rata-rata 1227.4 millisecond.

Dari semua analisis dapat diambil kesimpulan sistem dan metode yang digunakan mampu diimplementasikan secara nyata untuk memberikan alamat dalam sebuah jaringan WSN yang berbasiskan Arduino dan menggunakan modul komunikasi *radio frequency*.

BAB VI PENUTUP

Pada bab ini berisi kesimpulan dan saran yang didasrkan atas hasil pengujian dan analisis yang telah dilakukan dalam proses penelitian.

6.1 Kesimpulan

Berdasarkan rumusan masalah yang ada, hasil perancangan, implementasi dan pengujian yang dilakukan, maka dapat diambil kesimpulan sebagai berikut:

1. Sistem yang dikerjakan dapat saling berkomunikasi satu sama lain dan memberikan alamat dari satu *node* ke *node* lainya. Terbukti pada pengujian pemberian alamat, *node* yang ada mendapat alamat dari *node* lainya yang bertindak sebagai *parent* ketika jaringan WSN menyala dengan rata-rata waktu yang dibutuhkan 251.3 ms untuk mendapatkan alamat dari *parent*, sedangkan rata-rata waktu yang dibutuhkan dari awal *node* menyala hingga dapat menjadi *parent* adalah 1304 ms.
2. Sistem mampu memberikan alamat terhadap node yang baru bergabung dengan jaringan WSN melalui proses solicitation. Terbukti pada pengujian *node* baru mendapatkan alamat dari *node* yang ada disekitarnya yang dipilih menjadi *parent*. Pemberian alamat terhadap node baru tersebut membutuhkan rata-rata waktu 1209.5 ms.
3. Sistem mampu memberikan alamat kepada *node* yang menjadi *child* yang *parent*-nya *down* atau mati sehingga sehingga *child* kehilangan parent nya. Node yang kehilangan *parent* bisa mendapatkan alamat kembali dari proses *solicitation* kepada *node-node* yang telah memiliki alamat disekitarnya, waktu yang dibutuhkan untuk proses *solicitation* ini rata-rata 1227.4 ms.
4. Jumlah *node* yang dapat ditangani oleh sistem adalah sebanyak 2^8 *node* atau sekitar 256 node. Jumlah ini didapat dari jumlah maksimal panjang bit alamat yang diterapkan yaitu 8 bit. Kemudian jumlah *child* yang dapat dimiliki tiap *node parent* terbatas hanya 4 *node*.

6.2 Saran

Saran untuk pengembangan lebih lanjut pemilihan parent dapat dilakukan dengan memilih *parent* berdasarkan tingkat sinyal komunikasi yang paling baik maupun dengan parameter lainya seperti jarak. Sehingga pemilihan *parent* bisa lebih baik daripada dengan mengambil *parent* yang pertama kali mengirimkan paket *CONFIRM* nya.

DAFTAR PUSTAKA

[1] J. Elson and D. Estrin, "An Address-Free Architecture for Dynamic Sensor Networks," *University of Southern California,* pp. 1-2, 2000.

[2] S. PalChaudhuri, S. Du, A. K. Saha and D. B. Johnson, "TreeCast A Stateless Addressing and Routing Architecture for Sensor Network," *International Parallel and Distributed Processing Symposium,* pp. 1-8, 2004.

[3] F. Ye and R. Pan, "A Survey od Addressing Algorithms for Wireless Sensor Networks," *Wireless Communications, Networking and Mobile Computing, 2009. WiCom '09. 5th International Conference,* pp. 1-7, 2009.

[4] K. Sohraby, D. Minoli and T. Znati, Wireless Sensor Networks Technology, Protocols, and Applications, New Jersey: John Wiley & Sons, Inc., 2007.

[5] G. Gridling and B. Weiss, Introduction to Microcontrollers, Vienna: Vienna University of Technology Institute of Computer Enginering Embedded Computing Systems Group, 2007.

[6] F. Djuandi, "Pengenalan Arduino," p. 1, 8 January 2011.

[7] "ArduinoBoardProMini," 23 Juli 2015. [Online]. Available: http://arduino.cc/en/Main/ArduinoBoardProMini.

[8] "Preliminray Product Spesification nRF24L01," Nordic Semiconductor, Tonstad, 2006.