IMPLEMENTASI TEKNIK AGREGASI DATA MENGGUNAKAN *CLUSTER BASED ALGORITHM* UNTUK PENGIRIMAN DATA SENSOR PADA *WIRELESS SENSOR NETWORK* (WSN)

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Annisa Widuri Murti Utami
NIM: 165150200111106

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2021

PENGESAHAN

IMPLEMENTASI TEKNIK AGREGASI DATA *MENGGUNAKAN CLUSTER BASED ALGORITHM* UNTUK PENGIRIMAN DATA SENSOR *PADA WIRELESS SENSOR NETWORK* (WSN)

SKRIPSI

Diajukan untuk memenuhi sebagaian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh:
Annisa Widuri Murti Utami
NIM: 165150200111106

Skripsi ini telah diuji dan dinyatakan lulus pada
30 Juni 2021

Telah diperiksa dan disetujui oleh

Dosen Pembimbing I	Dosen Pembimbing II
Dany Primanita Kartikasari, S.T., M.Kom.	Adhitya Bhawiyuga, S.Kom., M.Sc.
NIP: 19771116 200501 2 003	NIK: 19890720 201803 1 002

Mengetahui,
Ketua Jurusan Teknik Informatika

Achmad Basuki, S.T., M.MG., Ph.D.
NIP. 19741118 200312 1 002

PERNYATAAN ORISINALITAS

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

Apabila ternyata di dalam 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, 30 Juni 2021

Annisa Widuri Murti Utami

NIM: 165150200111106

ABSTRAK

Annisa Widuri Murti Utami, Implementasi Teknik Agregasi Data Menggunakan *Cluster Based Algorithm* untuk Pengiriman Data Sensor pada *Wireless Sensor Network (WSN)*

Pembimbing: Dany Primanita Kartikasari, S.T., M.Kom. dan Adhitya Bhawiyuga, S.Kom., M.Sc.

Pertanian merupakan sebuah seni dan ilmu yang digunakan untuk mengolah tanah, menanam tanaman, dan memelihara ternak. Lahan pertanian yang ada di Indonesia memiliki kualitas tanah yang berbeda-beda di setiap daerahnya. Kualitas tanah untuk pertanian dapat berdampak pada tanaman hasil pertanian. Tanah yang diolah dengan tepat akan memiliki kualitas tanah yang lebih subur dibandingkan dengan tanah yang tidak diolah dengan tepat. Untuk mengetahui kualitas tanah dari lahan pertanian dapat diketahui dengan salah satu faktor yaitu dengan kelembapan tanah. Maka dari itu untuk mempermudah mengetahui kelembapan tanah dibuatlah sebuah *Wireless Sensor Network* (WSN) yang dapat membantu memantau kondisi kelembapan tanah. Pada WSN ini menerapkan *Cluster Based Algorithm* dengan protokol yang digunakan adalah *Low Energy Adaptive Clustering Hierarchy* (LEACH). WSN akan melakukan pencarian *cluster head* lalu node *sensor* akan mengirimkan hasil *data sensing* kelembapan tanah kepada *cluster head* untuk selanjutnya akan dilakukan agregasi data dengan hasil data agregasi akan selanjutnya dikirimkan ke node *sink* lalu akan diteruskan ke *cloud.* Agregasi data yang dilakukan dapat membuat *sink* menyala lebih lama 30 menit dibandingkan dengan node *sink* yang menerima data tanpa dilakukan agregasi data. Node *sink* yang tidak menerima data agregasi juga menggunakan beban daya yang lebih besar dibandingkan dengan *sink* yang menerima data agregasi. Hasil pengujian rata-rata waktu respons untuk mencari *cluster head* baru memiliki rata-rata waktu 6023,4 *miliseconds.* Jarak maksimal antar node untuk saling mengirimkan data adalah sebesar 100 meter.

Kata kunci: *Wireless Sensor Network, Cluster Based Algorithm,* LEACH, Agregasi data

ABSTRACT

Annisa Widuri Murti Utami, *Architectural Design of Cluster-Based Wireless Sensor Networks Using Statistical Data Aggregation*

***Supervisors:* Dany Primanita Kartikasari, S.T., M.Kom. and Adhitya Bhawiyuga, S.Kom., M.Sc.**

Agriculture is an art and science used to cultivate land, grow crops, and raise livestock. Agricultural land in Indonesia has different soil quality in each region. Soil quality for agriculture can have an impact on agricultural crops. Properly cultivated soil will have a more fertile soil quality than soil that is not properly cultivated. The quality of the soil from agricultural land can be determined by one factor, soil moisture. Therefore, to make it easier to determine soil moisture, a Wireless Sensor Network (WSN) was created. This system can help to monitor the condition of soil moisture content. WSN applies a Cluster Based Algorithm and the protocol that used in WSN is a Low Energy Adaptive Clustering Hierarchy (LEACH). WSN will search for the cluster head, then the sensor node will send the results of soil moisture sensing data to the cluster head for further data aggregation. Then, the results of the aggregation data will be send to the sink node and then forwarded to the cloud Data aggregation that is performed can make the sink stay longer than 30 minutes compared to the sink node that receives data without data aggregation. Sink nodes that do not receive aggregation data also use more load current compared to sinks that receive aggregation data. The test results of the average response time to find a new cluster heads have an average time of 6023.4 milliseconds. The maximum distance between nodes to transmit data to each other is 100 meters.

Keywords: Wireless Sensor Network, Cluster Based Algorithm, LEACH, Data Aggregation

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Pertanian merupakan sebuah seni dan ilmu yang digunakan untuk mengolah tanah, menanam tanaman, dan memelihara ternak. Dalam ilmu tersebut juga mencakup persiapan produk tanaman dan hewan untuk digunakan dan didistribusikan ke pasar (Rutledge, et al., 2011). Lahan pertanian yang ada di Indonesia sendiri memiliki kualitas tanah yang berbeda-beda di setiap daerahnya. Kualitas tanah untuk pertanian dapat berdampak pada tanaman hasil pertanian. Tanah yang diolah dengan tepat akan memiliki kualitas tanah yang lebih subur dibandingkan dengan tanah yang tidak diolah dengan tepat. Tanah yang subur memiliki ciri-ciri mengandung 45% bahan organik, 5% zat anorganik, 25% air, dan 25% udara. Selain tanah yang subur, jenis tanah yang digunakan untuk menanam tanaman dari bermacam-macam jenis juga dapat mempengaruhi hasil dari pertanian. Jenis tanah yang baik dapat dilihat dari partikel tanah yang digunakan untuk sirkulasi air dan udara dari tanaman tersebut. (D, 2015)

Untuk mengetahui kualitas tanah dari lahan pertanian dapat diketahui dengan salah satu faktor yaitu dengan kelembapan tanah. Faktor kelembapan tanah juga dapat mempengaruhi tanaman hasil pertanian. Apabila kondisi kandungan kelembapan tanah cukup kering maka dapat menyebabkan pertambahan tinggi tanaman akan terhambat, selain itu juga dapat mengakibatkan tanaman akan mengalami cekaman air dimana stomata daun akan menutup dan akan mempengaruhi penurunan laju transpirasi. Sedangkan, apabila kondisi kandungan kelembapan tanah terlalu basah maka dapat meyebabkan permukaan tanah menjadi lembab dan akan mengakibatkan tumbuhnya jamur yang menjadi penyakit bagi tanaman.

Berdasarkan uraian diatas tentang kelembapan tanah, maka untuk membantu memantau kondisi kelembapan tanah dapat dibuat sebuah *Wireless Sensor Network* (WSN). *Wireless Sensor Network* (WSN) merupakan sebuah jenis jaringan yang dibuat dari sejumlah node sensor ringan berukuran kecil yang digunakan untuk tujuan *sensing* dan *monitoring*. Node sensor akan didistribusikan dalam suatu wilayah geografis. Selanjutnya, seluruh node sensor akan bekerja sama satu sama lain untuk membawa data yang telah didapatkan melalui jaringan ke node utama (Khalifeh, et al., 2020). Pada penelitian oleh Khalifeh, et al., (2020) juga menyebutkan bahwa fungsi penting dari WSN adalah untuk memantau fenomena di lingkungan fisik dan melaporkan hasil data *sensing* ke *sink*. WSN biasanya terdiri dari sejumlah perangkat sensor dengan energi baterai yang dapat digunakan tanpa pengawasan. Lalu pada WSN juga terdapat *routing,* yaitu merupakan proses meneruskan pesan atau data antar node sensor (Aung, et al., 2016). WSN ini nantinya akan mengimplementasikan metode agregasi data. Agregasi data merupakan sebuah prosedur dimana sebuah node akan melakukan pengumpulan data dari beberapa node yang lain. Hal ini dilakukan karena untuk mengurangi transmisi data. Dengan proses ini maka dapat mengurangi informasi

yang tidak dibutuhkan saat dilakukan pengumpulan data (Pourghebleh & Navimipour, 2017). Sehingga untuk menerapkan teknik agregasi pada WSN ini akan membutuhkan node *sensor* sebagai perangkat yang akan mengumpulkan data, pusat pengumpulan data (*sink)* yang juga akan digunakan sebagai perangkat yang menjalankan agregasi data dan juga *cloud* sebagai perangkat yang menampilkan data yang telah didapatkan.

Contoh *routing algorithm* yang dapat digunakan pada *Wireless Sensor Network* (WSN) adalah *flat-based routing* dan *hierarchical-based routing*. *Flat-based routing* menerapkan fungsi dan peran yang sama pada semua node, sedangkan *hierarchical-based routing* menerapkan peran yang berbeda-beda pada node-node nya. *Hierarchical-based routing* memiliki *management of nodes* dan perluasan jaringan lebih baik daripada *flat-based routing,* terutama untuk digunakan pada WSN dalam skala besar (Aung, et al., 2016). Salah satu contoh dari *hierarchical-based routing* yang dapat digunakan adalah *Cluster Based Algorithm.* Algoritma ini bekerja dengan cara node *sensor* mengirimkan data ke *Cluster Head* terdekat dan mengirimkan hasil *data sensing* kepada *sink*. *Cluster Based Algorithm* ini sangat cocok untuk digunakan pada lingkungan berskala besar, yang memiliki satu server pusat dan banyak node yang berjalan pada suatu aplikasi sesuai dengan hasil penelitian (Pourghebleh & Navimipour, 2017).

Contoh dari *routing protocol* yang dapat digunakan pada *Cluster Based Algorithm* adalah protokol LEACH atau *Low Energy Adaptive Clustering Hierarchy.* LEACH sendiri memiliki beberapa varian seperti LEACH, LEACH-C, LEACH-VH dan lainnya. LEACH menggunakan pendekatan hierarkis untuk mengatur jaringan menjadi satu set *cluster.* Setiap *cluster* akan dikelola oleh *Cluster Head* yang telah dipilih. *Cluster Head* akan bertanggung jawab untuk melakukan tugas yang telah diberikan (Sohraby, et al., 2007). LEACH dirancang untuk mengumpulkan dan mengirimkan data ke node *sink,* sehingga cocok untuk digunakan dalam penelitian ini.

Pada sistem ini memiliki tiga komponen yaitu *sensor* yang mengambil data, pusat pengumpulan data (*sink)*, dan *cloud* sebagai penerima data. *Cluster Based Algorithm* diimplementasikan pada *Wireless Sensor Network* (WSN) yang akan melakukan proses agregasi data. Protokol LEACH diimplementasikan untuk mengatur jaringan menjadi beberapa set cluster juga mengatur pengiriman data dari *sensor* ke pusat pengumpulan data (*sink*).

1.2 Identifikasi Masalah

Lahan pertanian yang cukup besar di Indonesia memiliki kualitas tanah yang berbeda-beda di tiap daerah. Kualitas tanah sendiri dapat menentukan hasil tanaman yang ditanam oleh petani. Untuk itu, pada penelitian ini akan dibuat sebuah *Wireless Sensor Network* (WSN) yang dapat membantu memantau kondisi kandungan kelembapan tanah. WSN ini akan mengimplementasikan metode agregasi data.

Teknik agregasi data merupakan sebuah teknik untuk melakukan prosedur dimana sebuah node akan melakukan pengumpulan data dari beberapa node yang lain. Hal ini dilakukan karena untuk mengurangi transmisi data. Maka dari itu untuk menerapkan Teknik agregasi data ini dibangun sebuah *Wireless Sensor Network* (WSN). WSN ini akan menerapkan sebuah *routing algorithm* dan *routing protocol.*

Pada WSN, *routing algorithm* yang akan digunakan adalah *Cluster Based Algorithm.* Pada *Cluster Based Algorithm* akan membutuhkan sebuah *routing protocol* dimana dalam protokol yang akan digunakan adalah *Low Energy Adaptive Clustering Hierarchy* (LEACH). Maka, pada sistem akan membutuhkan node *sensor* sebagai perangkat yang akan melakuakan *data sensing*, *sink* sebagai perangkat pengumpulan data dan *cloud* sebagai yang akan menampilkan hasil data.

Untuk melakukan agregasi data maka perlu dilakukan beberapa proses pada WSN. Penerapan agregasi pada WSN akan dimulai dengan node *sink* akan memilih *cluster head* lalu node *sensor* akan mengirimkan data pada *cluster head* untuk oleh *cluster head* akan dilakukan agregasi data. Hasil agregasi data pada *cluster head* akan dikirimkan kepada node *sink* untuk oleh *sink* selanjutnya akan dikirimkan pada *cloud* untuk ditampilkan.

1.3 Rumusan Masalah

1. Bagaimana hasil implementasi algoritma *Cluster Based Algorithm* pada sistem monitoring kelembapan tanah?
2. Bagaimana agregasi dilakukan pada sistem monitoring kelembapan tanah?
3. Bagaimana menampilkan data hasil dari agregasi data?

1.4 Tujuan

Tujuan dari penelitian ini adalah sebagai berikut:

1. Mengimplementasikan algoritma *Cluster Based Algorithm* untuk monitoring kelembapan tanah.
2. Mengetahui cara kerja dari agregasi data.
3. Menampilkan data kelembapan tanah secara *real-time.*

1.5 Manfaat

Manfaat dari penelitian ini adalah sebagai berikut:

1. Mempermudah untuk mengetahui data kelembapan tanah melalu sistem monitoring yang telah dibuat.
2. Menghasilkan sistem monitoring kelembapan tanah dengan menerapkan metode agregasi data.

1.6 Batasan Masalah

Batasan masalah dalam penelitian ini adalah sebagai berikut:

1. Sistem monitoring kelembapan tanah akan diterapkan dengan implementasi dua *cluster* dengan satu *cluster* dengan empat node *sensor* dan *cluster* lainnya dengan tiga node *sensor*.
2. *Cluster Based Algorithm* akan menggunakan algoritma LEACH (*Low Energy Adaptive Clustering Hierarchy*).
3. Penelitian yang dilakukan masih bersifat *prototype,* belum diterapkan pada luas lahan tertentu.

1.7 Sistematika Pembahasan

Bagian ini berisi struktur skripsi ini mulai Bab Pendahuluan sampai Bab Penutup dan deskripsi singkat dari masing-masing bab. Diharapkan bagian ini dapat membantu pembaca dalam memahami sistematika pembahasan isi dalam skripsi ini.

BAB I PENDAHULUAN

Bab pendahuluan adalah bagian yang menjelaskan latar belakang dari dilakukannya penelitian ini. Dari latar belakang ini maka akan ditemukan permasalahan yang akan digunakan sebagai rumusan masalah pada penelitian ini. Pada bab ini juga akan didapatkan tujuan, manfaat dan batasan dari penelitian ini.

BAB II LANDASAN KEPUSTAKAAN

Pada bab landasan kepustakaan terdapat pembahasan dari teori-teori dasar, metode dan model yang berkaitan dengan penelitian. Pembahasan ini akan digunakan sebagai landasan dari penelitian ini.

BAB III METODOLOGI

Pada bab metodologi menjelaskan mengenai metode, teknik dan langkah-langkah pengerjaan pada penelitian ini.

BAB IV ANALISIS KEBUTUHAN DAN PERANCANGAN SISTEM

Pada bab analisis kebutuhan perancangan sistem berisi penjelasan dari perancangan sistem pada penelitian ini. Pada bab ini juga terdapat informasi kebutuhan yang diperlukan pada penelitian ini.

BAB V IMPLEMENTASI SISTEM

Pada bab ini berisi penjelasan dari implementasi yang telah dibuat pada penelitian ini.

BAB VI PENGUJIAN SISTEM

Pada bab pengujian sistem akan dilakukan pengujian pada sistem yang telah dibuat pada penelitian ini.

BAB VII PENUTUP

Bab penutup berisi kesimpulan dan saran dari penulis. Kesimpulan didapat dari hasil pengujian dan analisis penelitian.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Penelitian dengan judul "Monitoring Kelembaban Tanah Pertanian Menggunakan *Soil Moisture Sensor* FC-28 Dan Arduino Uno" membahas tentang kesulitan monitoring tanah yang menjadi media tanam untuk tanaman hortikultura oleh para petani di kabupaten Bone Bolango Provinsi Gorontalo. Pada pembuatan *monitoring* ini dilakukan perancangan sebuah *Internet of Things (IoT)* dengan menggunakan *sensor* kelembapan tanah FC-28 untuk mengukur kelembapan tanah. Kemudian nilai kelembapan tanah akan dikirimkan kepada Arduino uno yang bertindak sebagai mikrokontroler. Dari Arduino uno nantinya akan diteruskan kembali kepada webserver (Husdi, 2018). Sebuah sistem monitoring kelembapan tanah yang dapat menghasilkan hasil kondisi tanah berhasil diterapkan dalam pembuatan *Internet of Things (IoT)*. Perangkat Arduino uno dan *Soil Moisture Sensor* ini juga akan dipakai pada penelitian yang saat ini sedang dikerjakan.

Penelitian *"Measurement and Monitoring of Soil Moisture using Cloud IoT and Android System"* menyebutkan bahwa melakukan monitoring tanah merupakan sebuah aspek yang penting dalam pertanian karena memiliki dampak langsung pada produksi dan pemeliharaan tanaman. Untuk melakukan monitoring tanah, parameter yang perlu diperhatikan adalah kadar air, suhu, kelembapan dan tingkat pH. Maka dibuat sebuah sistem monitoring yang memanfaatkan penggunaan *Cloud* dan *Android System.* Sistem menggunakan teknik *Hygrometric* untuk mengukur kelembapan tanah. Karena penelitian ini dilakukan di India maka percobaan dilakukan dengan menggunakan tanah merah dan hitam. Sistem ini dibuat dengan mengembangkan sebuah *Sensing Units.* Sistem ini menggunakan *sensor* FC-287 yang berdaya rendah untuk mengumpulkan informasi kelembapan tanah. Setelah informasi dari *sensor* terkumpul maka dilakukan komunikasi antar gateway untuk mengirimkan data yang telah diukur kepada *cloud* untuk diproses lebih lanjut (Vani & Rao, 2016). Sebuah perangkat *mobile* yang berguna untuk merekam data *real-time* menggunakan teknologi *cloud* dan aplikasi Blynk telah dihasilkan dari Vani & Rao, (2016). Sehingga pengguna dapat memantau kondisi tanah kapan saja. Penggunaan *cloud* pada sistem monitoring ini juga akan diimplementasikan pada penelitian yang saat ini sedang dikerjakan.

Pada penelitian "*Data Aggregation Algorithm for Wireless Sensor Network*" menyebutkan bahwa tujuan data agregasi adalah untuk mengurangi data yang tidak diperlukan dan mengekstrak informasi dari data yang dikumpulkan. Agregasi data dibahas dengan disebutkan bahwa pada agregasi data terdapat beberapa fungsi untuk mengekstrak informasi data yaitu dengan bantuan fungsi seperti *Max, Mean,* atau *Median.* Pada penelitian memberikan contoh *Aggregation Tree* yang diterapkan pada tanah pertanian. Pada penelitian, beberapa *sensor* disebar dibeberapa lokasi dan node *sensor* akan melakukan *data sensing.* Pada contoh penelitian yang dilakukan peneliti melakukan percobaan

dengan menggunakan fungsi *mean* untuk menjalankan proses agregasi data yang dilakukan pada *sink,* dimana peneliti mencari rata-rata dari seluruh hasil *data sensing* yang telah dilakukan. Hasil dari agregasi data yang telah dilakukan akan dikirimkan dari *sink* ke *base station* (Kaur & Munjal, 2020). Penggunaan proses agregasi dengan fungsi *mean* pada sistem monitoring ini juga akan diimplementasikan pada penelitian yang saat ini sedang dikerjakan.

Pada penelitian "Metode *Routing* Protokol LEACH pada Jaringan *Sensor* Nirkabel Studi Kasus Sistem Pemantauan Suhu dan Kelambaban Udara" membahas tentang bagaimana Jaringan Nirkabel *Sensor* (JSN) sering digunakan untuk memantau kondisi-kondisi tertentu pada lokasi atau posisi objek pemantauan yang cukup berjauhan. Untuk tujuan dari pembuatan JSN, disebutkan bahwa diharapkan pengimplementasian JSN dapat memaksimalkan pemantauan dan dapat dilakukan dalam waktu yang cukup panjang. Studi kasus yang digunakan dalam pemantauan adalah berupa suhu dan kelembapan udara dengan mengimplementasikan *routing protocol* LEACH. Sistem JSN yang dibangun mengimplementasikan *routing protocol* LEACH dan sistem terdiri dari satu buah *sink node* dan sembilan buah node *sensor* yang berperan untuk melakukan *data sensing* dan mengirimkannya ke *sink. S*istem JSN yang dibangun ini digunakan untuk melihat hasil dari *routing protocol* LEACH dan membandingan dengan sistem yang dibangun tanpa menggunakan *routing protocol* LEACH (Sumiharto, et al., 2019). Pengimplementasian *routing protocol* LEACH dan jumlah node yang digunakan oleh Sumiharto, et al., (2019) juga diimplementasikan pada penelitian yang saat ini sedang dikerjakan.

2.2 Dasar Teori

2.2.1 *Wireless Sensor Network* (WSN)

Sensor Network adalah sebuah infrastuktur yang terdiri dari elemen *sensing (measuring), computing,* dan *communication* yang memberikan kemampuan untuk menginstrumentasi, mengamati, dan bereaksi terhadap peristiwa dan fenomena di lingkungan tertentu. Lingkungan yang dapat menerapkan *Sensor Network* dapat berupa dunia fisik, sistem biologis, atau *information technology (IT) framework.*

Terdapat empat komponen dasar dalam *sensor network,* yaitu: Perakitan sensor terdistribusi atau terlokalisasi, jaringan interkoneksi (dapat berupa *wireless-based*), titik pusat pengelompokan informasi, dan satu set sumber daya komputasi pada titik pusat untuk menangani *data correlation, event trending, status querying,* dan *data mining.* Pada *sensor network,* jumlah data yang akan dikumpulkan bisa berpotensi dalam jumlah besar, sehingga metode algoritmik untuk pengelolaan data juga memiliki peran penting dalam *sensor network* (Sohraby, et al., 2007).

Menurut Khalifeh, et al., (2020), *Wireless Sensor Network* (WSN) merupakan sebuah jenis jaringan yang dibuat dari sejumlah node sensor ringan berukuran kecil yang digunakan untuk tujuan *sensing* dan *monitoring*. Node

sensor didistribusikan dalam suatu wilayah geografis. Selanjutnya, seluruh node sensor akan bekerja sama satu sama lain untuk membawa data yang telah didapatkan melalui jaringan ke node utama. Sedangkan, menurut Aung, et al., (2016), fungsi penting dari WSN adalah untuk memantau fenomena di lingkungan fisik dan melaporkan hasil data *sensing* ke *sink*. WSN biasanya terdiri dari sejumlah perangkat sensor dengan energi baterai yang dapat digunakan tanpa pengawasan. Lalu pada WSN juga terdapat *routing,* yaitu merupakan proses meneruskan pesan atau data antar node sensor. Secara umum, *routing* pada WSN dapat dibagi menjadi *flat-based routing and hierarchical-based routing*, tergantung dari struktur jaringan yang digunakan. Pada *routing* terdapat protokol yang dapat digunakan, protokol pada *routing* dapat dianggap adaptif apabila parameter sistem tertentu dapat dikontrol untuk beradaptasi dengan kondisi jaringan saat ini dan dengan tingkat energi yang tersedia.

2.2.2 *Routing Protocol*

Pada WSN terdapat *routing protocol* yang harus diterapkan. Pemilihan *routing protocol* yang akan digunakan harus mempertimbangkan keterbatasan daya dan sumber daya dari node jaringan, dan kemungkinan *packet loss* dan *delay.* Untuk *routing protocol* terdapat satu kelas protokol yang menerapkan *flat network architecture,* yaitu dimana semua node dianggap memiliki kedudukan yang sama. Lalu terdapat kelas kedua yang mengatur jaringan dalam sebuah *cluster* dimana dalam kelas ini terdapat *cluster head* yang bertanggung jawab untuk meneruskan informasi dalam sebuah *cluster.*

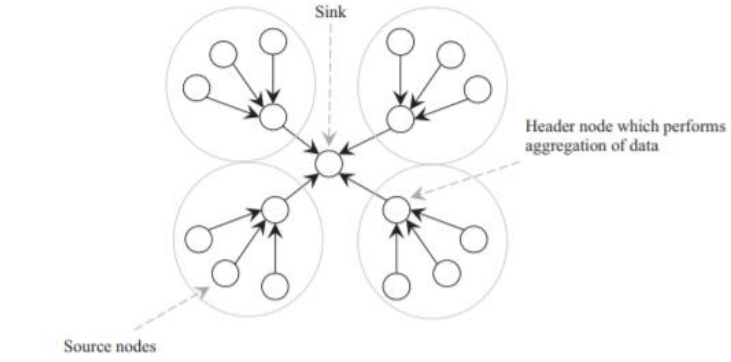
Lalu kelas ketiga dalam *routing protocol* ini menggunakan pendekatan *data-centric* yang dimana node sumber akan menanyakan atribut untuk fenomena sekitar daripada node *sensor* secara individu. Hal ini dilakukan untuk menetapkan tugas ke node *sensor* dan mengeksperisakn kueri terhadap atribut tertentu. Dan kelas terakhir, kelas keempat dari *routing protocol* menggunakan lokasi untuk mengalamatkan node *sensor. Routing protocol* berbasis lokasi berguna dalam aplikasi di mana posisi simpul dalam cakupan geografis jaringan relevan dengan kueri yang dikeluarkan oleh simpul sumber. Permintaan semacam itu dapat menentukan area tertentu di mana fenomena yang menarik dapat terjadi atau sekitar titik tertentu dalam lingkungan jaringan (Sohraby, et al., 2007).

2.2.3 *Cluster Based Algorithm*

Cluster Based Algorithm merupakan metode *Hierarchical Routing Algorithms.* Pada *Clustering,* seluruh jaringan sensor akan dibagi kedalam beberapa *cluster* atau beberapa *layers.* Transmisi antara *cluster* akan dikoordinasikan oleh tiap *Cluster Head.* Pada *clustering* menyediakan keuntungan berupa agregasi data yang dilakukan pada *cluster head* untuk meningkatkan kinerja dari WSN. Pada *clustering* terdapat dua jenis transmisi, yaitu *Single Hop Transmission* dan *Multiple Hop Transmission.*

Pada *Single Hop Transmission, cluster head* akan mengirimkan data ke *base station (sink)* secara langsung tanpa melalui *cluster head* yang lainnya. Ini adalah metode transmisi paling sederhana tanpa perlu mempertimbangkan informasi lain. Namun metode ini kemungkinan tidak cocok digunakan dalam jaringan skala besar karena ada batasan pada jarak transmisi antar sensor.

Sedangkan pada *Multiple Hop Transmission, cluster head* akan mengirimkan data kepada *cluster head* berikutnya lalu akan diteruskan hingga *base station (sink).* Metode ini dapat membagi lingkungan dengan jarak yang besar menjadi beberapa jarak yang lebih pendek untuk melakukan transmisi. Sehingga metode ini lebih cocok untuk digunakan untuk jaringan dengan skala yang lebih besar (Chan & Rudolph, 2015).

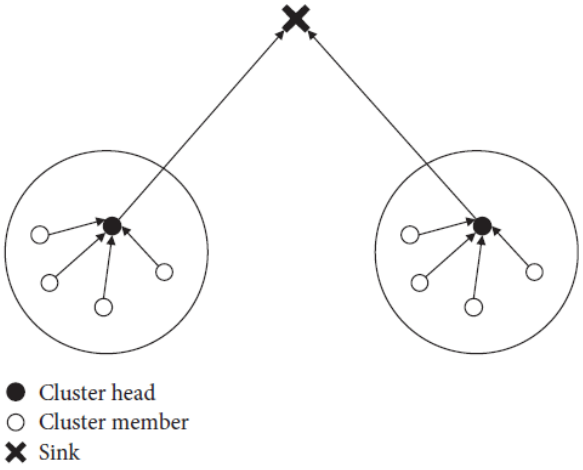


Gambar 2.1 *Cluster Based Algorithm*

Sumber: (Pourghebleh & Navimipour, 2017)

2.2.4 *Low Energy Adaptive Clustering Hierarchy* (LEACH)

Pada penerapan *Cluster Based Algorithm*, *routing protocol* yang dapat digunakan adalah *Low Energy Adaptive Clustering Hierarchy* (LEACH). Protokol ini dirancang oleh Heinzehnan et al dari Massachusetts Institute of Technology USA dan merupakan protokol klasik dengan efektif berdasarkan *effective energy routing protocol*. LEACH merupakan salah satu contoh dari *hierarchical routing protocol,* dimana mereka mengelompokkan node dalam jaringan dan tidak memerlukan *routing table* dengan informasi yang lengkap.



Gambar 2.2 *Protocol Low-Energy Adaptive Clustering Hierarchy* (LEACH)

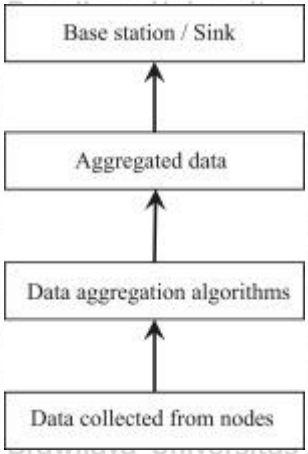
Sumber: (Huo, et al., 2020)

Pada protokol LEACH, dilakukan metode pemilihan *cluster head* terdistribusi, dimana dari beberapa node yang ada akan dipilih secara acak dan node lainnya akan menjadi anggota *cluster.* Lalu info *cluster head* akan dikirimkan dan node lainnya akan memilih *cluster head* untuk bergabung membentuk sebuah *cluster.* Lalu node anggota *cluster* akan mengumpulkan data (*data sensing)* dan mengirimkannya kepada *cluster head,* lalu *cluster head* akan meneruskan ke *sink* dengan metode komunikasi *single-hop. Cluster Head* pada LEACH dapat berkomunikasi langsung dengan *sink* (Huo, et al., 2020).

Pada protokol LEACH terdapat dua tahap yang berjalan. Tahap pertama adalah *setup phase*, yaitu tahap yang melakukan pemilihan *cluster head* dan pembentukan *cluster.* Lalu tahap selanjutnya merupakan *steady-state phase* yang berfokus pada tahap pengumpulan data, agregasi data dan pengiriman data ke *base station (sink). Cluster Head* sendiri memiliki tugas pertama yang merupakan mengumpulkan data-data dari anggota *cluster.* Setelah mengumpulkan data, *cluster head* menggabungkan data dalam upaya untuk menghilangkan redudansi dengan melakukan agregasi data. Lalu tugas kedua *cluster head* adalah mengirimkan data agregasi langsung ke *sink* (Sohraby, et al., 2007).

2.2.5 Agregasi Data

Agregasi data merupakan sebuah prosedur dimana sebuah node akan melakukan pengumpulan data dari beberapa node yang lain. Hal ini dilakukan karena untuk mengurangi transmisi data. Agregasi data juga bertanggung jawab untuk meningkatkan masa kegunaan sebuah jaringan dan juga untuk mengurangi konsumsi energi. Pada agregasi data akan ada redudansi spesifik yang ada dalam data dikumpulkan oleh node, sehingga dengan adanya proses ini maka akan mengurangi informasi yang tidak dibutuhkan (Pourghebleh & Navimipour, 2017). Agregasi data dapat dilihat pada gambar berikut:



Gambar 2.3 Proses Agregasi Data

Sumber: (Pourghebleh & Navimipour, 2017)

Pada proses agregasi data, hal yang pertama kali dilakukan adalah dengan melakukan pengumpulan data pada beberapa node. Setelah itu, akan ditentukan algoritma agregasi data yang akan diterapkan. Lalu data akan diagregasikan dan akan dikirim ke *base station/sink* melalui pemilihan rute yang efektif.

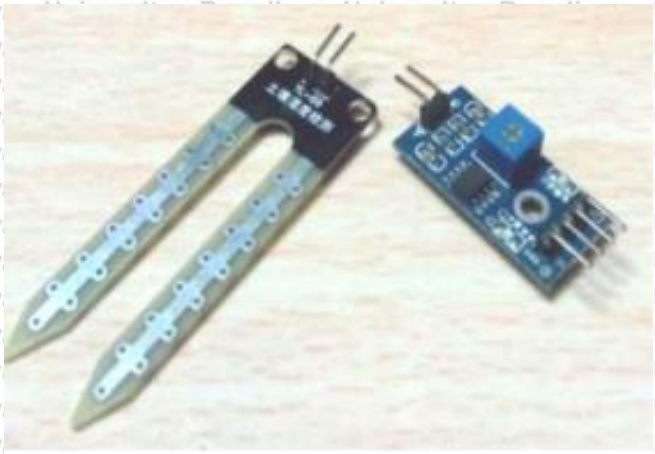
Sedangkan menurut Kaur & Munjal (2020), tujuan data agregasi adalah untuk mengurangi data yang tidak diperlukan dan mengekstrak informasi dari data yang dikumpulkan. Untuk melakukan agregasi data, terdapat beberapa fungsi untuk mengekstrak informasi salah satunya dengan fungsi *Mean.* Fungsi mean dapat digunakan dengan bantuan persamaan:

$$\text{Mean} = \frac{data_1 + data_2 + data_3 + data_4 + \ldots + data_n}{n}$$

Sumber: (Kaur & Munjal, 2020)

2.2.6 *Sensor* Kelembapan Tanah

Salah satu *sensor* kelembapan tanah yang dapat digunakan adalah *Soil Moisture Sensor*. *Sensor* ini merupakan *sensor* yang dapat mendeteksi kelembapan dalam tanah. pada *sensor* ini terdapat dua *pads* yang berfungsi sebagai probe untuk *sensor* yang akan bertindak sebagai resistor variabel. Untuk menggunakan *sensor* ini, peneliti perlu menyambungkan pin VCC dan GND ke perangkat mikrokontroler dan setelah itu hasil SIG akan diterima oleh peneliti. Berikut adalah gambar dari *Soil Moisture Sensor*:



Gambar 2.4 *Soil Moisture Sensor*

Sumber: (Evangeline, et al., 2019)

2.2.7 Arduino UNO

Arduino UNO adalah sebuah board mikrokontroler yang didasarkan pada Atmega328. Arduino UNO mempunyai 14 pin digital input/output (6 diantaranya dapat digunakan sebagai luaran PWM), 6 masukan analog, sebuah osilator 16 MHz, sebuah koneksi USB, sebuah *power jack,* sebuah ICSP *header*, dan sebuah tombol *reset*. Arduino UNO mampu mendukung mikrokontroler, dan dapat dikoneksikan dengan komputer mengunakan kabel USB (Kadir, 2013). Arduino UNO memuat segala hal yang dibutuhkan untuk mendukung sebuah mikrokontroler. Ini karena Arduino UNO merupakan mikrokontroler yang telah dilengkapi dengan library sehingga akan memudahkan pemakaian.

Arduino UNO menggunakan Bahasa pemrograman sederhana dalam pemakaiannya. Bahasa pemrograman yang digunakan oleh Arduino UNO adalah Bahasa pemrograman C. meski, Bahasa pemrograman yang digunakan sederhana, tetapi library yang dipakai dari Bahasa pemrograman ini telah lengkap. Berikut adalah gambar dari Arduino UNO:

Gambar 2.5 Arduino UNO Board

Sumber: (Husdi, 2018)

2.2.8 nRF24L01

nRF24L01 merupakan *single chip radio transceiver* yang menggunakan frekuensi pita gelombang radio sebesar 2.4-2.5 GHz ISM(*Industial, Scientific, and Medical)*. Pada nRF24L01, *transceiver* terdiri dari synthesizer frekuensi terintegrasi penuh (*fully integrated frequency synthesizer*), penguat daya (*power amplifier*), osilator kristal (*crystal oscillator*), demodulator, modulator, dan mesin protokol Enhanced ShockBurst™ (*Enhanced ShockBurst™ protocol engine*). Untuk daya output (*output power)*, saluran frekuensi (*frecuency channels),* dan pengaturan protokol pada nRF24L01 dapat diprogram melalui SPI *Interface*. Konsumsi daya yang digunakan oleh nRF24L01 saat ini cukup rendah yaitu sebesar 9.0 mA dengan daya output sebesar -6dBm dan 12.3mA dalam mode RX. (Nordic Semiconductor, 2006)

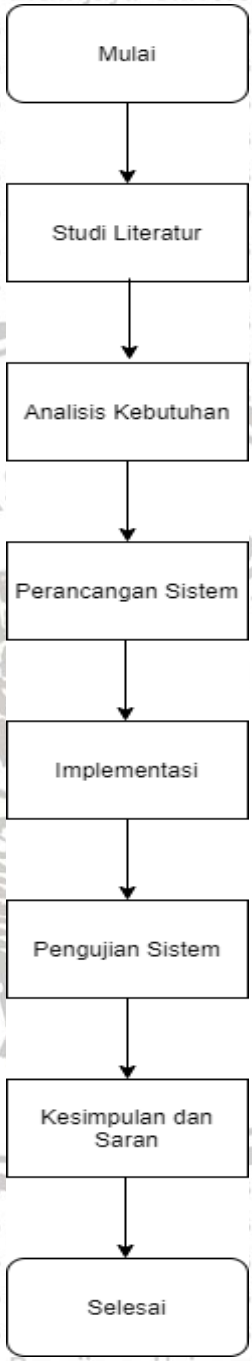
Gambar 2.6 Modul Komunikasi nRF24L01

2.2.9 *Thingspeak*

Thingspeak merupakan salah satu sebuah *opensource Internet of Things* yang dapat digunakan untuk mengumpulkan, memvisualisasikan, dan menganalisis aliran data langsung di *cloud*. Pada *Thingspeak*, data akan disimpan dalam *channel*. Setiap *channel* menyimpan hingga 8 bidang data. Peneliti dapat menggunakan perangkat yang terhubung internet dengan *ThingSpeak*. Saat mengirim data dari sebuah perangkat maka peneliti dapat menggunakan *library* asli untuk *platform prototype* perangkat keras yang digunakan seperti Arduino, ESP-8266, Particle dan *Raspberry Pi*. Peneliti juga dapat mengirim data ke ThingSpeak dari mesin atau *gateway* lokal menggunakan REST API atau MQTT API.

BAB 3 METODOLOGI

3.1 Kerangka Penelitian



Gambar 3. 1 Diagram Alir Penelitian

Penelitian ini menggunakan tipe penelitian implementatif perancangan. Tipe penelitian ini merupakan tipe penelitian yang menghasilkan produk sebagai solusi terhadap permasalahan dalam penelitian. Tipe penelitian ini menggunakan

pendekatan perancangan (design). Tipe penelitian implementatif perancangan merupakan tipe penelitian yang melakukan proses dari analisis perancangan hingga pengujian. Tahapan dari penelitian ini dimulai dari studi literatur hingga kesimpulan dan saran.

Tahapan pertama dari penelitian ini adalah studi literatur. Studi literatur merupakan tahapan dimana peneliti mencari dan menemukan referensi teori yang berhubungan dengan penelitian yang akan dilakukan. Tahap ini diperlukan sebagai pendalaman untuk hal-hal yang berhubungan dengan penelitian. Tahap selanjutnya adalah analisis kebutuhan. Pada tahap ini akan dibahas mengenai apa saja kebutuhan dari sistem yang dibuat. Analisis kebutuhan akan membantu peneliti untuk mengembangkan sistem secara terarah dan sistematis. Setelah analisis kebutuhan, selanjutnya adalah tahap perancangan sistem. Pada tahap ini dilakukan dengan merancang sistem sesuai dengan kebutuhan yang telah dideskripsikan. Hasil dari perancangan sistem ini nantinya akan diimplementasikan pada tahap implementasi. Hasil sistem yang dibuat pada tahap implementasi selanjutnya akan diuji pada tahapan pengujian sistem. Pada tahap pengujian sistem ini dipastikan seluruh kebutuhan telah dipenuhi oleh sistem. Dan yang terakhir adalah kesimpulan dan saran. Dimana pada tahap ini akan mejawab dari rumusan penelitian dan memberikan saran untuk penelitian yang akan dilakukan selanjutnya.

3.1.1 Studi Literatur

Studi literatur dilakukan dengan mencari referensi dari penelitian sebelumnya dan mempelajari teori-teori yang berkaitan dengan penelitian yang akan dilakukan yaitu implementasi *clusterbased algorithm* untuk melakukan agregasi data pada *wireless sensor network*. Sumber teori penelitian ini adalah jurnal dan *proceedings*. Dasar teori tersebut adalah sebagai berikut:

1. *Wireless Sensor Network* (WSN)

Dasar teori mengena *Wireless Sensor Network* (WSN) dibutuhkan untuk mengetahui bagaimana implementasi dari WSN untuk diterapkan sebagai sistem monitoring pada penelitian ini. Studi literatur untuk WSN dibutuhkan untuk mengetahui bagaimana penerapan dan cara kerja WSN yang akan diimplementasikan pada penelitian ini. WSN akan dibuat dari sejumlah node sensor ringan berukuran kecil yang digunakan untuk tujuan *sensing* dan *monitoring*

2. *Cluster Based Algorithm*

Dasar teori mengenai *Cluster Based Algorithm* dibutuhkan untuk mengetahui bagaimana implementasi dari algoritma untuk agregasi data tersebut. Studi literatur untuk *Cluster Based Algorithm* dibutuhkan untuk mengetahui arsitektur dari agregasi data yang dilakukan. Pada *Cluster Based Algorithm* dijelaskan bahwa seluruh jaringan akan dibagi dalam beberapa *cluster.*

3. *Routing Protocol* LEACH *(Low Energy Adaptive Clustering Hierarchy)*

Dasar teori mengenai LEACH dibutuhkan untuk mengetahui bagaimana implementasi dari *routing protocol* tersebut. Studi literatur untuk LEACH dibutuhkan untuk mengetahui komponen apa saja yang ada dalam *routing protocol* tersebut. Komponen yang disebutkan pada LEACH adalah seperti node yang melakukan *data sensing* (node *sensor*), *cluster head,* dan node untuk mengumpulkan data (*sink node)*

4. Agregasi Data

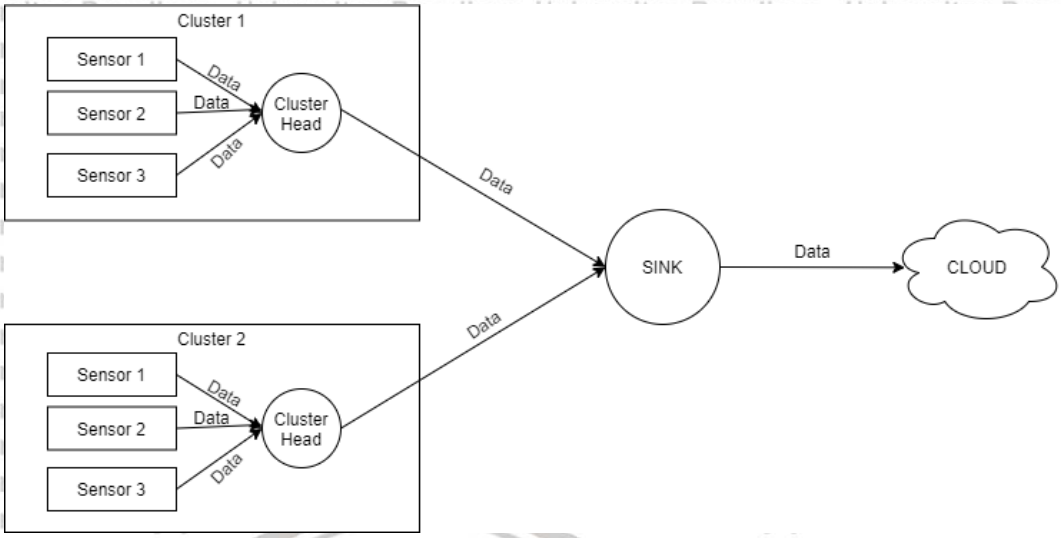
Dasar teori mengenai agregasi data dibutuhkan untuk mengetahui bagaimana agregasi data bekerja dan algoritma apa yang dapat digunakan untuk menjalankan proses agregasi data. Cara kerja dari agregasi data dibutuhkan untuk mengetahui dengan tepat cara kerja dari proses ini dan dapat mengimplementasikannya secara tepat.

3.1.2 Analisis Kebutuhan

Analisis kebutuhan sistem dibuat untuk mengarahkan peneliti pada kebutuhan apa saja yang diperlukan dalam penelitian. Pada analisis kebutuhan sistem akan membahas mengenai kebutuhan fungsional. Kebutuhan fungsional adalah kebutuhan yang berisi layanan apa saja yang harus disediakan oleh sistem. Pada kebutuhan fungsional membantu sistem untuk dapat dikembangkan secara terarah dan sistematis. Dari analisis kebutuhan fungsional juga diharapkan dapat memberi informasi tentang hasil yang akan dihasilkan oleh sistem yang telah dibuat oleh peneliti. Informasi ini dapat berupa berhasil atau tidaknya sistem yang telah dibuat oleh peneliti.

3.1.3 Perancangan Sistem

Pada perancangan sistem akan menjelaskan rancangan sistem yang akan dibuat dan perancangan pengujian. Berikut gambar dari perancangan sistem pada penelitian ini:



Gambar 3. 2 Diagram Blok Sistem

Perancangan dalam penelitian ini dibagi menjadi empat bagian yaitu:

1. Perancangan node sensor

Node sensor akan digunakan untuk menjalankan proses pengumpulan data. Node sensor akan dirancang dengan menyambungkan kepada mikrokontroler dan sensor yang akan digunakan. Node sensor akan diletakkan pada objek dari penelitian yang akan dilakukan.

2. Perancangan Komunikasi Data

Komunikasi data akan digunakan oleh node-node yang ada untuk saling berhubungan. Komunikasi data dirancang dengan menggunakan modul komunikasi nRF24L01 dan modul SIM800L GSM/GPRS. Modul komunikasi ini akan diletakkan pada node-node yang bersangkutan.

3. Perancangan *Clustering*

Clustering merupakan proses membagi jaringan menjadi beberapa *cluster,* dimana satu set *cluster* terdapat node *sensor* dan *cluster head.* Node *sensor* akan melakukan *data sensing* dan hasilnya akan dikirimkan kepada *cluster head.* Pada perancangan *clustering* akan dijelaskan rancangan bagaimana node *sensor* bergabung dengan *cluster head.*

4. Perancangan Agregasi Data

Agregasi data merupakan sebuah proses mengumpulkan data dari node *sensor.* Agregasi data akan dilakukan pada *cluster head.* Pada perancangan agregasi data akan dijelaskan bagaimana agregasi data akan dilakukan pada sistem ini.

5. Perancangan pengujian

Perancangan pengujian akan membahas mengenai proses pengujian yang akan dilakukan saat menguji sistem. Pada perancangan pengujian akan dijelaskan mengenai scenario pengujian fungsional untuk menguji kebutuhan

fungsional dan pengujian untuk *Response Time* terhadap pencarian *Cluster Head* baru, *Energy Efficiency*, dan Jarak Antar Node.

3.1.4 Implementasi

Implementasi akan menjelaskan mengenai proses implementasi *clusterbased algorithm* untuk melakukan agregasi data pada *wireless sensor network*. Implementasi pada pada penelitian ini dibagi kedalam dua bagian yaitu implementasi *clustering*, dan implementasi node *sink.*

1. Implementasi node sensor

Node sensor diimplementasikan dengan menggunakan komponen mikrokontroler, sensor, nRF24L01, dan *power adaptor*. Mikrokontroler akan melakukan komputasi, node sensor akan melakukan *data sensing* pada tanah, nRF24L01 sebagai modul komunikasi dan *power adaptor* digunakan sebagai sumber energi pada node sensor.

2. Implementasi *Clustering*

Clustering diimplementasikan pada node *sensor* kelembapan tanah dan *cluster head*. *Sensor* akan mengumpulkan data dari dalam tanah secara *real-time.* Setelah data dikumpulkan, hasilnya akan dikirimkan pada *cluster head.* Selanjutnya data pada *cluster head* akan dikirimkan kepada node *sink.*

3. Implementasi Agregasi Data

Agregasi data merupakan proses pengumpulan data dari node *sensor. Cluster head* akan melakukan agregasi data setelah data diterima. Data hasil agregasi data akan dikirimkan ke node *sink* yang selanjutnya akan dikirimkan ke *cloud* untuk ditampilkan sebagai hasil dari monitoring kelembapan tanah yang telah dibuat.

3.1.5 Pengujian Sistem

Pengujian dilakukan setelah melakukan implementasi, hal ini bertujuan untuk menguji sistem yang dibuat apakah memenuhi kebetuhan fungsional dan untuk mengetahui kinerja dari sistem. Pengujian sistem meliputi pengujian fungsional dan pengujian kinerja. Pengujian fungsional dilakukan didasari oleh kebutuhan fungsional. Pengujian kinerja dilakukan dengan hasil data *sensor* yang didapatkan secara *real-time*.

3.1.6 Kesimpulan dan Saran

Tahapan pengambilan kesimpulan dan saran berisi kesimpulan dan saran terhadap penelitian yang telah dilakukan. Kesimpulan didapatkan berdasarkan hasil perancangan, implementasi dan pengujian. Saran bertujuan untuk memberikan saran mengenai penelitian yang akan dilakukan selanjutnya.

BAB 4 KERANGKA PENELITIAN

Bab ini menjelaskan mengenai analisis dan perancangan sistem. Analisis kebutuhan sistem membahas mengenai kebutuhan sistem. Pada perancangan sistem peneliti melakukan perancangan sistem yang telah dideskripsikan kebutuhannya pada analisis kebutuhan.

4.1 Analisis Kebutuhan Sistem

Analisis kebutuhan sistem dibuat untuk melakukan pengembangan sistem dapat lebih terarah dan sistematis. Pada analisis kebutuhan sistem akan dibahas mengenai kebutuhan sistem.

4.1.1 Kebutuhan Fungsional Sistem

Pada analisis kebutuhan fungsional sistem akan dibahas mengenai kebutuhan yang harus disediakan oleh sistem. Tabel 4.1 adalah kebutuhan fungsional pada sistem.

Tabel 4.1 Tabel Kebutuhan Fungsional Sistem

No	Kode Fungsi	Kebutuhan Fungsional
1	KF-01-01	Node *sensor* harus dapat mengirimkan informasi id berupa angka kepada *sink* untuk dilakukan pemilihan *cluster head*
2	KF-01-02	Node *sensor* harus dapat melakukan *data sensing* dan mendapatkan data kelembapan tanah
3	KF-01-03	Node *sensor* harus dapat mengirimkan data id node dan kelembapan data kepada *cluster head* yang terdekat untuk membentuk satu *cluster*
4	KF-01-04	Node *sensor* yang terpilih sebagai *cluster head* harus dapat melakukan proses agregasi data setelah menerima data dari *node* sensor lainnya
5	KF-02-01	Node *sink* dapat menentukan *cluster head* dengan memilih dua angka terkecil dari node *sensor*
6	KF-02-02	Node *sink* dapat mencari *cluster head* baru ketika node *sink* tidak menerima data agregasi dari *cluster head* lebih dari 30 detik
7	KF-02-03	Node *sink* harus dapat menerima informasi dari *cluster head* yang mengirimkan data

8	KF-02-04	Node *sink* harus dapat melakukan pengiriman data agregasi yang diterima ke *cloud*
9	KF-03-01	Node *cloud* harus dapat menerima data terbaru dari *sink*
10	KF-03-02	*Cloud* harus dapat menampilkan data terbaru yang diterima dari node *sink*

Dari kebutuhan fungsional yang telah dijabarkan, diharapkan *Cluster Based Algorithm* berhasil diterapkan. Keberhasilan algoritma yang diterapkan akan membuat teknik agregasi data pada WSN dapat dilakukan. Teknik agregasi data yang diterapkan ini diharapkan dapat menghemat energi dari WSN yang telah dibangun.

4.2 Perancangan Sistem

Perancangan sistem merupakan tahapan dimana komponen-komponen yang ada pada kebutuhan sistem dirancang. Perancangan sistem dilakukan sebagai petunjuk agar sistem yang akan dibuat lebih terarah. Perancangan sistem meliputi gambaran umum sistem, perancangan alur sistem, perancangan *clustering,* perancangan node *sink,* dan perancangan pengujian.

4.2.1 Gambaran Umum Sistem

Sistem yang akan dibuat pada penelitian ini adalah implementasi Teknik Agregasi Data Menggunakan *Cluster Based Algorithm* Untuk Pengiriman Data Sensor Pada *Wireless Sensor Network* (WSN*)*. Pada sistem ini akan terdapat node *sensor, cluster head*, dan node *sink.*

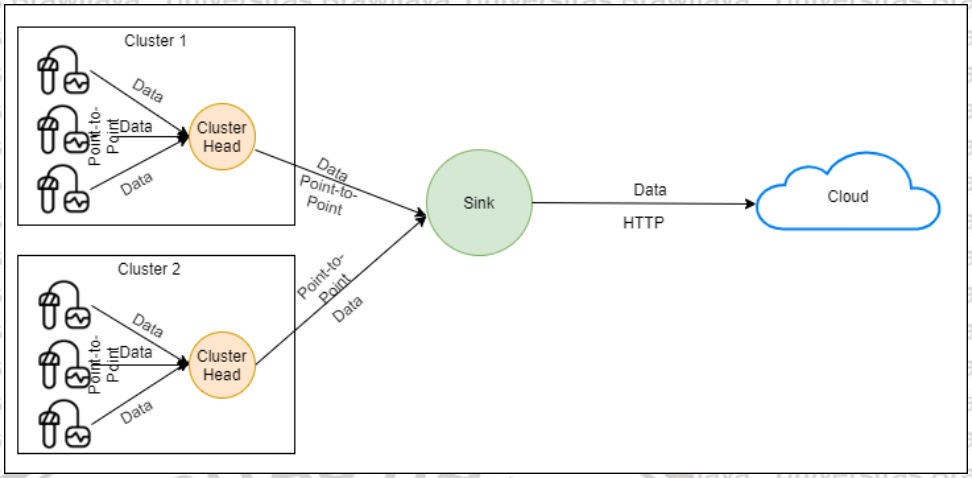
Node *sensor* pada sistem ini merupakan *sensor Soil Moisture Sensor*. Node *sensor* akan mengumpulkan data kelembapan dari tanah secara jangka waktu tertentu. Hasil dari node *sensor* ini akan dikirimkan kepada *cluster head*.

Cluster head pada sistem ini merupakan perangkat yang menerima pengumpulan data dari beberapa node *sensor.* Pada satu *cluster head* hanya akan menerima data dari node *sensor* yang berada pada satu *cluster* yang sama. *Cluster head* akan mengumpulkan data dari node *sensor* sebelum dikirimkan kepada node *sink*.

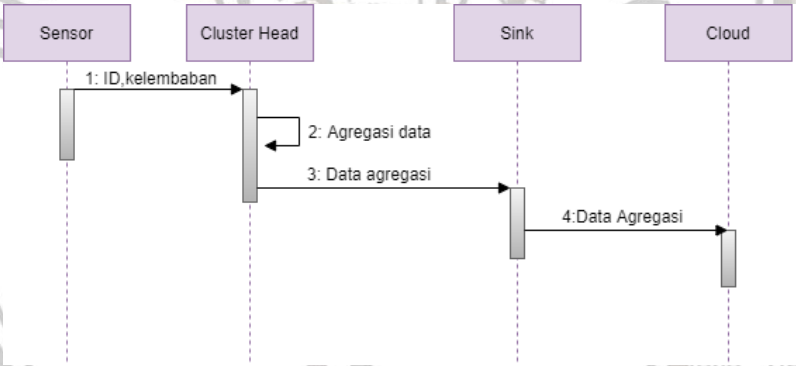
Node *sink* pada sistem ini merupakan perangkat yang akan menerima kumpulan data dari *cluster head* dan melakukan agregasi data. Pada node *sink* juga akan melakukan pemilihan *cluster head* terlebih dahulu sebelum proses agregasi data dilakukan. Node *sink* akan mengirimkan hasil agregasi data ke *cloud. Cloud* akan menerima data dari node *sink* dan akan menampilkan data pada layar.

Pada node *sensor* yang akan mengirimkan data ke *cluster head* akan menggunakan protokol *Point-to-Point* (PPP). Protokol ini merupakan protokol yang dapat digunakan untuk koneksi langsung antar dua node. Protokol ini juga

digunakan pada konesi antar *Cluster Head* dan node *sink.* Sedangkan untuk protokol yang digunakan dari node *sink* ke *cloud* adalah *Hypertext Transfer Protocol* (HTTP).



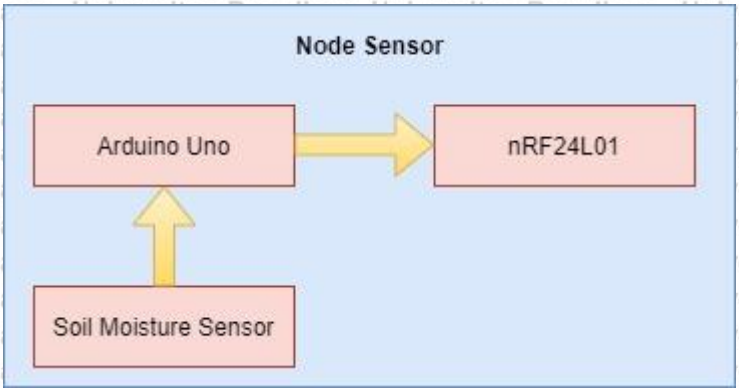
Gambar 4. 1 Rancangan Arsitektur Sistem



Gambar 4. 2 *Sequence Diagram* Sistem

4.2.2 Perancangan Node Sensor

Node sensor pada penelitian ini akan memiliki komponen mikrokontroler, sensor, nRF24L01, dan *power adaptor*. Mikrokontroler yang digunakan adalah Arduino uno dan sensor yang digunakan pada penelitian ini adalah *Soil Moisture Sensor* yang akan digunakan untuk mengukur kelembapan tanah. Pada gambar 4.3 menggambarkan rancangan dari node sensor yang akan digunakan pada penelitian ini.

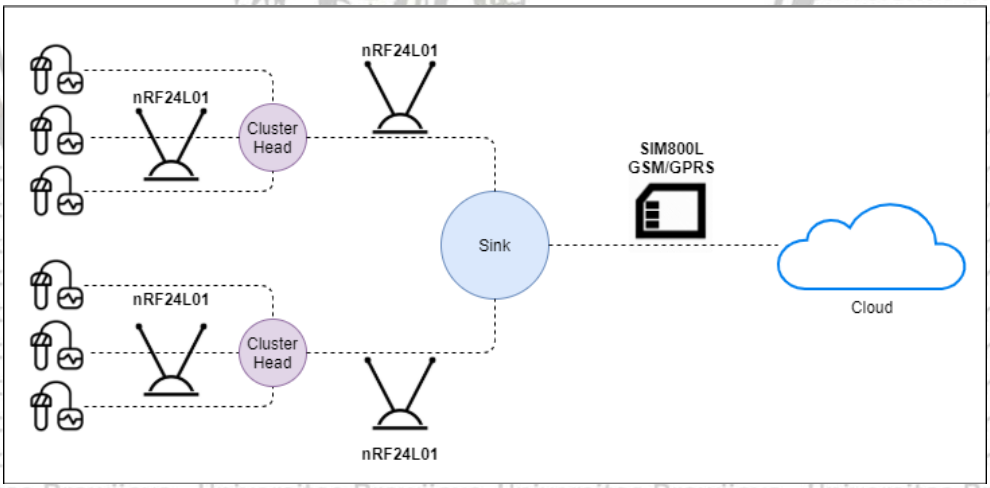


Gambar 4. 3 Perancangan Node *Sensor*

Terdapat dua data yang dimiliki oleh sensor node, yaitu data ID node dan hasil *sensing* yang telah dilakukan. Data ID node akan dikirimkan kepada *base station (sink)* untuk diproses dan dipilih ID yang akan dijadikan *cluster head.* sedangkan data hasil *sensing* akan dikirimkan kepada *cluster head* masing-masing node sensor untuk akan dilakukan proses agregasi data.

4.2.3 Perancangan Komunikasi Data

Perancangan komunikasi data akan dilakukan untuk merancang modul komunikasi apa yang akan digunakan antar node untuk saling berkomunikasi. Komunikasi data pada WSN di penelitian ini menggunakan modul komunikasi nRF24L01 dan modul SIM800L. Pada gambar 4.4 digambarkan rancangan dari komunikasi data yang diterapkan dalam WSN ini.



Gambar 4. 4 Perancangan Komunikasi Data

Pada rancangan komunikasi data dengan modul nRF24L01 digunakan untuk node *sensor* berkomunikasi dengan *cluster head* dan mengirimkan hasil *data sensing.* Modul komunikasi nRF24L01 juga digunakan untuk berkomunikasi antar *cluster head* dengan node *sink* untuk mengirimkan hasil data agregasi. Sedangkan dari node *sink* untuk mengirimkan data agregasi kepada *cloud* maka menggunakan bantuan modul SIM800L GSM/GPRS.

4.2.4 Perancangan Struktur Data

Perancangan struktur data dilakukan untuk merancang struktur data yang akan dikirim antar node. Struktur data pada sistem ini terdapat beberapa tipe. Untuk data dari id node dan data agregasi akan menggunakan data tipe int, lalu untuk data hasil *data sensing* merupakan data analog dan data yang dikirimkan dari *sink* ke *cloud* berupa string.

Tabel 4.2 Tabel Struktur Data pada Sistem

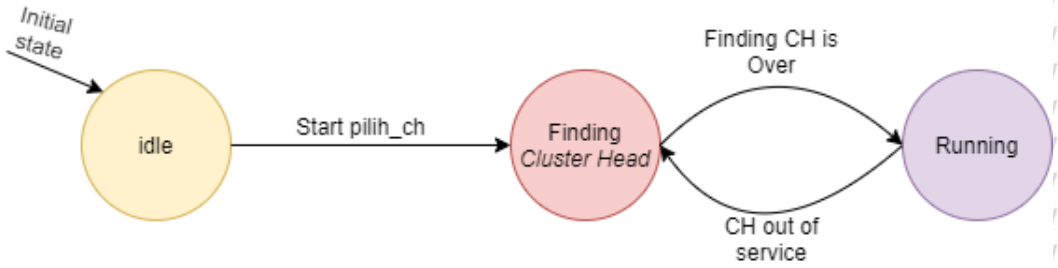
Tipe data	Format Data	Nama variable	value
int	Json	Id	Id dari node *sensor* yang akan dikirimkan ke *sink*
analogData (float)	Json	Kelembaban	Hasil *data sensing* dari kelembapan pada tanah
int	Json	data_agregasi	Berisi nilai data agregasi yang dikirimkan dari *cluster head* ke sink
String	Json	str	Berisi nilai data agregasi data yang akan dikirimkan oleh *sink* ke *cloud*

4.2.5 Perancangan *Clustering*

Clustering diimplementasikan pada node *sensor* kelembapan tanah dan *cluster head*. Node *sensor* yang dipakai merupakan merupakan *sensor Soil Moisture Sensor*. Node *sensor* akan mengumpulkan data dari dalam tanah. Setelah data dikumpulkan, hasilnya akan dikirimkan pada *cluster head.* Selanjutnya data pada *cluster head* akan dikirimkan kepada node *sink.*

Clustering akan dilakukan dengan menggunakan *routing algorithm* LEACH. Pada protokol LEACH terdapat dua tahap yang berjalan. Tahap pertama pada protocol LEACH adalah *setup phase*, yaitu tahap yang melakukan pemilihan *cluster head* dan pembentukan *cluster*. Pada gambar 4.4 menggambarkan proses yang berjalan pada tahap *clustering*. Pada tahap *initial state* node *sensor* akan mengirimkan info id berupa angka kepada node *sink,* yang lalu pada *tahap idle* node *sink* akan menerima data id node. Lalu dari tahap *idle* ke tahap *finding Cluster Head* akan menjalankan method pilih_ch yang akan melakukan pencarian *cluster head.* Pencarian *cluster head* dimulai dengan mengurutkan angka berdasarkan urutan dari angka terkecil. Setelah itu, node *sink* akan mengirimkan informasi dua angka yang terpilih sebagai *cluster head* kepada seluruh node *sensor.* Lalu, ketika *sink* mengirimkan id node *cluster head* terpilih maka node yang terpilih akan mencocokkan id *cluster head* yang diterima dengan id node tersebut apabila sama akan menjadikannya *cluster head* dan menjalankan fungsi sebagai *cluster head*Lalu, sisa node *sensor* yang tidak terpilih akan mencari *cluster head* terdekat

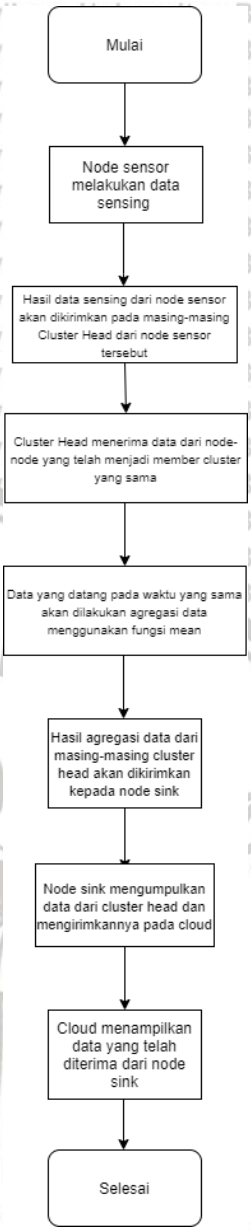
untuk ikut bergabung dan membentuk satu set *cluster.* Lalu, tahap kedua pada LEACH merupakan *steady-state phase* yang berfokus pada tahap pengumpulan data, agregasi data dan pengiriman data ke *base station (sink)*. Pada tahap *Running* seperti pada gambar, node *sensor* akan memulai untuk mengirimkan hasil *data sensing* ke *cluster head* untuk dilakukan agregasi data dan dikirimkan kepada *sink*. Pada *sink* yang berhasil menerima data agregasi akan selanjutnya dikirimkan kepada *cloud,* tetapi apabila *sink* tidak berhasil menerima data agregasi dari salah satu *cluster head* maka *sink* akan menganggap bahwa *cluster head* telah mati yang akan berada pada tahap *CH out of service* dan akan memulai untuk mencari *cluster head* baru.



Gambar 4. 5 *Finite State Machine Clustering*

4.2.6 Perancangan Agregasi data

Agregasi data pada sistem ini merupakan proses yang akan melakukan pengumpulan data yang diterima dari node *sensor.* Agregasi data akan dilakukan pada *cluster head.* Pada penelitian ini, agregasi yang digunakan adalah dengan mencari rata-rata dari nilai data kelembapan tanah yang dikirimkan pada tiap node *sensor.* Pada gambar 4.5 akan menunjukkan perancangan dari agregasi data.



Gambar 4. 6 Diagram Alir untuk Agregasi Data

Pada gambar 4.5 dijelaskan mengenai gambar diagram alir untuk agregasi data. Pada awal diagram dijelaskan bahwa node *sensor* akan melakukan *data sensing* dengan melakukannya pada tanah. Hasil *data sensing* akan dikirimkan dari node *sensor* kepada *cluster head. Cluster Head* sebelumnya telah dipilih pada tahap *Clustering,* dimana ketika *sink* mengirimkan id node *cluster head* terpilih maka node yang terpilih akan mencocokkan id *cluster head* yang diterima dengan id node tersebut apabila sama akan menjadikannya *cluster head* dan menjalankan fungsi sebagai *cluster head.* Selanjutnya, *Cluster head* akan mengumpulkan data dari node *sensor* untuk ditampung hingga node-node telah mengirimkan *data sensing*. Data dari anggota *cluster* yang datang pada waktu yang sama akan dilakukan tahap agregasi menggunakan fungsi *mean.* Fungsi ini dilakukan dengan

menjumlahkan seluruh data yang telah diterima lalu membagi nya dengan jumlah node yang telah mengirimkan *data sensing*. Setelah proses agregasi dilakukan, data hasil agregasi data akan dikirimkan oleh *cluster head* kepada node *sink*. Setelah data diterima, node *sink* akan mengirimkan data agregasi yang diterima dari *cluster head* kepada *cloud* untuk ditampilkan.

4.3 Perancangan Pengujian

Perancangan pengujian dilakukan untuk menguji sistem sesuai dengan kebutuhan yang diperlukan. Pada perancangan pengujian ini terdapat perancangan pengujian fungsional, perancangan pengujian untuk *response time* terhadap pencarian *Cluster Head baru*, perancangan pengujian untuk *energy efficiency,* dan perancangan pengujian untuk jarak antar node.

4.3.1 Perancangan Pengujian Fungsional

Perancangan pengujian fungsional dilakukan untuk memastikan bahwa sistem memenuhi kebutuhan yang dibutuhkan. Kebutuhan yang diuji diambil dari kebutuhan yang telah didefinisikan sebelumnya. Pada table 4.1 sebelumnya merupakan kebutuhan fungsional dari sistem sedangkan tabel 4.2 merupakan tabel skenario pengujian kebutuhan fungsional.

Tabel 4.3 Tabel Skenario Pengujian Fungsional

No	Kode Fungsi	Kebutuhan Fungsional	Skenario Pengujian
1	KF-01-01	Node *sensor* harus dapat mengirimkan data angka kepada *sink*	1. Menyalakan seluruh node yang ada 2. Seluruh node mengirimkan informasi id berupa angka kepada *sink* 3. *Sink* menerima data angka sesuai dengan banyaknya jumlah node yang ada
2	KF-01-02	Node *sensor* harus dapat melakukan *data sensing* dan mendapatkan data kelembapan tanah	1. Menyalakan seluruh node *sensor* 2. Bergabung ke *cluster head* 3. Melakukan *data sensing* dari tanah untuk mendapatkan data kelembapan tanah

3	KF-01-03	Node *sensor* harus dapat mengirimkan data id node dan kelembapan data kepada *cluster head* yang terdekat untuk membentuk satu *cluster.*	1. Sink mengirimkan info node sensor yang terpilih menjadi cluster head kepada sisa node sensor yang tidak terpilih 2. Node sensor yang tidak terpilih akan mencari id cluster head terdekat 3. Setelah mengetahui cluster head yang terdekat, node sensor mengirimkan data tersebut kepada cluster head tersebut
4	KF-01-04	Node *sensor* yang terpilih sebagai *cluster head* harus dapat melakukan proses agregasi data setelah menerima data dari *node* sensor lainnya	1. Node *sensor* mengirimkan data kepada *cluster head* 2. *Cluster head* akan melakukan proses agregasi data kepada data yang sebelumnya telah diterima
5	KF-02-01	Node *sink* dapat menentukan *cluster head* dengan memilih dua angka terkecil dari node *sensor*	1. Menyalakan seluruh node yang ada 2. Seluruh node mengirimkan angka kepada *sink* 3. *Sink* menerima data angka sesuai dengan banyaknya jumlah node yang ada 4. *Sink* mengurutkan angka yang diterima dari terkecil hingga terbesar 5. *Sink* memilih angka terkecil sebagai *cluster head* dan menginfokan kepada node *sensor*
6	KF-02-02	Node *sink* dapat mencari *cluster head* baru ketika node *sink* tidak menerima	1. Pemilihan *cluster head* dilakukan

		data agregasi dari *cluster head* lebih dari 30 detik	2. *Sink* menerima data agregasi dari *cluster head* 3. *Sink* menunggu hingga 30 detik untuk memastikan apakah *cluster head* belum mati, waktu 30 detik ditentukan karena jarak untuk dikirimkannya data ke node *sink* tidak terjadi tabrakan 4. Setelah 30 detik tidak menerima data, *sink* akan melakukan pencarian *cluster head* baru 5. *Cluster head* baru dipilih
7	KF-02-02	Node *sink* harus dapat menerima informasi dari *cluster head* yang mengirimkan data	1. Menyalakan node *Soil Moisture Sensor* sebagai node *sensor* 2. Mengimplementasikan kode program untuk menerima informasi dari *cluster head* 3. Menampilkan data pada *Serial Monitor*
8	KF-02-03	Node *sink* harus dapat melakukan pengiriman data agregasi yang diterima ke *cloud*	1. Node *sink* menghubungkan ke *cloud* 2. Node *sink* melakukan pengiriman data 3. Menampilkan data pada *cloud*
9	KF-03-01	*Cloud* harus dapat menerima data terbaru dari *sink*	1. Node *sink* menghubungkan ke *cloud* 2. *Cloud* menerima data dari node *sink*

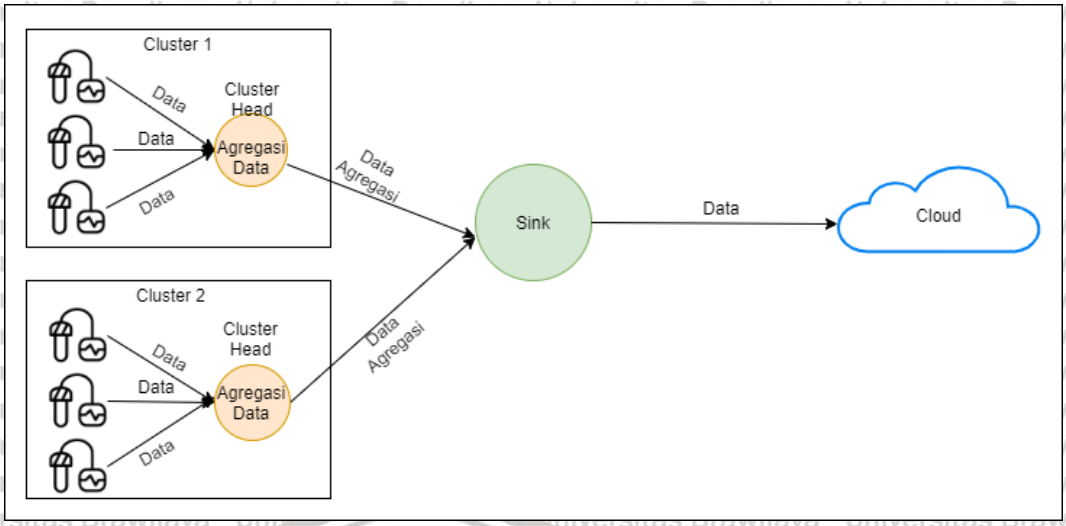
10	KF-03-02	*Cloud* harus dapat menampilkan data terbaru yang diterima dari node *sink*	1. Node *sink* menghubungkan *cloud* 2. *Cloud* menerima data dari node *sink* 3. *Cloud* menampilkan data yang diterima pada layar

4.3.2 Perancangan Pengujian *Response Time* terhadap pencarian *Cluster Head* baru

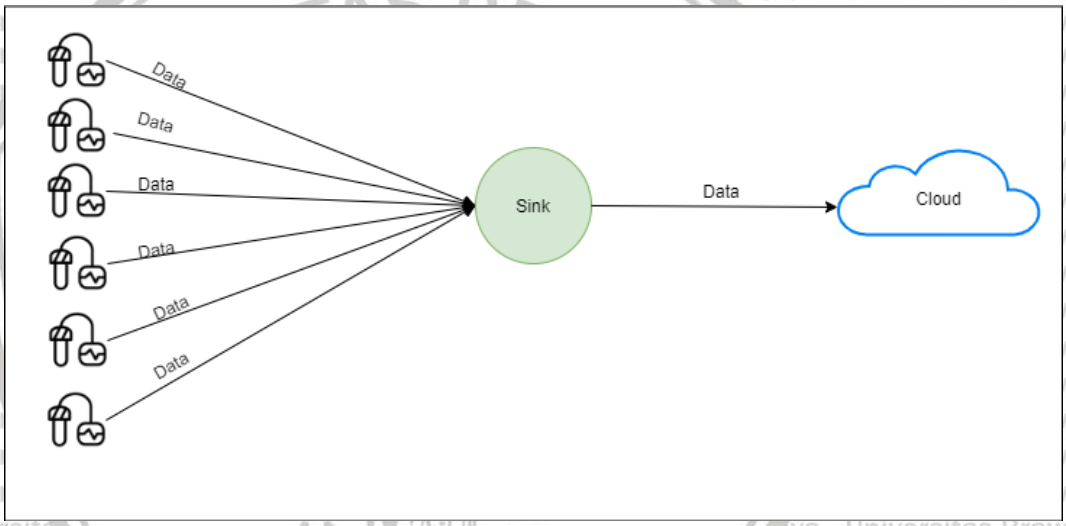
Pengujian waktu pencarian ini dilakukan untuk menguji lamanya waktu yang dibutuhkan untuk mencari sebuah *cluster head* yang baru saat sebuah *cluster head* yang sudah ada mati. Hal ini dilakukan untuk memantau lama waktu yang dibutuhkan untuk mencari *cluster head* baru agar Ketika terdapat *cluster head* yang mati tidak memerlukan waktu yang lama untuk mencari yang baru sehingga tidak menghambat proses dari sistem yang sedang berjalan. Pengujian dilakukan dengan mematikan salah satu *cluster head* lalu melihat proses pencarian *cluster head* baru pada *sink.* Pada *sink* akan dapat dilihat berapa lama waktu yang dibutuhkan untuk menentukan sebuah *cluster head* yang baru. *Respons time* yang diharapkan dari pengujian ini adalah pencarian *cluster head* baru bisa dilakukan dengan waktu yang sedikit.

4.3.3 Perancangan pengujian *Energy Efficiency*

Pengujian *energy efficiency* dilakukan untuk mengetahui apakah agregasi data memakan energi yang lebih sedikit dari pengiriman data biasa. Hal ini untuk membuktikan pernyataan bahwa dengan menggunakan agregasi data akan terjadi penghematan energi yang digunakan oleh WSN. Penggunaan agregasi data dilakukan pada *cluster head* setelah hasil *data sensing* yang dikirimkan oleh node *sensor* telah diterima. Pengujian dilakukan dengan mengirimkan hasil *data sensing* ke *sink* dengan melalui proses agregasi data juga tanpa melalui proses agregasi data. Dari proses pengujian ini akan diketahui bagaimana agregasi data dapat mempengaruhi energi efisiensi dengan memperhatikan *sink* yang menerima data agregasi dengan data biasa dan membandingkan berapa lama kedua *sink* dapat menyala.



Gambar 4. 7 Skema pengujian pengiriman data dengan Agregasi Data



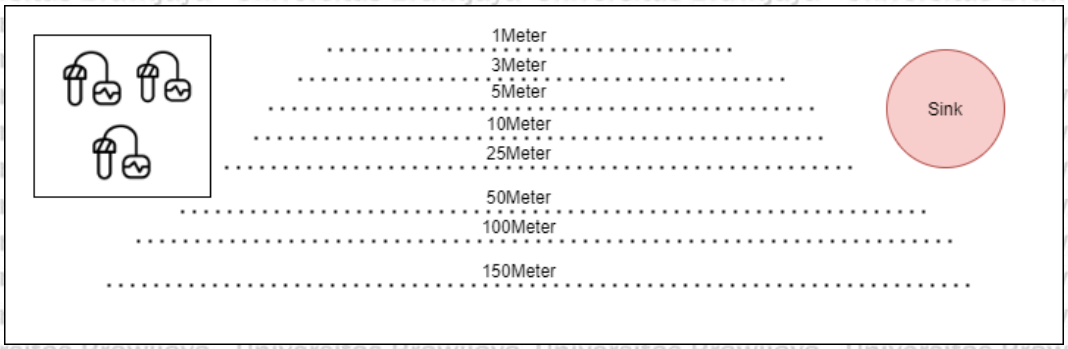
Gambar 4. 8 Skema Pengujian pengiriman data tanpa Agregasi Data

Pengujian yang dilakukan pada dua node *sink* yang berbeda diharapkan akan mendapatkan hasil yang berbeda. Dari pengujian yang dilakukan diharapkan pada node *sink* yang menerima data agregasi dari *cluster head* akan memiliki waktu nyala lebih lama dari yang tidak menerima hasil data agregasi.

4.3.4 Perancangan Pengujian Jarak Antar Node

Pengujian jarak antar node dilakukan untuk mengetahui seberapa batas jauh jarak antar node untuk mengirimkan data dari satu node ke node lain sehingga tidak menyebabkan error maupun *packet loss*. Hal ini dilakukan agar data yang dibutuhkan untuk proses agregasi data tidak mengalami *packet loss* sehingga data dari seluruh node dapat terkumpul dengan baik. Penentuan jarak antar node ini dilakukan dengan mengukur kemungkinan jarak terjauh yang ditempuh dengan memindahkan node-node pada lahan pertanian kebeberapa jarak berbeda pada tiap pengujian. Jarak yang diujikan merupakan jarak yang di uji coba dan

ditentukan sendiri dari peneliti. Setelah penempatan node pada lahan pertanian, maka proses pengiriman data akan dilakukan.



Gambar 4. 9 Skema Pengujian Jarak Antar Node

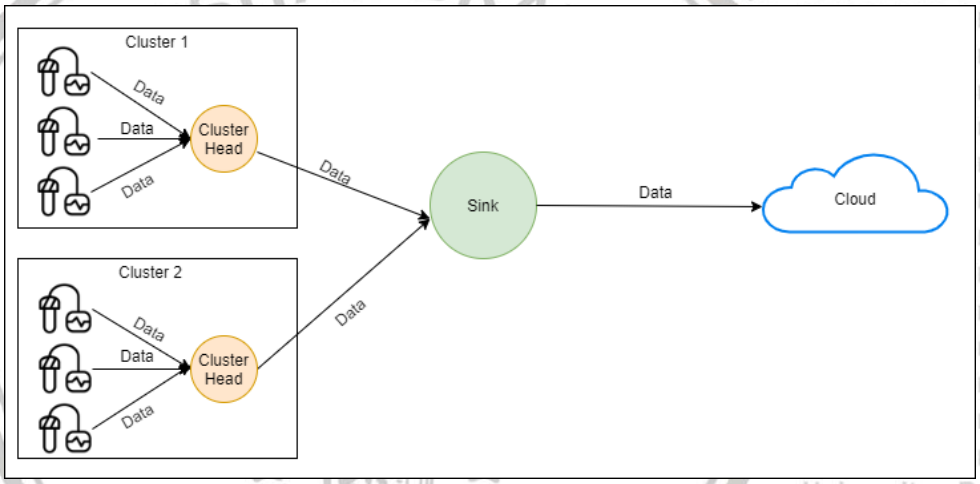
Pada saat proses pengiriman id dari masing-masing node ke *sink* dapat dilihat apakah jumlah dari angka yang didapat dengan jumlah node yang ada di lahan pertanian berjumlah sama atau tidak. Apabila data masih diterima pada node *sink* maka pada jarak tersebut belum menyebabkan *error* maupun *packet loss*, tetapi apabila data tidak diterima oleh node *sink* maka dapat diketahui apabila terdapat node yang tidak berada dalam jangkauan jarak dari *sink.*

BAB 5 IMPLEMENTASI

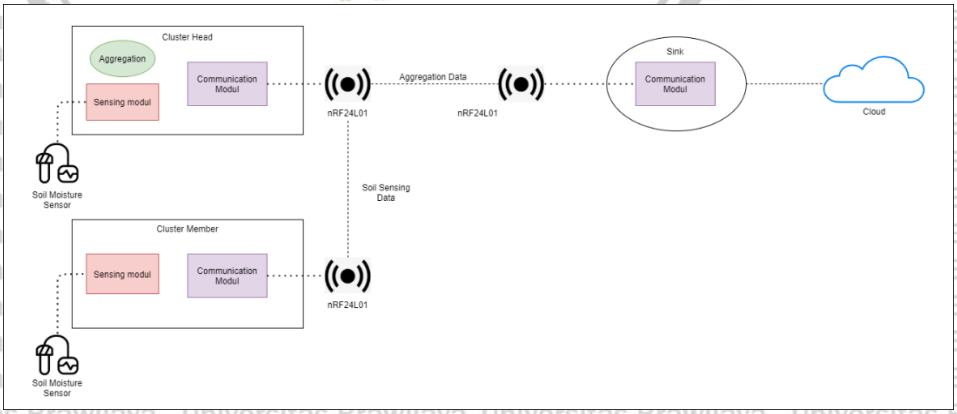
Implementasi merupakan tahapan dari penerapan sistem yang telah dirancang. Dalam penelitian ini, sistem *clustering* diimplementasikan dengan menggunakan algoritma *routing protocol* LEACH. Perangkat yang terlibat dalam penerapan sistem ini adalah node *sensor* sebagai node yang melakukan *data sensing* dan node *sink* sebagai perangkat yang melakukan pengumpulan data dan melakukan proses agregasi data untuk selanjutnya node *sink* akan melakukan pengiriman data ke *cloud*.

5.1 Lingkungan Sistem *Prototype*

Pada tahap ini dijelaskan perangkat apa saja yang digunakan dalam sistem yang dibangun. Sistem yang berupa *prototype* ini akan memiliki komponen perangkat keras dan juga perangkat lunak. Pada gambar 5.1 ditunjukkan rancangan arsitektur sistem *prototype* dan pada gambar 5.2 skema jaringan yang digunakan.



Gambar 5. 1 Arsitektur Sistem



Gambar 5. 2 Skema Jaringan pada sistem

5.1.1 Implementasi Perangkat Keras

Perangkat keras yang akan digunakan dalam implementasi sistem ini didasarkan pada kebutuhan yang dibutuhkan pada sistem, dimana sistem membutuhkan sebuah laptop untuk membangun sistem serta mikrokontroler, sensor, dan modul komunikasi sebagai satu rangkaian node *sensor.* Perangkat keras yang akan digunakan dalam Implementasi Teknik Agregasi Data Menggunakan *Cluster Based Algorithm* untuk Pengiriman Data Sensor pada *Wireless Sensor Network* (WSN) adalah sebagai berikut:

1. Laptop Lenovo ideapad S340

Penggunaan perangkat keras berupa laptop ini digunakan sebagai perangkat untuk pembangunan dari sistem. Berikut adalah spesifikasi laptop yang digunakan pada penelitian ini.

Tabel 5. 1 Tabel Spesifikasi Laptop

Operating System	Windows 10 Home Single Language 64-bit
Processor	AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx (8CPUs), ~2.1 GHz
Memory	8GB RAM
Storage	Up to 512GB M.2 PCIe SSD
GPU	Radeon Vega Graphics AMD 8

2. Arduino UNO

Pada WSN ini dibutuhkan sebuah mikrokontroler yang digunakan sebagai perangkat yang digunakan untuk menerima dan menjalankan program yang diunggah pada mikrokontroler. Berikut adalah spesifikasi dari mikrokontroler berupa Arduino UNO yang digunakan dalam penelitian ini.

Tabel 5. 2 Tabel Spesifikasi Arduino UNO

Operating Voltage	5V
Input Voltage	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20mA
DC Current for 3.3V Pin	50mA

Flash memory	32Kb (0.5 Kb *used by boatloader)*
SRAM	2Kb
EEPROM	1Kb

3. *Soil Moisture Sensor*

Pada WSN ini dibutuhkan sebuah node *sensor* yang digunakan sebagai perangkat untuk melakukan *data sensing*. Berikut adalah spesifikasi dari node *sensor* berupa *Soil Moisture Sensor* yang digunakan dalam penelitian ini.

Tabel 5. 3 Tabel Spesifikasi *Soil Moisture Sensor*

Input Voltage	3.3V/5V
Output Voltage	0-4.2V
DC	35mA
Value Range ADC	1024bit

4. nRF24L01

Pada WSN ini dibutuhkan sebuah modul komunikasi yang digunakan sebagai perangkat untuk melakukan komunikasi dan pengiriman data antar node. Berikut adalah spesifikasi dari modul komunikasi berupa nRF24L01 yang digunakan dalam penelitian ini.

Tabel 5. 4 Tabel Spesifikasi nRF24L01

Working Voltage	DC 1.9V-3.6V
Modulation	GFSK
Max transmission rate	2Mbps
Moment of maximum operating current	<15mA
Operating Frequency	2400-2524GHZ
Interface	SPI

5.1.2 Implementasi Perangkat Lunak

Perangkat lunak yang digunakan dalam implementasi didasarkan pada kebutuhan untuk melakukan pemrograman dan *operating system* yang dapat menerapkan kerangka kerja pemrograman. *Operating system* akan digunakan sebagai perangkat lunak yang menjalankan proses *install* dan mengoperasikan kerangka kerja pemrograman juga *cloud.* Lalu, kerangka kerja pemrograman yang digunakan akan berfungsi sebagai tempat untuk melakukan pemrograman sistem

yang dibuat. Dan *cloud* akan digunakan sebagai perangkat lunak untuk menampilkan data agregasi. Implementasi Teknik Agregasi Data Menggunakan *Cluster Based Algorithm* untuk Pengiriman Data Sensor pada *Wireless Sensor Network* (WSN) adalah terlihat pada tabel 5.5.

Tabel 5. 5 Tabel Spesifikasi Perangkat Lunak

Perangkat Lunak	Keterangan
Windows 10 Home Single Language 64-bit	*Operating System*
Arduino IDE	Kerangka Kerja Pemrograman
Thingspeak	*Cloud*

5.2 Implementasi *Clustering*

Clustering merupakan tahap awal pada sistem ini. *Clustering* dilakukan untuk menentukan *Cluster Head* dan node *sensor* lainnya yang akan bergabung untuk membentuk satu set *Cluster. Clustering* dimulai dengan node *sensor* yang mengirimkan data id node kepada node *sink,* untuk selanjutnya oleh node *sink* akan dipilih yang terkecil untuk selanjutnya ditentukan sebagai *cluster head*. info *cluster head* yang terpilih akan dikirimkan kembali kepada node *sensor* untuk selanjutnya node *sensor* yang tidak terpilih sebagai *cluster head* akan bergabung dengan *cluster head* terdekat sebagai anggota *cluster*nya. Pengiriman antar node akan memanfaatkan modul komunikasi nRF24L01.

5.2.1 Implementasi pada Node *Sensor* untuk Mengirimkan ID Node

Data id node akan dikirimkan dari node *sensor* ke node *sink* menggunakan modul komunikasi nRF24L01. Untuk mengirimkan data id node dari node *sensor* dilakukan pengiriman id node tersebut terlebih dahulu yang terdapat pada method pilih_ch() seperti pada tabel 5.1. Pada terdapat baris kode program yang melakukan perulangan untuk melakukan array data[0] dengan id dari node lalu mengirimkan id node tersebut kepada *sink.*

Tabel 5. 6 Implementasi mengirimkan id node *sensor*

Method pilih_ch()pada sensor

```
1  void pilih_ch() {
2    pm = millis();
3
4    while (1) {
5      if (millis() - pm_kirim > waktu_kirim) {
6        pm_kirim = millis();
7        data[0] = id;
8        nrf_send();
9        data[0] = 0;
```

```
10  }
…
```

Pada baris ke-8 pada method pilih_ch() terdapat pemanggilan method nrf_send().Pada tabel 5.2 yang merupakan method nrf_send() terdapat baris kode untuk melakukan proses pengiriman data yang ada pada node sensor ke node *sink*. Pada baris ke-2, memanggil method stopListening() yang berfungsi untuk berhenti menerima informasi pesan yang akan diterima. Sedangkan pada baris ke-6, method startListening() berfungsi untuk mengetahui *pipes* yang terbuka untuk menerima informasi pesan.

Tabel 5. 7 Implementasi method send() untuk mengirimkan id dari node *sensor*

```
Method nrf_send()pada sensor
1  void nrf_send() {
2    radio.stopListening();
3    if (radio.write( &data, sizeof(data) )) {
4      Serial.println("success");
5    }
6  radio.startListening();
7  }
```

Pada baris 3-5 berfungsi untuk mengirimkan data id kepada node *sink,* setelah dikirim maka akan menampilkan pesan "*success".* Method radio.write() pada baris 3 berfungsi untuk menuliskan data kepada *pipes* yang sedang terbuka.

5.2.2 Implementasi pemilihan *cluster head* pada node *sink*

Implementasi pemilihan *cluster head* pada sink terlebih dahulu dilakukan dengan menjalankan method pilih_ch() pada method void setup(). Pada tabel 5.3 baris 2 method void setup(), merupakan kode untuk memulai komunikasi dengan SIM800L pada 4800bps. Lalu pada baris 3-4 merupakan kode program untuk menggunakan komunikasi dengan serial pada 115200bps lalu akan menampilkan info "sink" pada serial monitor. Pada baris 5 merupakan kode program yang melakukan inisialisasi pada objek radio. Lalu, pada baris 6 terdapat fungsi setPALevel(RF24_PA_LOW) berfungsi untuk mengatur PA *Level low* untuk mencegah masalah pada *power supply.* Untuk baris 7-8 dilakukan proses membuka *pipes* untuk *writes* dan *read* di setiap radio dan baris 9 untuk mengetahui *pipes* yang terbuka untuk menerima informasi pesan.

Tabel 5. 8 Implementasi method setup() pada node *sink*

```
Method void setup()
1  void setup() {
2    SIM800L.begin(4800);
3    Serial.begin(115200);
```

```
4    Serial.println("sink");
5    radio.begin();
6    radio.setPALevel(RF24_PA_LOW);
7    radio.openWritingPipe(addresses[0]);
8    radio.openReadingPipe(0, addresses[0]);
9    radio.startListening();
10   pilih_ch();
11 }
```

Pada baris 10 dalam method void setup() merupakan kode program untuk memanggil method pilih_ch(). Method pilih_ch() merupakan method yang akan melakukan pemilihan *cluster head* pada sistem.

Tabel 5. 9 Implementasi memilih *Cluster Head* pada node *sink*

```
Method pilih_ch()pada sink
1   int id;
2   int kelembaban;
3   int data[3];
4   int ch_1 = 10, ch_2 = 10;
5
6   void pilih_ch() {
7     ch_1=10;
8     ch_2=10;
9     pm = millis();
10    while (1) {
11      nrf_receive();
12      if (data[0] > 0) {
13        if (data[0] < ch_1) {
14          ch_1 = data[0];
15        }
16        if (data[0] > ch_1 && data[0] < ch_2) {
17          ch_2 = data[0];
18          Serial.println();
19          Serial.println();
20        }
21      }
22      if (millis() - pm > waktu_cari) {
23        break;
24      }
25    }
26    delay(1000);
27    data[0] = ch_1;
28    nrf_send();
29    Serial.print("Cluster Head 1 : ");
30    Serial.println(ch_1);
31    delay(1000);
32    data[0] = ch_2;
33    nrf_send();
34    Serial.print("Cluster Head 2 :");
35    Serial.println(ch_2);
```

```
36
37    while (1) {
38      if (millis() - m_kirim > waktu_kirim) {
39        if (b == true) {
40          data[0] = ch_1;
41          b = false;
42        }
43        else {
44          data[0] = ch_1;
45          b  = true;
46        }
47        nrf_send();
48        count_kirim++;
49        if (count_kirim > jumlah_kirim) {
50          break;
51        }
52        pm_kirim = millis();
53      }
54    }
55    Serial.println("selesai");
      delay(3000);
  }
```

Pada tabel 5.4 baris 10-23 pada method pilih_ch() merupakan kode program perulangan yang dilakukan untuk mencari id node terkecil yang akan dijadikan sebagai *cluster head.* Pada baris 9 terdapat pemanggilan method nrf_receive(), yang dimana method ini akan melakukan penerimaan data dari seluruh node yang telah mengirimkan id nya kepada *sink.* Pada baris 12-15, dilakukan pemilihan id untuk *cluster head* pertama dengan memilih id terkecil pertama lalu dilanjutkan pada baris 16-19 dengan pemilihan id terkecil kedua untuk *cluster head* kedua.

Pada baris 26-35, dilakukan penentuan info id *cluster head* pertama dan kedua yang terpilih untuk selanjutnya dikirimkan kepada seluruh node dan ditampilkan pada serial monitor. Pengiriman data keluar dari *sink* kepada seluruh node dilakukan dengan memanggil method nrf_send() seperti pada baris 28 dan 33. Pada baris 37-52 merupakan perulangan yang mengatur waktu pengiriman informasi *cluster head* 1 dan *cluster head* 2 kepada node lainnya. Pada baris 55 merupakan kode program untuk menampilkan "selesai".

Tabel 5. 10 Implementasi mengirimkan id node *Cluster head* terpilih

Method nrf_send()pada sink	
1	`void nrf_send() {`
2	`  radio.stopListening();`
3	`  if (!radio.write( &data, sizeof(data) )) {`
4	`    Serial.println(F("failed"));`
5	`  }`
6	`  else {`

```
 7       Serial.println("terkirim");
 8     }
 9     radio.startListening();
10   }
```

Untuk mengirimkan data id node dari *sink* digunakan fungsi radio.write(). Pada table 5.5 menunjukkan baris kode untuk melakukan proses pengiriman data yang ada pada *sink* ke node *sensor.* Pada baris ke-2, memanggil method stopListening() yang berfungsi untuk berhenti menerima informasi pesan yang akan diterima. Sedangkan pada baris ke-9, method startListening() berfungsi untuk mengetahui *pipes* yang terbuka untuk menerima informasi pesan.

Tabel 5. 11 Implementasi penerimaan data pada node *sink*

Method nrf_receive()pada sink	
1	void nrf_receive() {
2	if (radio.available()) {
3	while (radio.available()) {
4	radio.read(&data, sizeof(data));
5	}
6	radio.startListening();
7	}
8	}

Untuk menerima data pada *sink*, dilakukan kode program seperti table 5.6 pada baris 2 terdapat seleksi kondisi untuk mengecek variable *timestamp* yang diterima. Lalu pada baris 3-4, dilakukan perulangan dimana saat data telah siap maka data akan diterima dengan menggunakan fungsi radio.read(). Sedangkan pada baris ke-6, method startListening() berfungsi untuk mengetahui *pipes* yang terbuka untuk menerima informasi pesan.

Data id *cluster head* yang terpilih akan diterima pada node sensor. Pada node sensor, akan dilakukan penerimaan data id *cluster head* terpilih dari *sink*. Proses ini dilakukan dengan memanggil method pilih_ch() pada node sensor yang akan dipanggil pada method void setup(). Pada baris 2-4 merupakan kode program untuk menggunakan komunikasi dengan serial pada 115200bps lalu akan menampilkan info "id node=" dan info dari id node yang berjalan pada serial monitor. Pada baris 5 merupakan kode program yang melakukan inisialisasi pada objek radio. Lalu, pada baris 6 terdapat fungsi setPALevel(RF24_PA_LOW) berfungsi untuk mengatur PA *Level low* untuk mencegah masalah pada *power supply.* Untuk baris 7-8 dilakukan proses membuka *pipes* untuk *writes* dan *read* di setiap radio dan baris 9 untuk mengetahui *pipes* yang terbuka untuk menerima informasi pesan.

Tabel 5. 12 Implementasi method setup() pada node *sensor*

Method void setup()	
1	void setup() {

```
2    Serial.begin(115200);
3    Serial.print("id node=");
4    Serial.println(id);
5    radio.begin();
6    radio.setPALevel(RF24_PA_LOW);
7    radio.openWritingPipe(addresses[0]);
8    radio.openReadingPipe(0, addresses[0]);
9    radio.startListening();
10   pilih_ch();
11  }
```

Pada baris 10 dalam method void setup() merupakan kode program untuk memanggil method pilih_ch(). Method pilih_ch() merupakan method yang akan mengecek apakah id node dan id *cluster head* sama sehingga sebuah node akan menjadi *cluster head*.

Tabel 5. 13 Implementas iuntuk mengecek sebuah id *Cluster Head*

```
Method pilih_ch()pada node sensor
1   int id = 1;
2   int data[3];
3   boolean ch;
4   unsigned long pm, waktu_cari = 20000;
5   unsigned long pm_kirim, waktu_kirim = 2000;
6   void pilih_ch() {
7    ....
8
9       nrf_receive();
10
11      if (data[0] == id) {
12        ch = true;
13        Serial.println("cluster head");
14        break;
15      }
16      else {
17        ch = false;
18      }
19      if (millis() - pm > waktu_cari) {
20        Serial.println("break");
21        break;
22      }
23    }
24
25   if (ch == true) {
26     Serial.println("cluster head..");
27   }
28    else {
29      Serial.println("node");
30    }
31
32    Serial.println("selesai");
```

```
33 }
```

Pada tabel 5.8 baris ke-9 dilakukan pemanggilan method nrf_receive() untuk menerima data id *cluster head* yang telah dikirimkan oleh *sink*. Lalu, pada baris 11-23 terdapat seleksi kondisi dimana seleksi kondisi digunakan untuk menetapkan apakah id node sama dengan id *cluster head* yang telah dikirimkan oleh *sink*. Pada seleksi kondisi apabila data data[0] sama dengan nilai id maka nilai bolean dari variable Boolean ch akan menjadi *true*, dimana apabila nilai ch adalah *true* maka node akan menjadi *cluster head.* Sedangkan, apabila kondisi tidak terpenuhi maka variable Boolean ch akan bernilai *false*. Lalu, pada baris 25-32 akan melakukan proses mencetak informasi pada serial monitor. Terdapat seleksi kondisi dengan kondisi apabila nilai variable Boolean ch adalah *true* maka akan mencetak informasi "Cluster Head" dan apabila kondisi ini tidak terpenuhi maka akan mencetak informasi "Node" pada serial monitor.

5.2.3 Implementasi pengiriman data dari node *sensor* ke *cluster head*

Setelah *cluster head* terpilih, maka node yang tidak terpilih akan mengirimkan data id dan hasil *sensing* kepada *cluster head* yang terdekat dengan node tersebut. Pada node, *data sensing* akan dilakukan pada method cek_sensor() seperti pada tabel 5.9 yang akan dipanggil pada method void loop(). Pada method cek_sensor() dilakukan pembacaan data pada pin analog yang telah ditentukan untuk mendapatkan hasil dari kelembapan tanah.

Tabel 5. 14 Implementasi *Data Sensing*

Method cek_sensor()	
1	void cek_sensor() {
2	kelembaban = map(analogRead(pin_sensor), 0, 1023,
3	100, 0);
4	}

Pada method void loop() seperti tabel 5.10, dilakukan pemanggilan method cek_sensor() untuk mendapatkan hasil kelembapan tanah. Pada baris 5-9 dilakukan seleksi kondisi dengan kondisi apabila nilai millis() dikurangi m_kirim lebih besar dari waktu_kirim maka nilai dari id node dan kelembapan akan diisikan kepada data[0] dan data [1]. Setelah data diisikan kepada array data[], selanjutnya akan dikirimkan ke *cluster head* dengan memanggil method nrf_send().

Tabel 5. 15 Implementasi mendapatkan data id dan kelembapan dari sebuah node dan mengirimkannya ke *Cluster Head*

Method loop()

```
1  void loop() {
2    cek_sensor();
3    ...
4
5    if (millis() - pm_kirim > waktu_kirim) {
6      data[0] = id;
7      data[1] = kelembaban;
8      pm_kirim = millis();
9      nrf_send();
```

5.2.4 Implementasi penerimaan data pada *cluster head*

Pada *cluster head* akan dilakukan *data sensing* terlebih dahulu pada dirinya dengan memanggil method cek_sensor() pada method void loop(). Pada method cek_sensor() dilakukan pembacaan data pada pin analog yang telah ditentukan untuk mendapatkan hasil dari kelembapan tanah.

Tabel 5. 16 Implementasi *Data Sensing* pada *Cluster Head*

Method cek_sensor()	
1	void cek_sensor() {
2	kelembaban = map(analogRead(pin_sensor), 0, 1023,
3	100, 0);
4	}

Pada tabel 5.11 yang merupakan method void loop(), dilakukan pemanggilan method cek_sensor() untuk mendapatkan hasil kelembapan tanah. Pada baris 3-4 dilakukan seleksi kondisi dengan kondisi apabila nilai variable Boolean ch *true*, maka akan memanggil method nrf_receive() untuk menerima data id dan hasil *data sensing* yang telah dikirimkan oleh node lainnya. Lalu, pada baris 6-9 dilakukan seleksi kondisi dengan kondisi apabila nilai millis() dikurangi m lebih besar dari waktu_cari maka nilai dari id node dan kelembapan yang terima akan diisikan kepada data[0] dan data [1]. Setelah keluar dari seleksi kondisi, pada baris 12-13 nilai data[0] akan diisikan pada variable id_receive dan nilai dari id[1] akan diisikan pada variable kelembapan_receive.

Tabel 5. 17 Implementasi mendapatkan id dan kelembapan tanah dari node lain pada *Cluster Haed*

Method loop()

```
1  void loop() {
2    cek_sensor();
3    if (ch == true) {
4      nrf_receive();
5
6      if (millis() - m > waktu_cari) {
7        m = millis();
8        data[0] = id;
9        data[1] = kelembaban;
10     }
11
12     id_receive = data[0];
13     kelembaban_receive = data[1];
```

5.2.5 Implementasi pemilihan *Cluster Head* baru

Implementasi ini dilakukan apabila terdapat *Cluster Head* terdahulu yang mati maka program akan mulai mencari node yang masih menyala untuk memilih *Cluster Head* baru. Proses ini dilakukan dengan jika *sink* tidak menerima data dari *cluster head* maka *sink* akan memulai penentuan *cluster head* baru dalam waktu 30 detik. Proses ini terlebih dahulu dimulai pada method cek_data() pada *sink* seperti pada tabel 5.13 baris ke 2-4 dimana terdapat seleksi kondisi dengan syarat apabila nilai dari millis()- m_ch1 >cari_baru atau millis()- m_ch2 >cari_baru maka akan menampilkan "Mencari Cluster Head Baru", baris ini untuk mengecek apabila dalam waktu 30 detik tidak ada data yang diterima oleh *sink* dari *cluster head* maka akan menjalankan program pada baris 6-17. Pada baris 6-17, apabila tidak ada data yang diterima maka nilai dari data [0] dan data [1] akan diisi dengan nilai 100 yang nanti akan dikirimkan ke node sensor untuk menandakan bahwa *sink* akan memulai pencarian *cluster head* yang baru. Lalu setelah mengirimkan data maka nilai data [0] dan data [1] akan diisi kembali dengan 0 dan akan menjalankan kembali method pilih_ch().

Tabel 5. 18 Implementasi untuk menentukan *Cluster Head baru*

```
Method cek_data()
1  void cek_data() {
2    if (millis() - pm_ch1 > cari_baru || millis() -
3  pm_ch2 > cari_baru) {
4    Serial.println("Mencari Cluster Head Baru");
5
6      data[0] = 100;
7      data[1] = 100;
8      nrf_send();
9      nrf_send();
10     nrf_send();
11     delay(1000);
12     data[0] = 0;
13     data[1] = 0;
14     pilih_ch();
```

```
15
16      pm_ch1 = millis();
17      pm_ch2 = millis();}
18  ...
19
```

Pada method loop() node sensor pada tabel 5.14 terdapat program yang apabila menerima data [0] dan data [1] bernilai 100 maka data [0] dan data [1] akan diganti dengan 0 dan akan menjalankan method pilih_ch() pada node sensor.

Tabel 5. 19 Implementasi untuk mengetahui *Cluster Head* mati pada node *Sesnor*

Method loop()

```
1  void loop() {
2    ...
3   else {
4      if (data[0] == 100 && data[1] == 100) {
5        data[0] = 0;
6        data[1] = 0;
7        pilih_ch();
8      }
```

5.3 Implementasi agregasi data

5.3.1 Proses agregasi data pada *cluster head*

Proses agregasi data akan dilakukan apabila *clustering* telah dilakukan dan pada *cluster head* telah menerima hasil data *sensing* dari node yang telah bergabung disekitarnya. Hasil data yang diterima selanjutnya akan digunakan untuk proses agregasi data yang dilakukan pada *cluster head.* setelah hasil agregasi data didapatkan, hasil ini akan selanjutnya dikirimkan pada *sink.*

Proses agregasi ini akan dilakukan pada method void loop() pada *cluster head* seperti pada tabel 5.15. Pada baris 17-22 terdapat seleksi kondisi dengan syarat apabila nilai variable kelembaban_receive lebih besar sama dengan 0 dan lebih kecil sama dengan 100 maka nilai kelembaban_receive akan ditambahkan ke dalam array kelompok_kelembaban [id_receive]. Sedangkan nilai dari variable data_agregasi dan jumlah_anggota bernilai 0. Lalu, pada baris 23-26, terdapat perulangan untuk menampilkan nilai id node dan data yang diterima kepada serial monitor.

Pada baris 28-35 terdapat seleksi kondisi dengan syarat apabila nilai kelompok_kelembaban [x] lebih besar dari 0 maka nilai dari variable jumlah anggota akan bertambah dan nilai dari data_agregasi akan ditambahkan dengan nilai kelompok_kelembaban[x]. Lalu setelah proses tersebut nilai dari variable data_agregasi akan dibagi dengan jumlah_anggota. Pada baris 37- 45 merupakan seleksi kondisi dengan kondisi apabila nilai millis() dikurangi m_kirim lebih besar dari waktu_kirim maka dalam seleksi kondisi ini akan menampilkan hasil data agregasi pada variable data_agregasi dan selanjutnya akan dikirimkan pada *sink* dengan memanggil method nrf_send().

Tabel 5. 20 Implementasi Agregasi Data pada *Cluster Head*

```
Method void loop()
1   int id_receive;
2   int kelembaban;
3   int kelembaban_receive;
4   int data[3];
5   unsigned long pm, waktu_cari = 20000;
6   unsigned long pm_kirim, waktu_kirim = 2000;
7
8   int kelompok_kelembaban[10];
9   int data_agregasi;
10  int jumlah_anggota;
11  void loop(){
12   ...
13  Id_receive = data[0];
14  kelembaban_receive = data[1];
15
16
17      if (kelembaban_receive >= 0 && kelembaban_receive
18  <= 100) {
19        kelompok_kelembaban[id_receive] =
20  kelembaban_receive;
21        data_agregasi = 0;
22        jumlah_anggota = 0;
23        for (int x = 1; x < 10; x++) {
24          Serial.print(x);
25          Serial.print("=");
26          Serial.println(kelompok_kelembaban[x]);
27
28        if (kelompok_kelembaban[x] > 0) {
29          jumlah_anggota++;
30          data_agregasi += kelompok_kelembaban[x];
31        }
32      }
33        data_agregasi /= jumlah_anggota;
34        Serial.println();
35      }
36
37      if (millis() - m_kirim > waktu_kirim) {
38        m_kirim = millis();
```

```
39      Serial.println("--------------------------");
40      Serial.print("kirim data_agregasi=");
41      Serial.println(data_agregasi);
42      data[0] = id;
43      data[1] = data_agregasi;
44      nrf_send();
45    }
```

5.3.2 Penerimaan data agregasi pada node *sink*

Data agregasi yang telah dikrimkan oleh *cluster head* akan diterima oleh *sink.* Penerimaan data ini dilakukan pada method void loop() dengan memanggil method nrf_receive() terlebih dahulu. Lalu pada baris 5 dilakukan pemanggilan cek_data(), dimana method ini merupakan method untuk menampilkan data agregasi yang diterima dari *cluster head.* Pada baris 8-10, dilakukan seleksi kondisi dengan syarat apabila nilai variable detik lebih besar dari nilai interval_upload maka nilai detik akan diisikan menjadi 0 dan akan memanggil method upload() yang berfungsi untuk mengirimkan hasil data agregasi kepada *cloud.*

Tabel 5. 21 Implementasi penerimaan data agregasi pada node *sink*

```
Method loop()
 1  int interval_upload = 60;
 2  int detik;
 3  void loop() {
 4    nrf_receive();
 5    cek_data();
 6    delay(1000);
 7    detik++;
 8    if (detik > interval_upload) {
 9      detik = 0;
10      upload();
11    }
12
13  }
```

Pada method cek_data()yang terlihat pada tabel 5.17, dilakukan untuk menampilkan data agregasi yang telah diterima. Pada baris 10 terdapat seleksi kondisi apabila nilai daric h_1 lebih kecil dari 10 dan ch_2 lebih kecil dari 10 maka pada baris 11-16 dilakukan seleksi kondisi apabila nilai variable id sama dengan nilai ch_1 maka nilai variable data_1 akan diisi dengan nilai data[1] lalu selanjutnya akan ditampilkan pada serial monitor. Lalu, pada baris 18-23 juga dilakukan seleksi kondisi apabila nilai variable id sama dengan nilai ch_2 maka nilai variable data_2 akan diisi dengan nilai data[1] lalu selanjutnya akan ditampilkan pada serial monitor.

Tabel 5. 22 Implementasi menampilkan data pada node *sink*

Method cek_data()

```
1  int id;
2  int kelembaban;
3  int data[3];
4  int ch_1 = 10, ch_2 = 10;
5  int data_1, data_2;
6  unsigned long pm_ch1, pm_ch2;
7
8  void cek_data() {
9  ...
10   if (ch_1 < 10 && ch_2 < 10) {
11     if (id == ch_1) {
12       pm_ch1 = millis();
13       data_1 = data[1];
14       Serial.print("Data Agregasi dari Cluster Head 1
15 : ");
16       Serial.println(data_1);
17     }
18     if (id == ch_2) {
19       pm_ch2 = millis();
20       data_2 = data[1];
21       Serial.print("Data Agregasi dari Cluster Head 2
22 : ");
23       Serial.println(data_2);
24     }
25   }
26   Serial.println();
27   Serial.println();
   }
```

5.3.3 Pengiriman data agregasi dari node *sink* ke *cloud*

Pada implementasi untuk pengiriman data ke *cloud* terdapat method ShowSerialData() dan upload() pada *sink.* Method ShowSerialData() berisikan kode program untuk membaca data dari modul SIM800L dan mengirimkannya ke serial monitor.

Tabel 5. 23 Implementasi untuk modul SIM800L

Method ShowSerialData()

```
1  void ShowSerialData() {
2    while (SIM800L.available() != 0)
3      Serial.write(SIM800L.read());
4    delay(2000);
5  }
```

Sedangkan method upload() berisikan program untuk melakukan *upload* data yang dikumpulkan pada *sink* ke *cloud.* Pada baris 2-5 terdapat seleksi kondisi apabila modul SIM800L dapat melakukan pembacaan data dari SIM800L dan mengirimkan ke serial monitor lalu akan menampilkan tulisan "upload". Pada baris 7-13 dilakukan beberapa perintah untuk melakukan *register network,* melakukan koneksi ke GPRS *network, deactive GPRS PDP context* dan mengetahui status terkini dari koneksi. Lalu pada baris 16-20 merupakan kode program untuk melakukan pengaturan untuk APN atau *Access Point Name.* Pada baris 21-22 merupakan kode baris untuk mendapatkan IP address local. Lalu pada baris 28-34 merupakan kode untuk membuka koneksi TCP dan mengakses *Thingspeak* lalu mengirimkan data melalui koneksi tersebut. Pada baris 37 merupakan kode untuk mengakses *cloud* dan mengirimkan data dalam bentuk String pada variable str. Lalu pada baris 48 merupakan kode untuk menutup koneksi yang terbuka.

Tabel 5. 24 Implementasi untuk mengirim data pada *cloud* dari *sink*

Method upload()

```
 1  void upload() {
 2    if
 3  (SIM800L.available())Serial.write(SIM800L.read());
 4
 5    Serial.println("upload");
 6
 7    SIM800L.println("AT"); delay(1000);
 8    SIM800L.println("AT+CPIN?"); delay(1000);
 9    SIM800L.println("AT+CREG?"); delay(1000);
10    SIM800L.println("AT+CGATT?"); delay(1000);
11    SIM800L.println("AT+CIPSHUT"); delay(1000);
12    SIM800L.println("AT+CIPSTATUS"); delay(2000);
13    SIM800L.println("AT+CImUX=0");  delay(2000);
14
15    SIM800L.println("AT+CSTT=\"" + apn + "\"");
16  delay(1000);
17    ShowSerialData();
18    SIM800L.println("AT+CIICR");    delay(2000);
19    ShowSerialData();
20
21    SIM800L.println("AT+CIFSR");    delay(2000);
22    ShowSerialData();
23
24    SIM800L.println("AT+CIPSPRT=0"); delay(2000);
25    ShowSerialData();
26
27
28  SIM800L.println("AT+CIPSTART=\"TCP\",\"api.thingspeak.
29  com\",\"80\"");
30    delay(3000);
31    ShowSerialData();
32
33    SIM800L.println("AT+CIPSEND");
34    delay(2000);
```

```
35    Serial.println();
36    ShowSerialData();
37    String str ="GET
38  https://api.thingspeak.com/update?api_key=" +
39  Write_API_key + "&field1=" + String(data_1) +
40  "&field2=" + String(data_2);
41    Serial.println(str); delay(2000);
42    SIM800L.println(str); delay(4000);
43    ShowSerialData();
44
45    SIM800L.println((char)26); delay(4000);
46    SIM800L.println();
47    ShowSerialData();
48    SIM800L.println("AT+CIPSHUT");
49    delay(500);
50    ShowSerialData();
51    str = "";
52
53    pm_ch1 = millis();
54    pm_ch2 = millis();
55  }
```

BAB 6 PENGUJIAN SISTEM

Pengujian sistem dilakukan setelah melakukan tahap implementasi. Pengujian sistem dibuat untuk melihat apakah sistem yang dibuat telah sesuai dengan kebutuhan fungsional dari sistem. Pengujian juga dilakukan untuk mengetahui kinerja dari sistem yang telah dibuat. Pengujian sistem yang dilakukan meliputi pengujian fungsional dan pengujian non-fungsional. Pengujian fungsional didasari oleh kebutuhan fungsional. Sedangkan pengujian non-fungsional yang dilakukan terdapat

6.1 Pengujian Fungsional

Pengujian fungsional dilakukan untuk memastikan apakah sistem telah memenuhi kebutuhan fungsional yang telah dijelaskan.

6.1.1 Pengujian kode KF-01-01

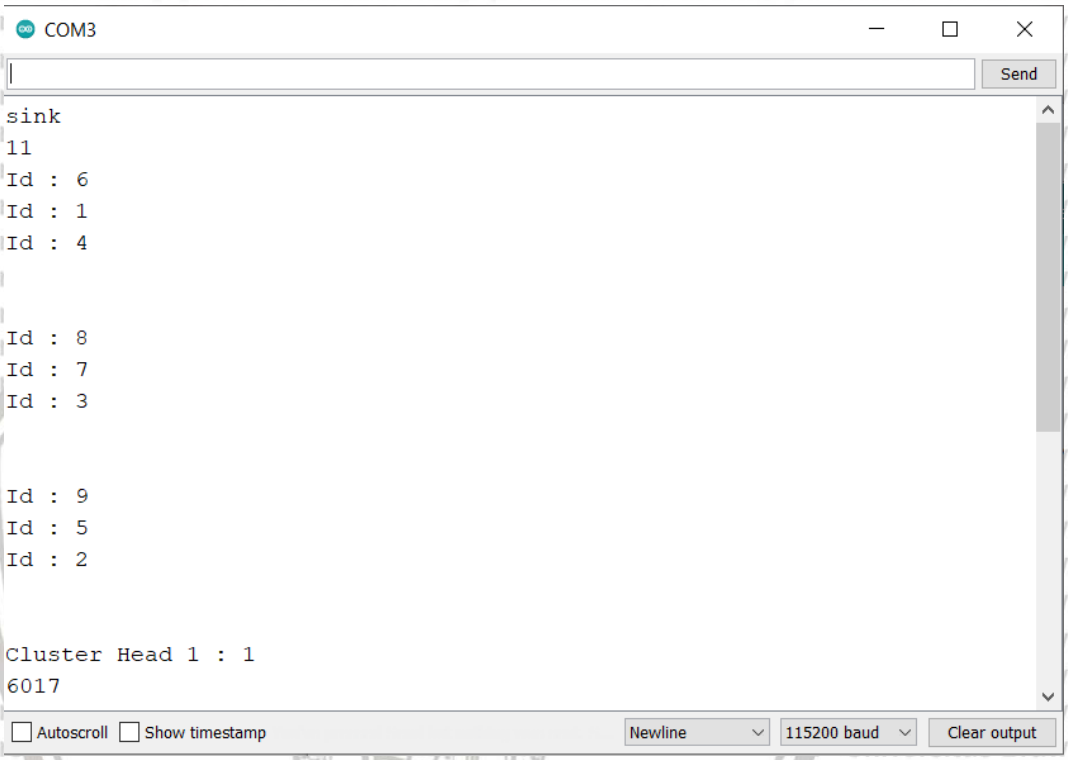
Pengujian kode KF-01-01 dilakukan dengan scenario pengujian dan memiliki hasil pengujian seperti yang ditunjukan pada Tabel 6.1

Tabel 6. 1 Tabel Pengujian Kode KF-01-01

Kode Fungsi	KF-01-01
Kebutuhan Fungsional	Node *sensor* harus dapat mengirimkan data angka kepada *sink*
Scenario Pengujian	1. Menyalakan seluruh node yang ada 2. Seluruh node mengirimkan informasi id berupa angka kepada *sink* 3. *Sink* menerima data angka sesuai dengan banyaknya jumlah node yang ada
Hasil yang Diharapkan	Node *sensor* dapat mengirimkan data angka sebagai id node kepada node *sink*
Hasil Pengujian	Node *sensor* berhasil mengirimkan data angka sebagai id node kepada node *sink*

Hasil pengujian kode KF-01-01 node *sensor* dapat mengirimkan data angka sebagai id node kepada node *sink.* Pada gambar 6.1 merupakan tampilan id dari node *sensor*. Sedangkan pada gambar 6.2 merupakan tampilan dari node *sink* saat menerima data id dari node *sensor.*

Gambar 6. 1 Id pada *sensor*



Gambar 6. 2 Data Id node yang diterima pada *sink*

6.1.2 Pengujian Kode KF-01-02

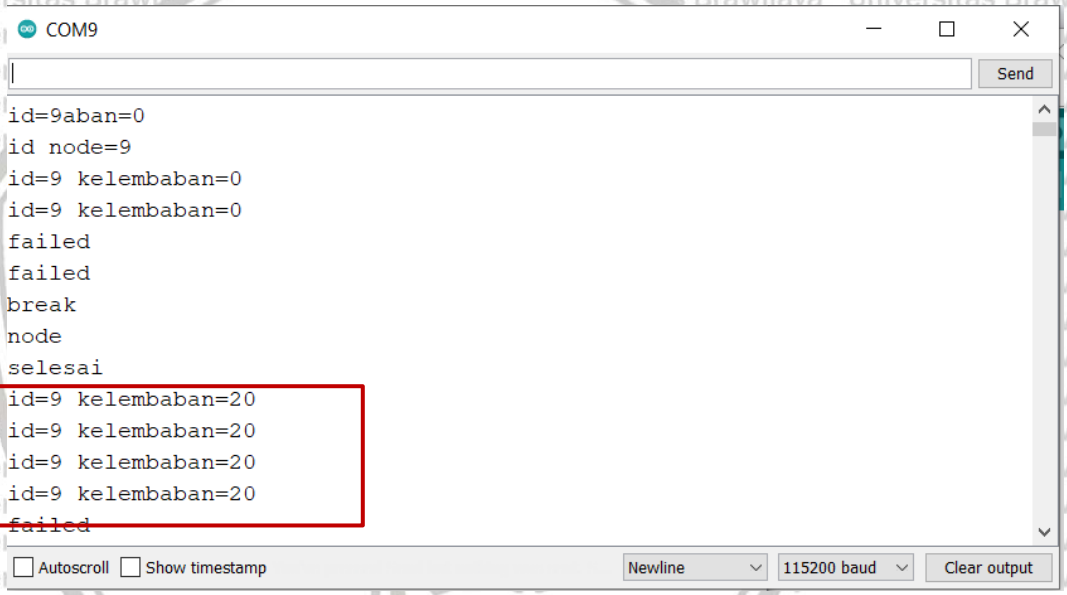
Pengujian kode KF-01-02 dilakukan dengan scenario pengujian dan memiliki hasil pengujian seperti yang ditunjukan pada Tabel 6.2

Tabel 6. 2 Tabel Pengujian Kode KF-01-02

Kode Fungsi	KF-01-02
Kebutuhan Fungsional	Node *sensor* harus dapat melakukan *data sensing* dan mendapatkan data kelembaban tanah
Scenario Pengujian	1. Menyalakan seluruh node *sensor* 2. Bergabung ke *cluster head*

	3. Melakukan *data sensing* dari tanah untuk mendapatkan data kelembapan tanah
Hasil yang Diharapkan	Node *sensor* dapat melakukan *data sensing* dan menampilkan serta mengirimkannya kepada *cluster head*
Hasil Pengujian	Node *sensor* berhasil melakukan *data sensing* dan menampilkan serta mengirimkannya kepada *cluster head*

Hasil pengujian kode KF-01-02 node *sensor* dapat melakukan *data sensing* dan menampilkan serta mengirimkannya kepada *cluster head.* Pada gambar 6.3 merupakan tampilan dari node *sensor* saat melakukan *data sensing.* Sedangkan pada gambar 6.4 merupakan tampilan dari *cluster head* saat menerima data dari node *sensor.*



Gambar 6. 3 *Data sensing* pada node *sensor*

Gambar 6. 4 Hasil *data sensing* pada *cluster head* dari node *sensor*

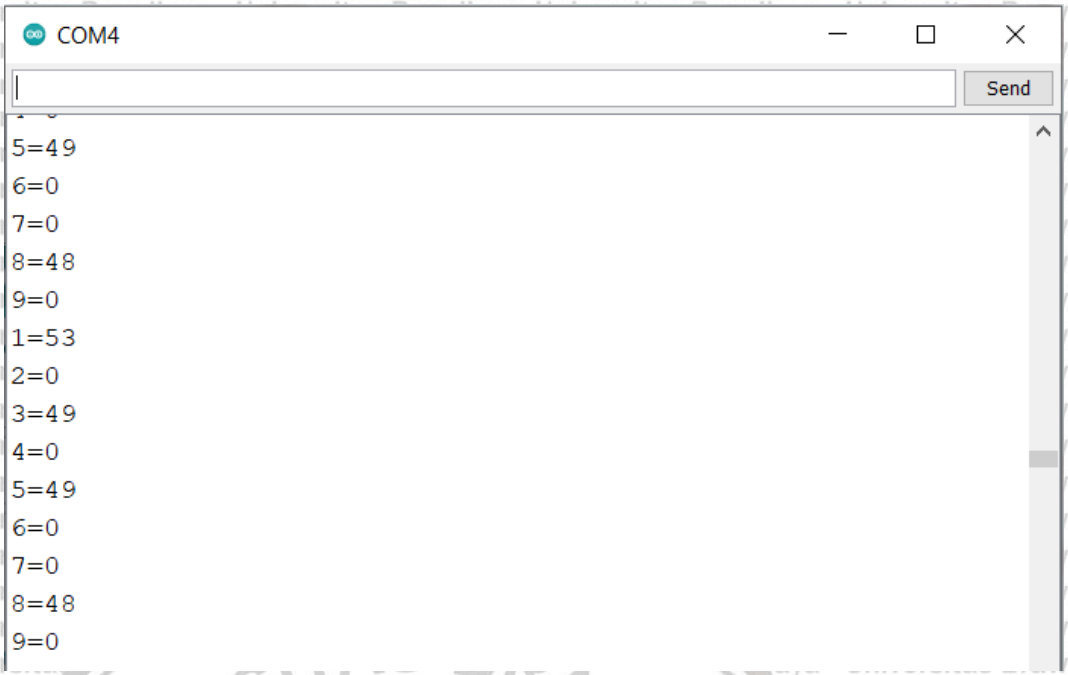
6.1.3 Pengujian Kode KF-01-03

Pengujian kode KF-01-03 dilakukan dengan scenario pengujian dan memiliki hasil pengujian seperti yang ditunjukan pada Tabel 6.3

Tabel 6. 3 Tabel Pengujian Kode KF-01-03

Kode Fungsi	KF-01-03
Kebutuhan Fungsional	Node *sensor* harus dapat mengirimkan data id node dan kelembapan data kepada *cluster head* yang terdekat untuk membentuk satu *cluster.*
Scenario Pengujian	1. Sink mengirimkan info node sensor yang terpilih menjadi cluster head kepada sisa node sensor yang tidak terpilih 2. Node sensor yang tidak terpilih akan mencari id cluster head terdekat 3. Setelah mengetahui cluster head yang terdekat, node sensor mengirimkan data tersebut kepada cluster head tersebut
Hasil yang Diharapkan	Node *sensor* dapat mengirimkan data id node dan kelembapan data kepada *cluster head* yang terdekat untuk membentuk satu *cluster.*
Hasil Pengujian	Node *sensor* berhasil mengirimkan data id node dan kelembapan data kepada *cluster head* yang terdekat untuk membentuk satu *cluster.*

Hasil pengujian kode KF-01-03 node *sensor* dapat mengirimkan data id node dan kelembapan data kepada *cluster head* yang terdekat untuk membentuk satu *cluster.* Pada gambar 6.5 merupakan tampilan dari *cluster head* saat menerima data id dan kelembapan tanah dari node *sensor* yang terdekat.



Gambar 6. 5 Hasil *data sensing* yang diterima pada *cluster head*

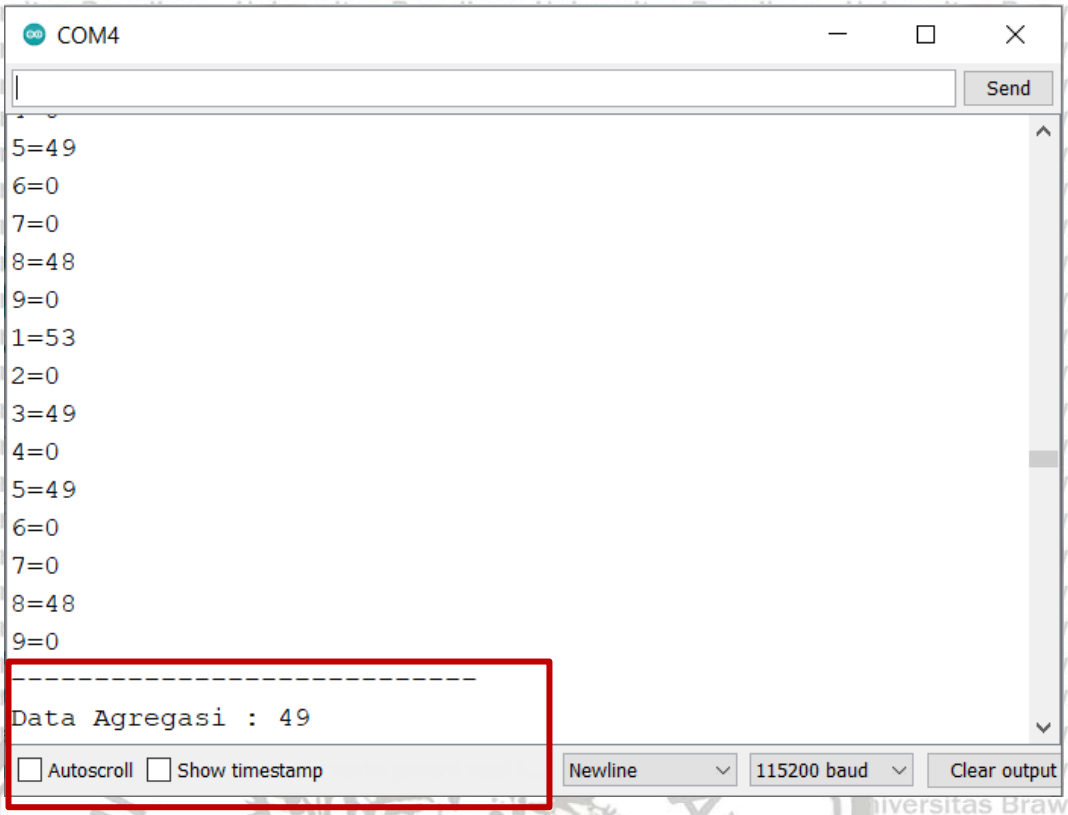
6.1.4 Pengujian kode KF-01-04

Pengujian kode KF-01-04 dilakukan dengan scenario pengujian dan memiliki hasil pengujian seperti yang ditunjukan pada Tabel 6.4

Tabel 6. 4 Tabel Pengujian Kode KF-01-04

Kode Fungsi	KF-01-04
Kebutuhan Fungsional	Node *sensor* yang terpilih sebagai *cluster head* harus dapat melakukan proses agregasi data setelah menerima data dari *node* sensor lainnya
Scenario Pengujian	1. Node *sensor* mengirimkan data kepada *cluster head* 2. *Cluster head* akan melakukan proses agregasi data kepada data yang sebelumnya telah diterima
Hasil yang Diharapkan	*Cluster head* dapat melakukan proses agregasi data dari data yang telah diterima dari node *sensor*
Hasil Pengujian	*Cluster head* berhasil melakukan proses agregasi data dari data yang telah diterima dari node *sensor*

Hasil pengujian kode KF-01-04 *cluster head* dapat melakukan proses agregasi data dari data yang telah diterima dari node *sensor*. Pada gambar 6.6 merupakan tampilan dari *cluster head* saat menerima data id dan kelembapan tanah dari node *sensor* yang terdekat dan menampilkan hasil agregasi data yang telah dilakukan.



Gambar 6. 6 Hasil agregasi data pada *cluster head*

6.1.5 Pengujian Kode KF-02-01

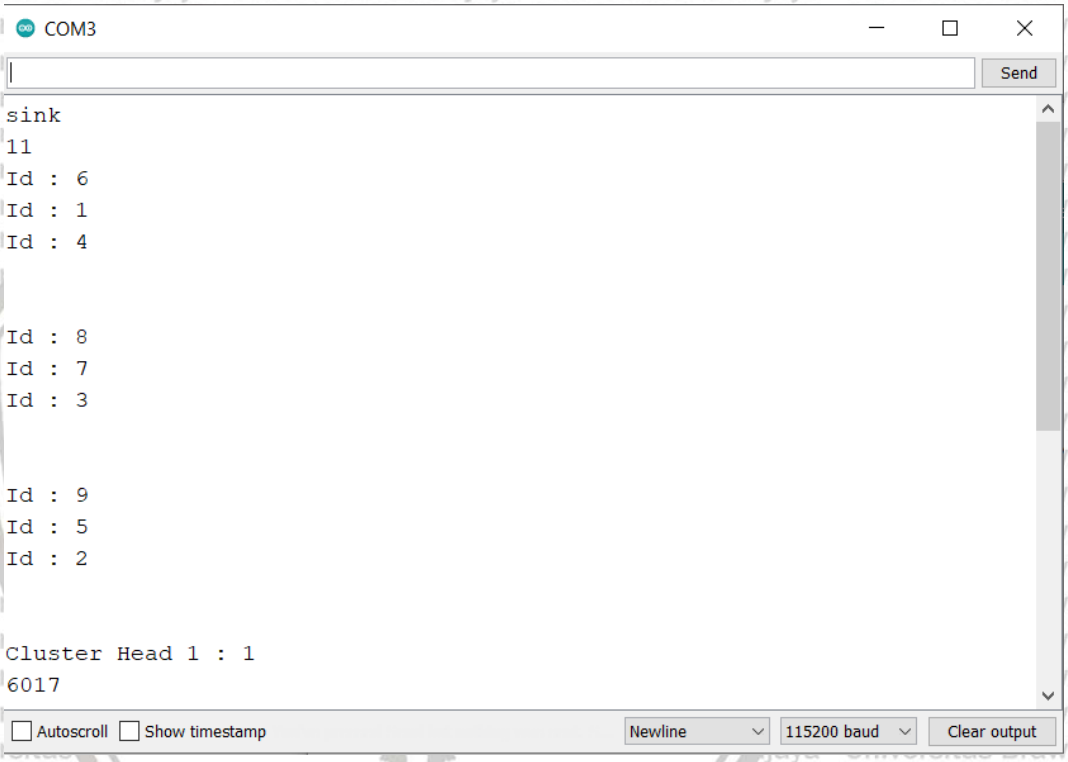
Pengujian kode KF-02-01 dilakukan dengan scenario pengujian dan memiliki hasil pengujian seperti yang ditunjukan pada Tabel 6.4

Tabel 6. 5 Tabel Pengujian Kode KF-02-01

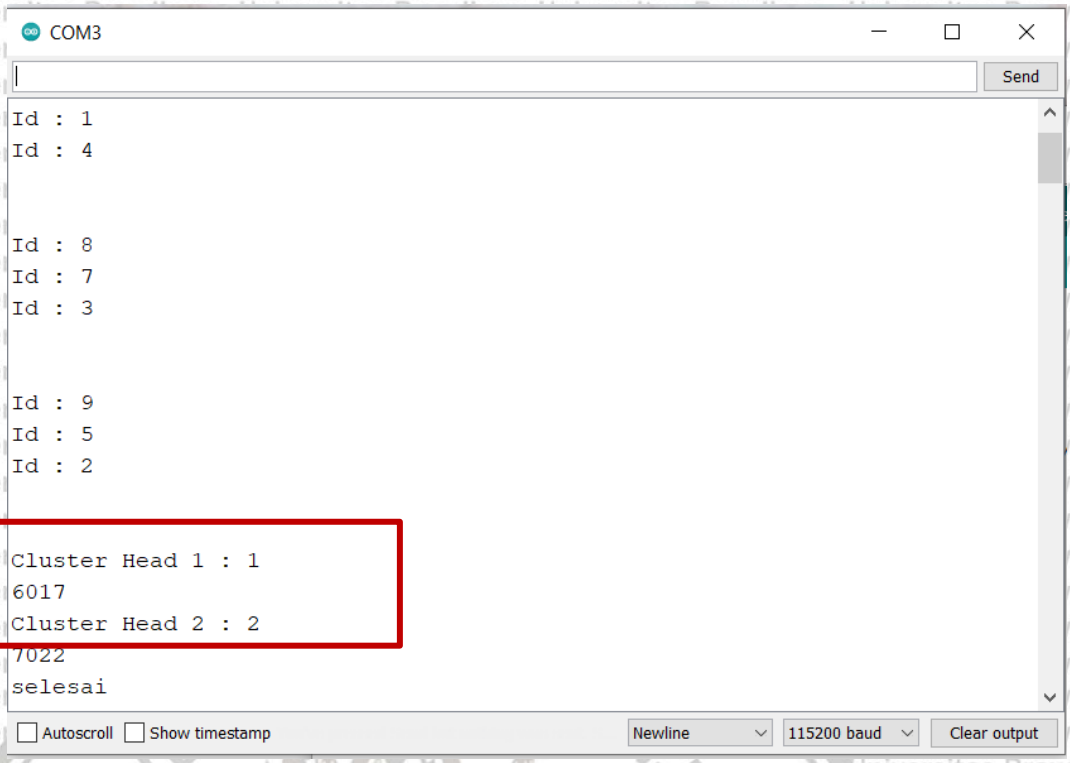
Kode Fungsi	KF-02-01
Kebutuhan Fungsional	Node *sink* dapat menentukan *cluster head* dengan memilih dua angka terkecil dari node *sensor*
Scenario Pengujian	1. Menyalakan seluruh node yang ada 2. Seluruh node mengirimkan angka kepada sink 3. Sink menerima data angka sesuai dengan banyaknya jumlah node yang ada 4. Sink mengurutkan angka yang diterima dari terkecil hingga terbesar 5. Sink memilih angka terkecil sebagai cluster head dan menginfokan kepada node *sensor*
Hasil yang Diharapkan	Node *sink* dapat menerima id node dari tiap node *sensor* dan menentukan dua *cluster head* dari pemilihan angka terkecil

Hasil Pengujian	Node *sink* berhasil menerima id node dari tiap node *sensor* dan menentukan dua *cluster head* dari pemilihan angka terkecil

Hasil pengujian kode KF-02-01 node *sink* dapat menerima id node dari tiap node *sensor* dan menentukan dua *cluster head* dari pemilihan angka terkecil Pada gambar 6.7 merupakan tampilan dari *sink* yang mendapatkan data id dari node *sensor* yang menyala. Sedangkan pada gambar 6.8 merupakan tampilan dari node *sink* saat telah menentukan *cluster head* dan hasil *cluster head* yang telah dipilih.



Gambar 6. 7 Id node yang diterima pada node *sink*



Gambar 6. 8 Cluster Head yang terpilih pada node *sink*

6.1.6 Pengujian Kode KF-02-02

Pengujian kode KF-02-02 dilakukan dengan scenario pengujian dan memiliki hasil pengujian seperti yang ditunjukan pada Tabel 6.5

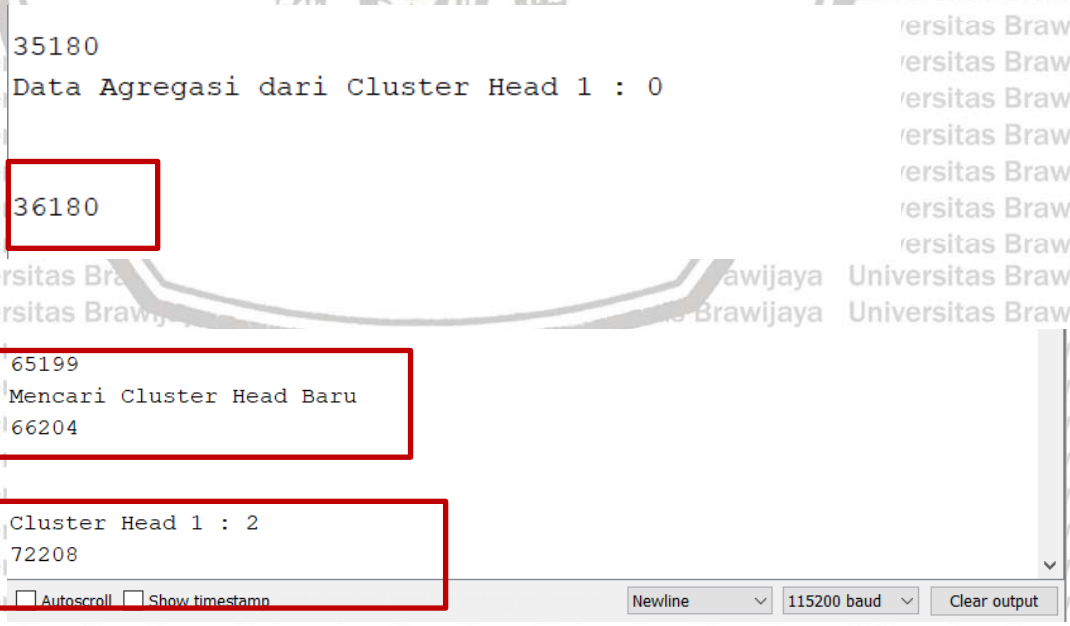
Tabel 6. 6 Tabel Pengujian Kode KF-02-02

Kode Fungsi	KF-02-02
Kebutuhan Fungsional	Node *sink* dapat mencari *cluster head* baru ketika node *sink* tidak menerima data agregasi dari *cluster head* lebih dari 30 detik
Scenario Pengujian	1. Pemilihan *cluster head* dilakukan 2. *Sink* menerima data agregasi dari *cluster head* 3. *Sink* menunggu hingga 30 detik untuk memastikan apakah *cluster head* belum mati 4. Setelah 30 detik tidak menerima data, *sink* akan melakukan pencarian *cluster head* baru 5. *Cluster head* baru dipilih

Hasil yang Diharapkan	Node *sink* dapat memilih *cluster head* yang baru
Hasil Pengujian	Node *sink* berhasil memilih *cluster head* yang baru

Hasil pengujian kode KF-02-02 node *sink* dapat memilih *cluster head* yang baru. Pada gambar 6.9 merupakan tampilan saat *sink* memilih *cluster head* awal. Sedangkan pada gambar 6.10 merupakan tampilan saat *sink* menunggu selama 30 detik untuk menerima data agregasi dari *cluster head* dan apabila tidak menerima maka mulai mencari *cluster head* baru dan mendapatkan hasil *cluster head* baru.

Gambar 6. 9 Pencarian *cluster head* awal



Gambar 6. 10 *sink* menunggu 30 detik dan memilih *cluster head* baru

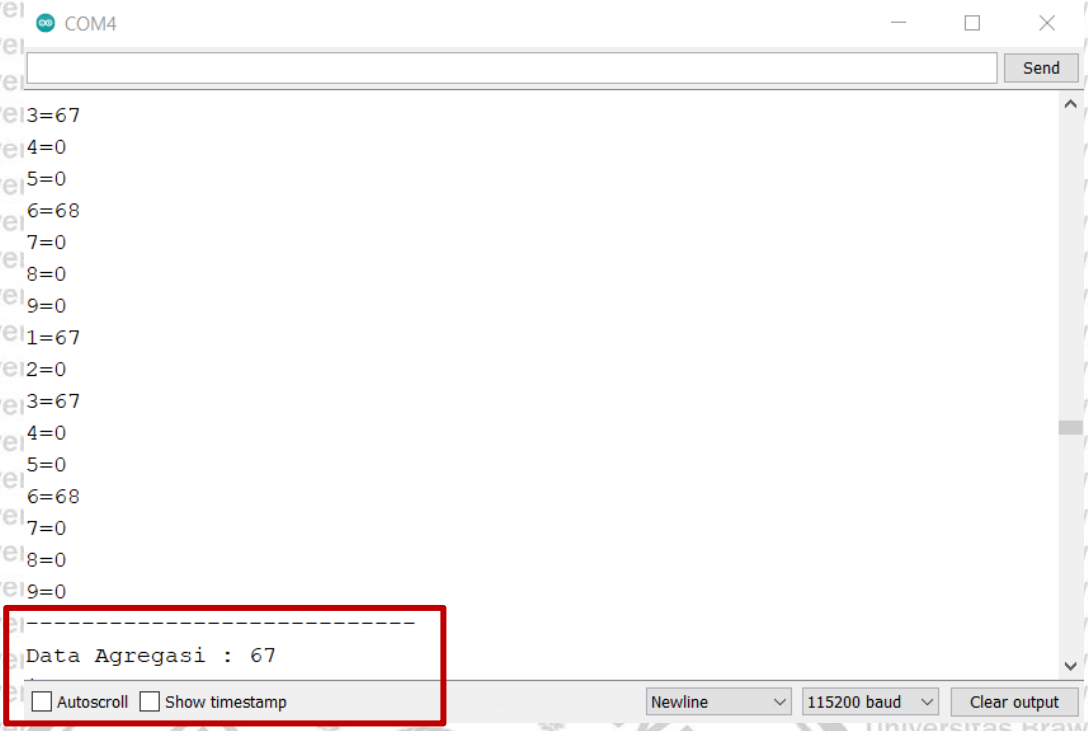
6.1.7 Pengujian Kode KF-02-03

Pengujian kode KF-02-03 dilakukan dengan scenario pengujian dan memiliki hasil pengujian seperti yang ditunjukan pada Tabel 6.5

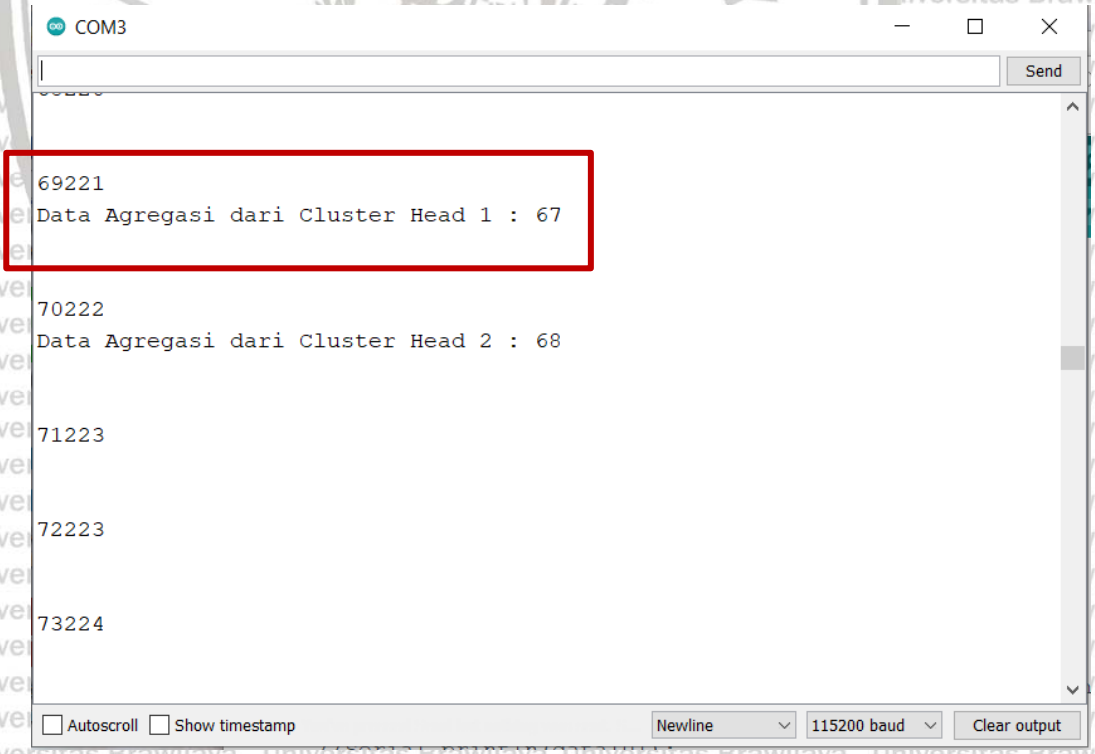
Tabel 6. 7 Tabel Pengujian Kode KF-02-03

Kode Fungsi	KF-02-03
Kebutuhan Fungsional	Node *sink* harus dapat menerima informasi dari *cluster head* yang mengirimkan data
Scenario Pengujian	1. Menyalakan node *Soil Moisture Sensor* sebagai node *sensor* 2. Mengimplementasikan kode program untuk menerima informasi dari *cluster head* 3. Menampilkan data pada *Serial Monitor*
Hasil yang Diharapkan	Node *sink* dapat menerima data dari *cluster head* dan menampilkan data pada *serial monitor*
Hasil Pengujian	Node *sink* berhasil menerima data dari *cluster head* dan menampilkan data pada *serial monitor*

Hasil pengujian kode KF-02-03 node *sink* dapat menerima data dari *cluster head* dan menampilkan data pada *serial monitor*. Pada gambar 6.11 merupakan tampilan dari data agregasi yang ada pada *cluster head*. Sedangkan pada gambar 6.12 merupakan tampilan dari node *sink* yang menerima data agregasi dari *cluster head.*



Gambar 6. 11 Data agregasi pada *cluster Head*



Gambar 6. 12 Data agregasi yang diterima paad node *sink*

6.1.8 Pengujian Kode KF-02-04

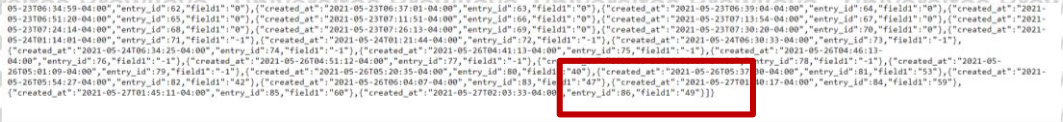
Pengujian kode KF-02-03 dilakukan dengan scenario pengujian dan memiliki hasil pengujian seperti yang ditunjukan pada Tabel 6.7

Tabel 6. 8 Tabel Pengujian Kode KF-02-04

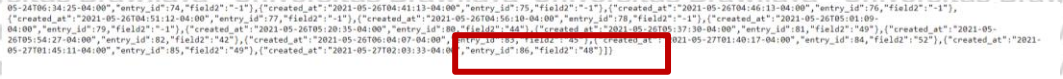
Kode Fungsi	KF-02-04
Kebutuhan Fungsional	Node *sink* harus dapat melakukan pengiriman data agregasi yang diterima ke *cloud*
Scenario Pengujian	1. Node *sink* menghubungkan ke *cloud* 2. Node *sink* melakukan pengiriman data 3. Menampilkan data pada *cloud*
Hasil yang Diharapkan	Node *sink* dapat melakukan pengiriman data hasil agregasi ke *cloud*
Hasil Pengujian	Node *sink* berhasil melakukan pengiriman data hasil agregasi ke *cloud*

Hasil pengujian kode KF-02-03 node *sink* data dari *cluster head* dan menampilkan data pada *serial monitor*. Pada gambar 6.13 merupakan tampilan *serial monitor* pada *sink* yang menampilkan proses pengiriman data agregasi ke *cloud*. Proses pengiriman yang berhasil dapat dilihat dari status "SEND OK" dan terdapatnya *entry id* dari data yang dikirim kepada *cloud.* Sedangkan pada gambar 6.14 merupakan tampilan dari data di *cloud* yang berupa JSON yang menerima data agregasi dari *sink* pada *field 1* dengan *entry id* yang sama dengan yang telah dikirim dari node *sink.* Sedangkan pada gambar 6. 15merupakan tampilan dari data di *cloud* yang berupa JSON yang menerima data agregasi dari *sink* pada *field 2* dengan *entry id* yang sama dengan yang telah dikirim dari node *sink*

Gambar 6. 13 Pengiriman data agregasi ke *cloud*



Gambar 6. 14 Data agregasi yang diterima *cloud* pada *field 1*



Gambar 6. 15 Data agregasi yang diterima *cloud* pada *field* 2

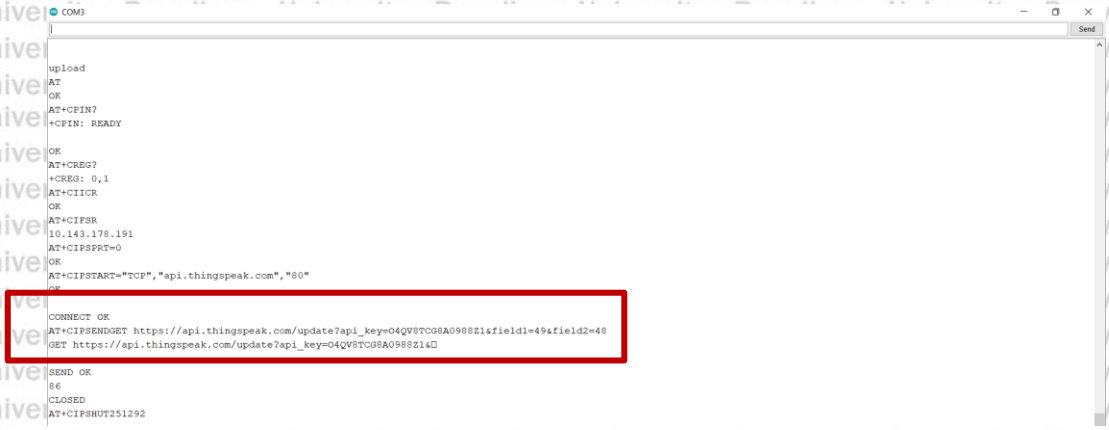
6.1.9 Pengujian Kode KF-03-01

Pengujian kode KF-03-01 dilakukan dengan scenario pengujian dan memiliki hasil pengujian seperti yang ditunjukan pada Tabel 6.8

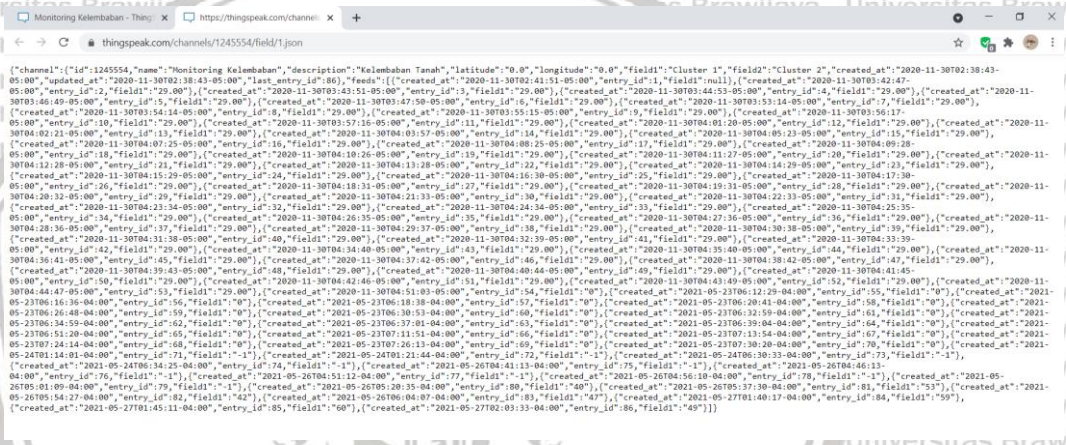
Tabel 6. 9 Tabel Pengujian Kode KF-03-01

Kode Fungsi	KF-03-01
Kebutuhan Fungsional	*Cloud* harus dapat menerima data terbaru dari *sink*
Scenario Pengujian	*1.* Node *sink* menghubungkan ke *cloud* *2.* *Cloud* menerima data dari node *sink*
Hasil yang Diharapkan	*Cloud* dapat menerima hasil data agregasi dari node *sink*
Hasil Pengujian	*Cloud* berhasil menerima hasil data agregasi dari node *sink*

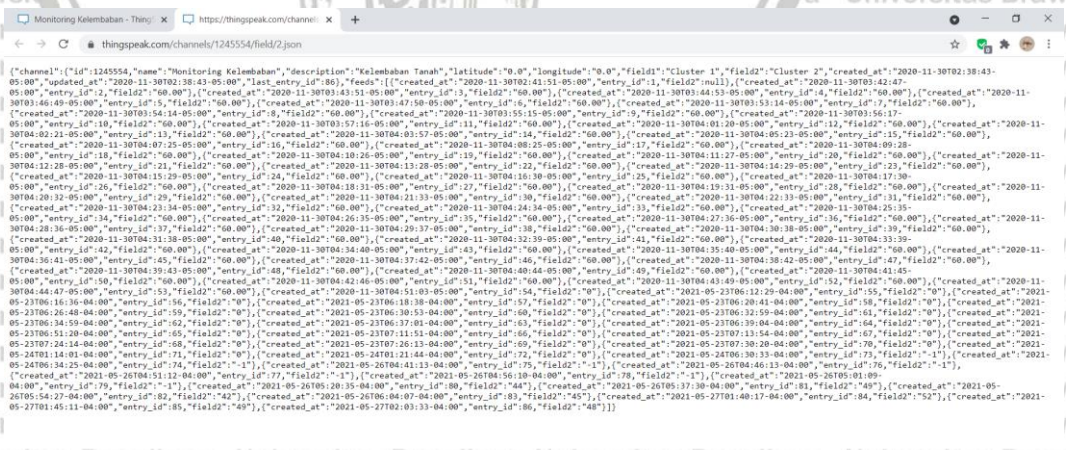
Hasil pengujian kode KF-03-01 *cloud* dapat menerima hasil data agregasi dari node *sink*. Pada gambar 6.16 merupakan tampilan dari *sink* yang sedang menghubungkan ke *cloud.* Proses mengakses *cloud* dari node *sink* adalah dengan mengakses link yang terdapat pada variable “str” pada program yang ditampilkan pada *serial monitor.* Sedangkan pada gambar 6.17 merupakan tampilan dari data yang diterima *cloud* berupa JSON yang menerima data agregasi dari *sink* pada *field 1.* Sedangkan pada gambar 6.18 merupakan tampilan dari data yang diterima *cloud* berupa JSON yang menerima data agregasi dari *sink* pada *field 2.*



Gambar 6. 16 Proses mengirimkan data dari node *sink* ke *cloud*



Gambar 6. 17 Data agregasi yang diterima *cloud* untuk *cluster head* 1



Gambar 6. 18 Data agregasi yang diterima *cloud* untuk *cluster head* 2

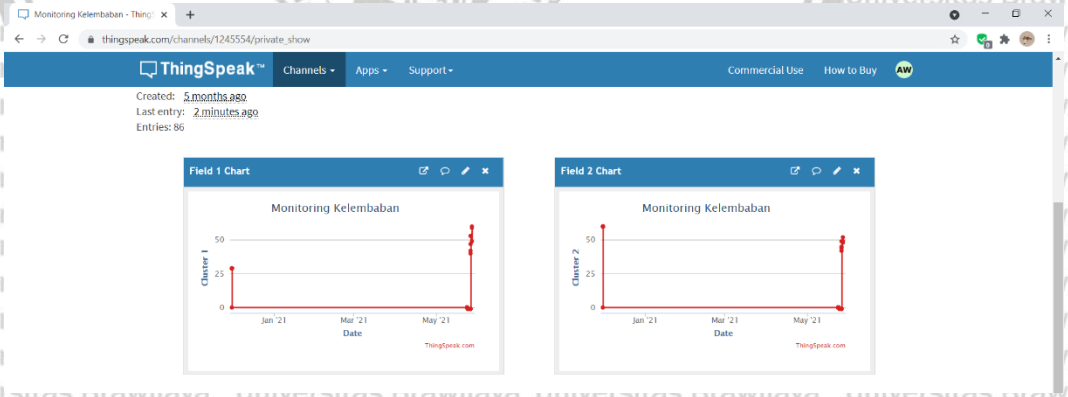
6.1.10 Pengujian Kode KF-03-02

Pengujian kode KF-03-02 dilakukan dengan scenario pengujian dan memiliki hasil pengujian seperti yang ditunjukan pada Tabel 6.9

Tabel 6. 10 Tabel Pengujian Kode KF-03-02

Kode Fungsi	KF-03-02
Kebutuhan Fungsional	*Cloud* harus dapat menampilkan data terbaru yang diterima dari node *sink*
Scenario Pengujian	1. Node *sink* menghubungkan *cloud* 2. *Cloud* menerima data dari node *sink* 3. *Cloud* menampilkan data yang diterima pada layar
Hasil yang Diharapkan	*Cloud* dapat menampilkan data terbaru yang dikirimkan oleh node *sink*
Hasil Pengujian	*Cloud* berhasil menampilkan data terbaru yang dikirimkan oleh node *sink*

Hasil pengujian kode KF-03-02 *cloud* dapat menampilkan data terbaru yang dikirimkan oleh node *sink*. Pada gambar 6.19 merupakan tampilan diagram yang ditampilkan pada *cloud* yang telah menerima data agregasi yang telah dikirimkan oleh node *sink.*



Gambar 6. 19 Data yang ditampilkan pada *cloud*

6.2 Pengujian Non Fungsional

6.2.1 Pengujian *Response Time* terhadap pencarian *Cluster Head* baru

Pengujian pencarian *cluster head* baru dilakukan untuk menguji *response time* dari program untuk melihat berapa lama waktu yang diperlukan untuk program mencari *cluster head baru* apabila terdapat *cluster head* yang telah mati. Pengujian ini dihitung disaat program memulai untuk menjalankan pencarian

cluster head baru hingga *cluster head* yang baru ditemukan. Pengujian ini dilakukan sebanyak 10 kali.

Tabel 6.11 merupakan tabel program untuk menampilkan waktu pada *serial monitor.* Pada program baris ke 2 ditambahkan perintah untuk mencetak fungsi millis() yang merupakan fungsi untuk menjalankan waktu internal pada program dengan satuan *miliseconds.*

Tabel 6. 11 Kode program untuk menampilan waktu pencarian *cluster head* baru

Method loop()

```
1  void pilih_ch() {
2    Serial.println(millis());
3    ch_1=10;
4    ch_2=10;
5    pm = millis();
6  ...
7  }
```

Tabel 6.12 merupakan hasil pengujian *response time* yang telah dilakukan 10 kali. Pengujian dilakukan dengan menghitung selisih waktu dari saat program memulai mencari *cluster head* dengan waktu saat *cluster head* baru telah dipilih. Dari tabel tersebut didapatkan bahwa rata-rata *response time* sebesar 6023,4 *miliseconds.*

Tabel 6. 12 *Response Time* untuk pencarian *cluster head* baru

Percobaan	*Response Time (miliseconds)*
1	6002
2	6003
3	6033
4	6028
5	6033
6	6002
7	6033
8	6033
9	6034
10	6033

Rata-rata	6023,4

6.2.2 Pengujian *Energy Efficiency*

Pengujian ini dilakukan untuk mengetahui apakah node *sink* yang menerima data agregasi memiliki *energy efficiency* yang lebih baik daripada dari node *sink* yang menerima data yang tidak mengalami agregasi. Pengujian ini dilakukan dengan mencoba melakukan pada WSN dengan menyalakan node *sink* yang menerima data agregasi dan node *sink* yang tidak menerima data agregasi. Node diberikan baterai dengan nilai *idle* 4.1V dan 0.37A. Kedua node akan diperhatikan dan dihitung lama waktu menyala lalu dibandingkan dari lama kedua WSN menyala. Dari hasil pengujian dibawah dapat dilihat bahwa lama waktu menyala dari *sink* yang menerima data agregasi dengan *sink* yang menerima data langsung terdapat perbedaan. Lama waktu menyala dari *sink* yang menerima data agregasi adalah selama 500 menit atau 8.3 jam terhitung dari pukul 15.40 hingga 23.55, sedangkan lama waktu menyala dari *sink* yang menerima data langsung tanpa agregasi data adalah selama 470 menit atau 7.83 jam terhitung dari pukul 13.10 hingga 21.00. Sehingga hasil dari pengujian ini menunjukkan bahwa lama waktu menyala dari *sink* yang menerima data agregasi lebih lama dari *sink* yang menerima data tanpa dilakukan agregasi data. Untuk menghitung energi yang digunakan pada dua *sink* digunakan persamaan

$$battery\ life = \frac{battery\ capacity}{loadcurrent}$$

Sumber: (Freeman, Ph.D., 2016)

Untuk persamaan diatas, *battery life* merupakan lamanya baterai dapat bertahan, lalu *battery capacity* merupakan kapasitas dari baterai yang mana sebesar 4800 mAh dan *loadcurrent* merupakan beban daya yang digunakan. Untuk node *sink* pertama yang menerima data agregasi didapatkan *loadcurrent* sebesar 578.3 mA dan untuk *loadcurrent* dari node *sink* kedua yang tidak menerima data agregasi adalah sebesar 613 mA. Dari hasil yang didapat terlihat bahwa node *sink* yang tidak menerima data agregasi menggunakan beban daya lebih besar dibandingkan oleh node *sink* yang menerima data agregasi.

Untuk membuat *sink* menerima data tanpa dilakukan agregasi data maka pada node *sensor* terdapat kode program pada method loop() untuk melakukan *data sensing* dan mengirimkannya kepada node *sink*. Pada baris 6 merupakan kode program untuk melakukan *data sensing,* yang lalu pada baris 8 data id node akan disimpan terlebih dahulu disimpan pada data[0] lalu hasil *data sensing* disimpan pada data[1] seperti pada baris 9. Setelah itu data akan dikirimkan kepada node *sink*.

Tabel 6. 13 Kode program pada node *sensor* untuk melakukan data sensing dan tanpa melakukan agregasi data

Method void loop()

```
1  void loop() {
2      if (millis() - m_kirim > waktu_kirim) {
3      m_kirim = millis();
4      Serial.println("-----------------------------");
5      Serial.print("Kirim data ");
6      kelembaban = map(analogRead(pin_sensor), 0,
7  1023, 100, 0);
8      data[0] = id;
9      data[1] = kelembaban;
10     nrf_send();
11     }
12   }
```

Lalu pada *sink,* terdapat kode program untuk menerima hasil *data sensing* yang telah dikirimkan dari node *sensor.* Pertama pada method loop() melakukan penerimaan data terlebih dahulu, lalu akan memanggil method cek_data() dimana method cek_data() pada baris 10-19 berisi untuk menerima data yang telah dikirimkan dari node *sensor* lalu mengisikan data[0] yang dikirim pada variable id dan data[1] pada variable kelembapan untuk selanjutnya ditampilkan pada *serial monitor.*

Tabel 6. 14 Kode program pada *sink* untuk menerima hasil *data sensing* tanpa agregasi data

sink

```
1  void loop() {
2    nrf_receive();
3    cek_data();
4
5    delay(1000);
6
7    ...
8    }
9
10 void cek_data() {
11   nrf_receive();
12   id = data[0];
13   kelembaban= data[1];
14   Serial.print("Id : ");
15   Serial.print(id);
16   Serial.print("Kelembaban : ");
17   Serial.print(kelembaban);
18 }
```

6.2.3 Pengujian Jarak antar Node

Pengujian jarak ini dilakukan untuk menemukan jarak yang maksimal antara node apabila menggunakan modul komunikasi nRF240L1. Pengujian dilakukan pada area penuh hambatan seperti adanya pohon, bangunan, *antenna* dari perangkat lain dan pemancar sinyal pada area tersebut. Pengujian dilakukan dengan beberapa jarak berbeda hingga ditemukan jarak maksimal untuk menerima data antar node yang telah menyala. Pada pengujian ini node yang digunakan merupakan node sensor dengan id 1, 2 dan 9 serta node *sink* yang berperan untuk menerima data yang dikirimkan dari node *sensor.*

Tabel 6.15 Merupakan hasil dari pengujian yang telah dilakukan. Pengujian dilakukan dengan mencoba beberapa jarak untuk mengirimkan data antar node.

Tabel 6. 15 Hasil pengujian jarak untuk WSN

No	Jarak	Id Node: 1	Id Node: 2	Id Node: 9	Waktu data diterima (*miliseconds)*
1	1 Meter	Berhasil	Berhasil	Berhasil	6003
2	3 Meter	Berhasil	Berhasil	Berhasil	6019
3	5 Meter	Berhasil	Berhasil	Berhasil	6014
4	10 Meter	Berhasil	Berhasil	Berhasil	6017
5	25 Meter	Berhasil	Berhasil	Berhasil	6021
6	50 Meter	Berhasil	Berhasil	Berhasil	30104
7	100 Meter	Tidak Berhasil	Tidak Berhasil	Tidak Berhasil	-
8	150 Meter	Tidak Berhasil	Tidak Berhasil	Tidak Berhasil	-

Dari hasil pengujian yang dilakukan pada beberapa jarak tertentu didapatkan hasil seperti tabel diatas. Keberhasilan pengujian didapatkan dari berhasil diterimanya data yang dikirimkan dari node ke *sink.* Dari percobaan yang dilakukan, tingkat keberhasilan sehingga tidak menyebabkan *packet loss* adalah sebesar 75%.

BAB 7 PENUTUP

Bab ini berisi kesimpulan dan saran terhadap penelitian yang akan dilakukan. Kesimpulan dan saran didapatkan berdasarkan dari hasil perancangan, implementasi hingga pengujian.

7.1 Kesimpulan

Kesimpulan yang didapatkan dari penelitian yang telah dilakukan adalah:

1. *Cluster Based Algorithm* untuk melakukan agregasi data pada *Wireless Sensor Network* (WSN) berhasil diimplementasikan. Implementasi *Cluster Based Algorithm* pada penelitian menggunakan protokol *Low Energy Adaptive Clustering Hierarchy* (LEACH) dimana diterapkan pada satu node *sink* dan sembilan node *sensor.* Waktu response pencarian *cluster head* baru memiliki rata-rata sebesar 6023,4 *miliseconds.*
2. Data kelembapan tanah yang telah didapatkan akan dilakukan proses agregasi data dengan menggunakan fungsi *mean,* dimana dengan fungsi ini maka agregasi data akan dilakukan dengan menemukan rata-rata hasil nilai kelembapan tanah dari node sensor pada satu *cluster* yang sama. Agregasi data dilakukan untuk menghemat energi dari node dengan mengurangi jumlah redudansi data. Lalu selanjutnya hasil agregasi data akan dikirimkan kepada *cloud.*
3. Node sink berhasil meneruskan data agregasi pada cloud yang dikumpulkan dari node sensor dan di agregasi oleh *cluster head*. Node *sink* mengirimkan data ke *cloud* melalui modul komunikasi GSM/GPRS.

7.2 Saran

Sistem yang dibangun pada penelitian ini belum sempurna sehingga perlu dilakukan penelitian untuk pengembangan sistem selanjutnya. Adapun saran yang dapat peneliti sarankan yaitu:

1. Menggunakan protokol selain LEACH dalam mengimplementasikan *Cluster Based Algorithm* dikarenakan protokol LEACH masih memiliki beberapa kelemahan dan kekurangan dalam sistem yang telah dibuat. Protokol lainnya yang dapat digunakan adalah seperti LEACH-C, *Power Efficient Gathering in Sensor Information Systems* (PEGASIS), *Hybrid Energy Efficient Distributed* (HEED) dan *Threshold Sensitive Energy Efficient Sensor Network* (TEEN).
2. Menambahkan objek lain untuk dilakukan *data sensing* pada WSN yang akan dibangun. Objek yang ditambahkan akan dapat menambahkan fungsi yang lebih baik pada WSN.

DAFTAR REFERENSI

Aung, T. H., Mon, S. S. Y. & Nwe, C. M., 2016. COMPARATIVE STUDY ON ROUTING PROTOCOLS OF WIRELESS SENSOR NETWORK FOR PRECISION AGRICULTURE. *International Journal of Advanced Computational Engineering and Networking,* 4(5), pp. 29-33.

Aung, T. H. et al., 2016. Implementation Of The Precision Agriculture Using LEACH Protocol Of Wireless Sensor Network. *INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH.*

Cahyadi, W., Wahyudi, M. A. & Sarwono, C. S., 2018. Analisis Perbandingan Konsumsi Energi dan Masa Hidup Jaringan pada Protokol LEACH, HEED, dan PEGASIS di Wireless Sesnor Network. *Jurnal Rekayasa Elektrika,* Volume 14, pp. 128-135.

Chan, H. F. & Rudolph, H., 2015. *New Energy Efficient Routing Algorithm for Wireless Sensor Network.* s.l., s.n.

D, R., 2015. *BPTP Kaltim.* [Online] Available at: http://kaltim.litbang.pertanian.go.id/ind/index.php?option=com_content&view=article&id=735&Itemid=59

Evangeline, C. S., Lenin, A., Chandra, J. K. & Prasath, J., 2019. Monitoring and Control of Vital Parameters in Greenhouse using Internet of Things. *International Journal of Innovative Technology and Exploring Engineering,* 8(9), p. 850.

Freeman, Ph.D., B. A., 2016. *https://hearinghealthmatters.org/.* [Online] Available at: https://hearinghealthmatters.org/waynesworld/2016/batterylifecounseling/ [Accessed 15 June 2021].

Huo, J., Deng, X. & Al-Neshmi, H. M. M., 2020. Design and Improvement of Routing Protocol for Field Observation Instrument Networking Based on LEACH Protocol. *Journal of Electrical and Computer Engineering,* Volume 2020.

Husdi, 2018. Monitoring Kelembaban Tanah Pertanian Menggunakan Soil Moisture Sensor FC-28 dan Arduino Uno. *ILKOM Jurnal Ilmiah,* Volume 10, p. 237.

Kadir, A., 2013. *Panduan Praktis Mempelajari Aplikasi Mikrokontroler dan Pemrogramannya menggunakan Arduino.* Yogyakarta: Penerbit Andi.

Kaur, M. & Munjal, A., 2020. Data aggregation algorithms for wireless sensor network: A review. *Ad Hoc Networks.*

Khalifeh, A., Abid, H. & Darabkh, K. A., 2020. *Optimal Cluster Head Positioning Algorithm for Wireless Sensor Networks.* [Online] Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7374284/ [Accessed 16 February 2021].

Nordic Semiconductor, 2006. *Preliminary Product Specification Single chip 2.4 GHz Transceiver,* Norway: Nordic Semiconductor ASA.

Pourghebleh, B. & Navimipour, N. J., 2017. Data aggregation mechanisms in the Internet of things: A systematic review of the literature and recommendations for future research. *Journal of Network and Computer Applications,* Volume 97, pp. 23-24.

Rutledge, K. et al., 2011. *Agriculture National Geographic.* [Online] Available at: https://www.nationalgeographic.org/encyclopedia/agriculture/

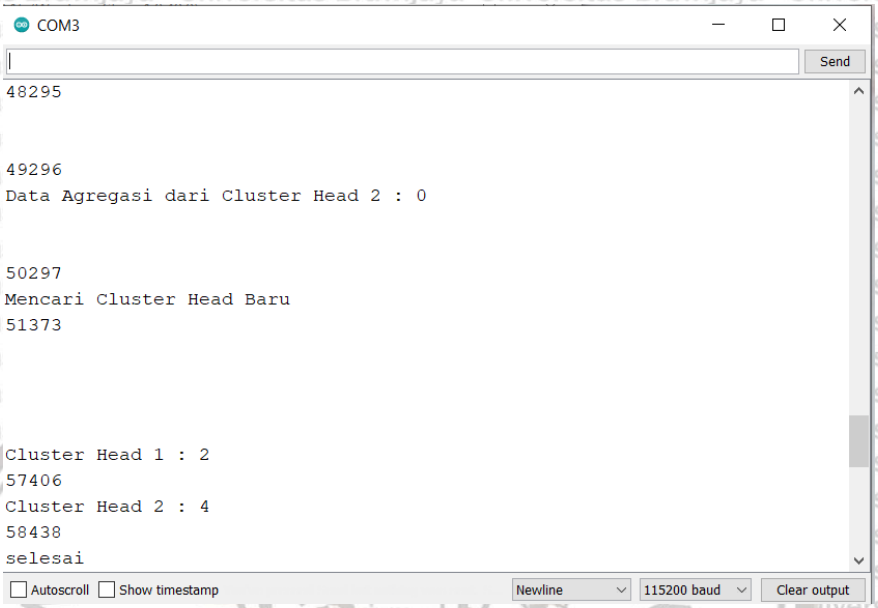
Sohraby, K., Minoli, D. & Znati, T., 2007. *Wireless Sensor Network: Technology, Protocols, and Applications.* New Jersey: John Wiley & Sons, Inc., Hoboken.

Sumiharto, R., Ilma, R. & R., 2019. Metode Routing Protocol LEACH pada Jaringan Sensor Nirkabel Studi Kasus Sistem Pemantauan Suhu dan Kelembaban Udara. *Indonesia Journal of Electronics and Instrumentation Systems ,* Volume 9, pp. 87-96.

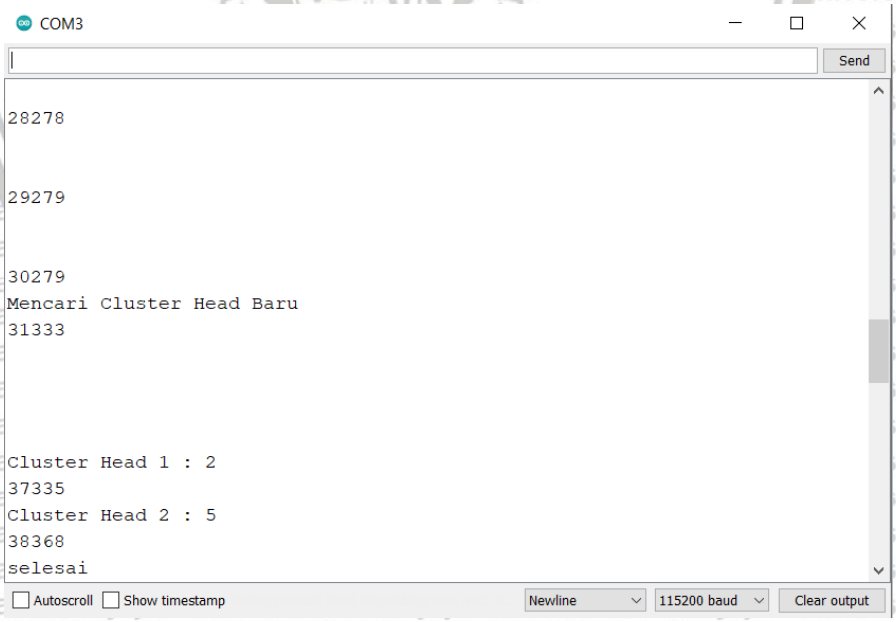
Vani, P. & Rao, K. R., 2016. Measurement and Monitoring of Soil Moisture using Cloud IoT and Android System. *Indian Journal of Science and Technology,* Volume 9.

LAMPIRAN A GAMBAR PENGUJIAN *RESPONSE TIME*

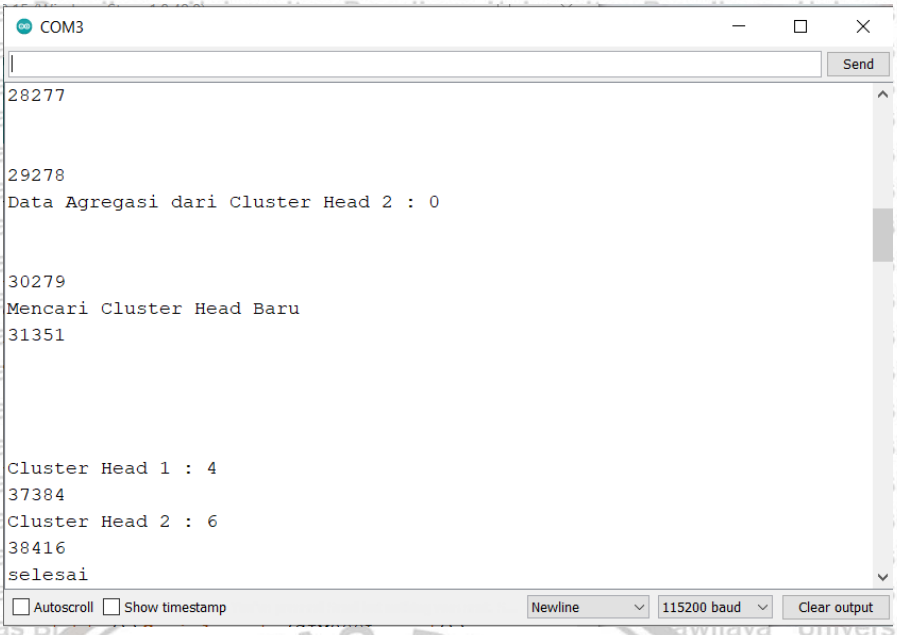
Pada lampiran ini merupakan gambar dari percobaan pengujian dari *response time* untuk mencari *cluster head* baru. Pengujian dilakukan sebanyak 10 kali percobaan, berikut gambar dari pengujian



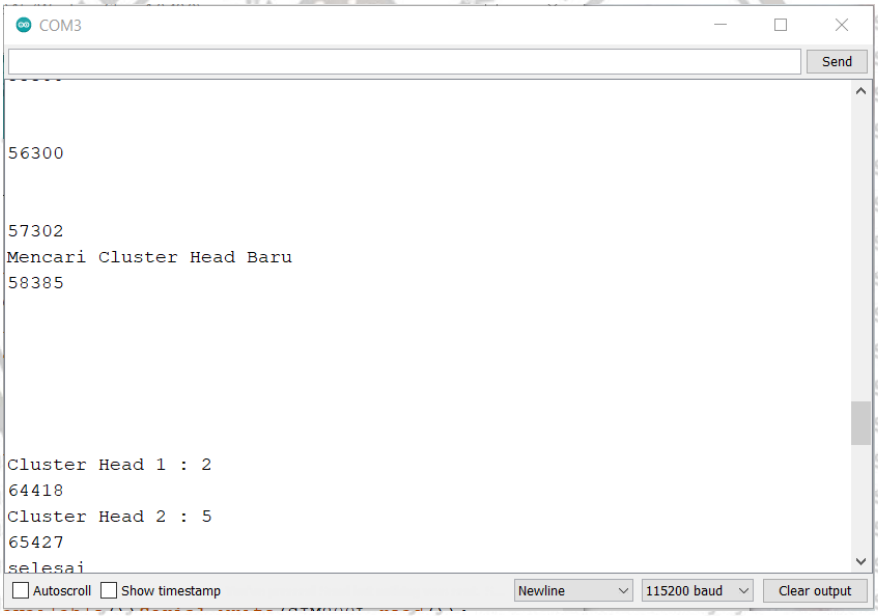
Gambar A. 1 Pengujian *Response Time* untuk mencari *cluster head* baru 1



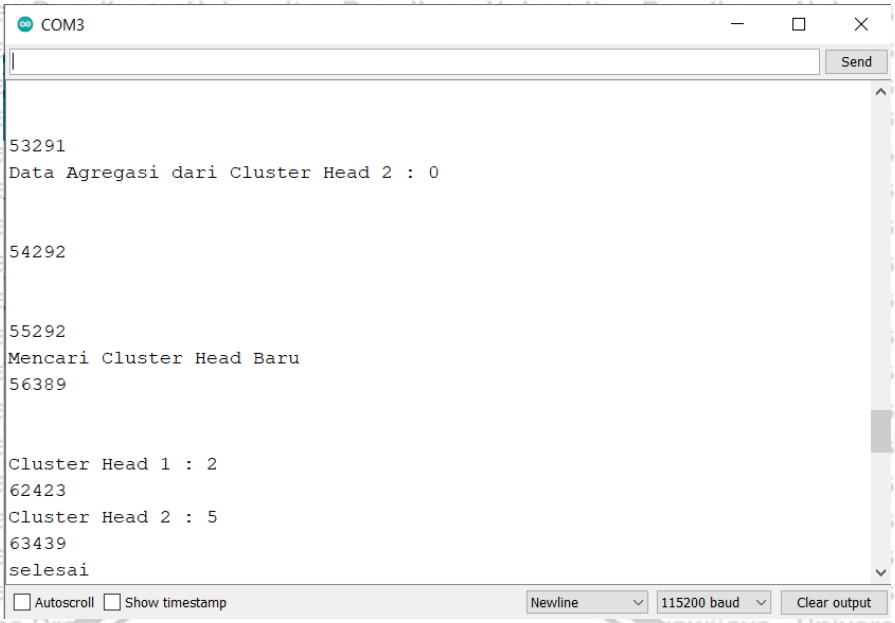
Gambar A. 2 Pengujian *Response Time* untuk mencari *cluster head* baru 2



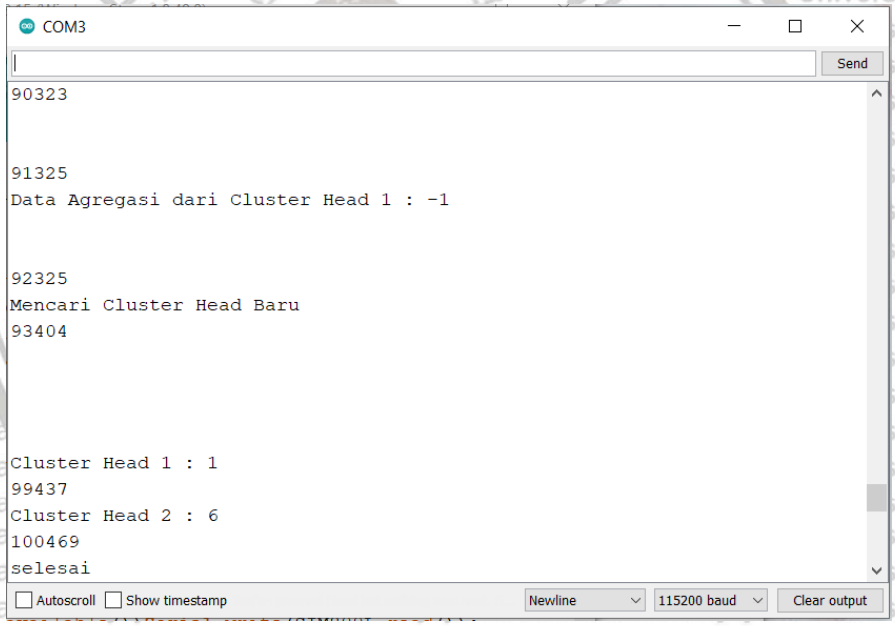
Gambar A. 3 Pengujian *Response Time* untuk mencari *cluster head* baru 3



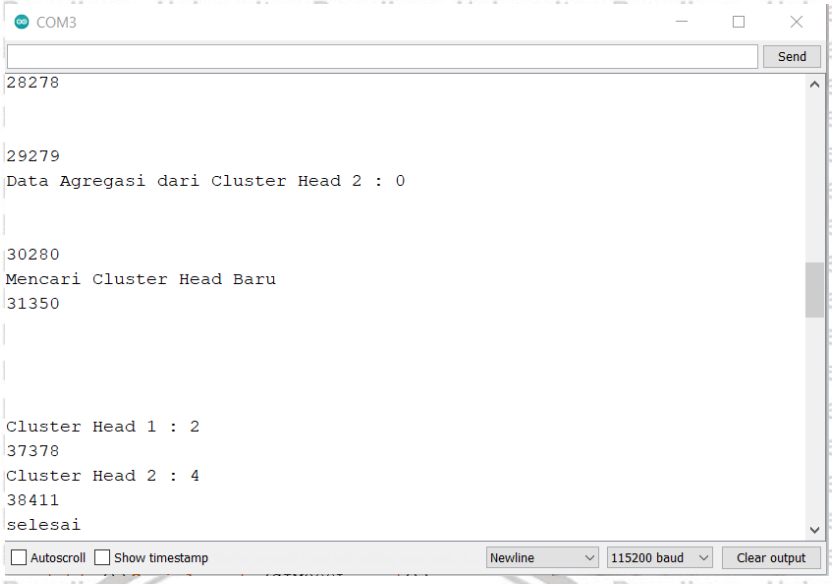
Gambar A. 4 Pengujian *Response Time* untuk mencari *cluster head* baru 4



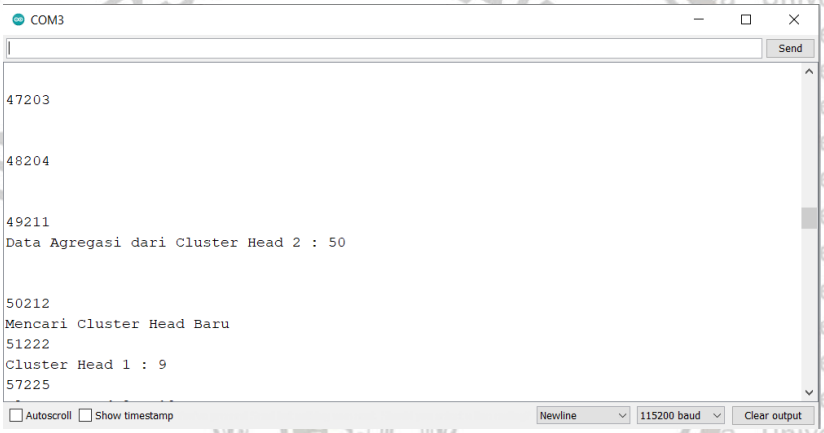
Gambar A. 5 Pengujian *Response Time* untuk mencari *cluster head* baru 5



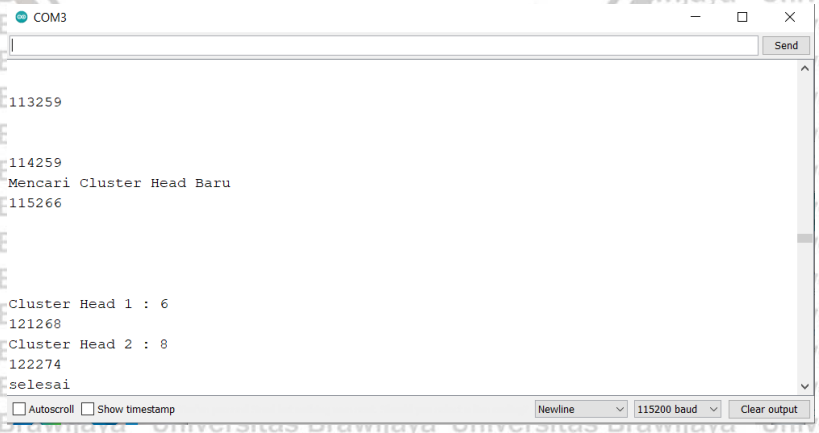
Gambar A. 6 Pengujian *Response Time* untuk mencari *cluster head* baru 6



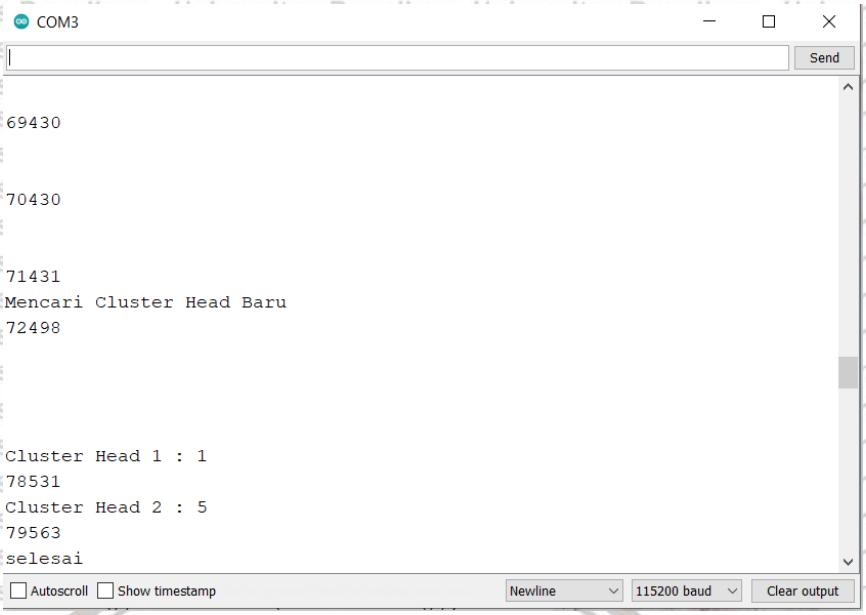
Gambar A. 7 Pengujian *Response Time* untuk mencari *cluster head* baru 7



Gambar A. 8 Pengujian *Response Time* untuk mencari *cluster head* baru 8



Gambar A. 9 Pengujian *Response Time* untuk mencari *cluster head* baru 9



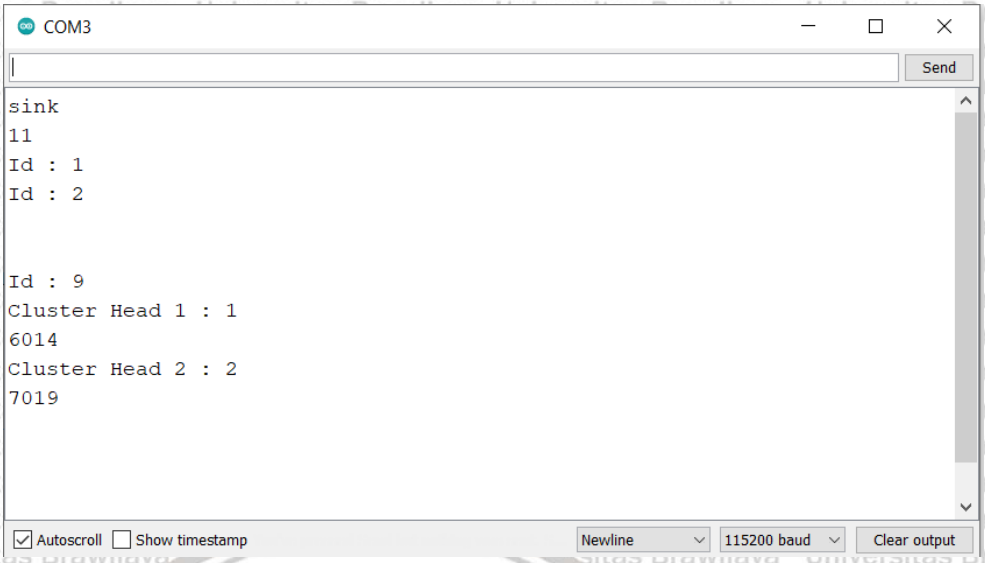
Gambar A. 10 Pengujian *Response Time* untuk mencari *cluster head* baru 10

LAMPIRAN B GAMBAR PENGUJIAN JARAK NODE

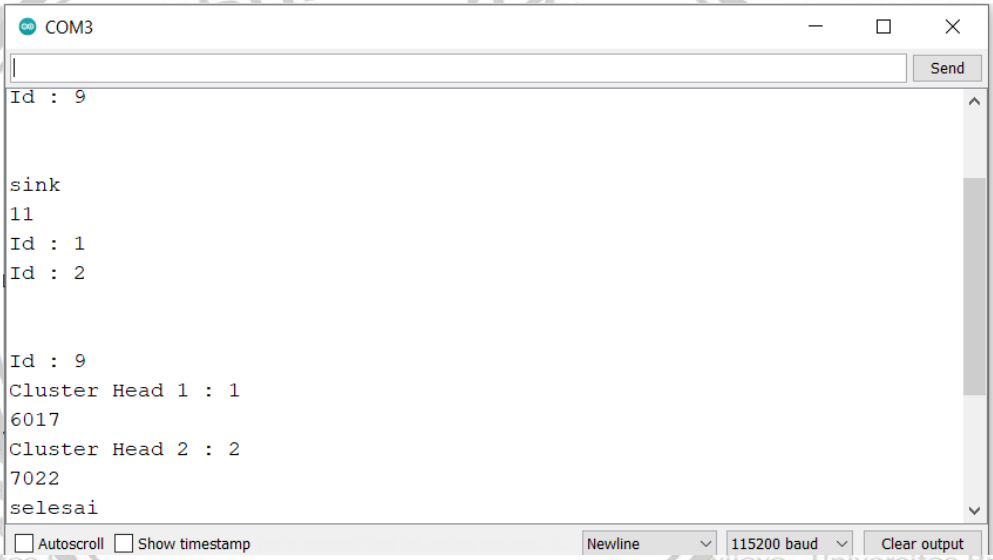
Pada lampiran ini merupakan gambar dari percobaan pengujian dari jarak node. Pengujian dilakukan dengan beberapa jarak yang berbeda, yaitu 1 meter, 3 meter, 5 meter, 10 meter, 25 meter, 50 meter, 100 meter dan 150 meter. Berikut gambar dari pengujian:

Gambar B. 1 Pengujian jarak node pada jarak 1 Meter

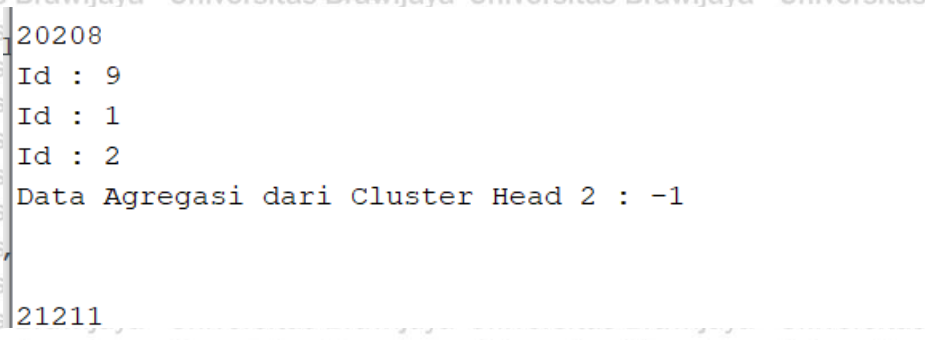
Gambar B. 2 Pengujian jarak node pada jarak 3 Meter



Gambar B. 3 Pengujian jarak node pada jarak 5 Meter



Gambar B. 4 Pengujian jarak node pada jarak 10 Meter



Gambar B. 5 Pengujian jarak node pada jarak 25 Meter

```
28103


29104
Id : 2
Id : 1


30104
Id : 9
```

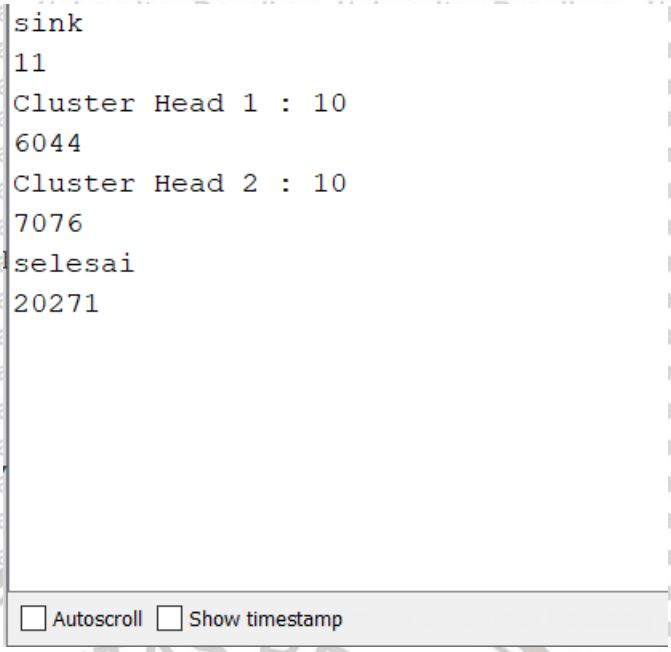
Gambar B. 6 Pengujian jarak node pada jarak 50 Meter

```
124623


sink
11
Cluster Head 1 : 10
6044
Cluster Head 2 : 10
7076
selesai
20271

Autoscroll  Show timestamp
```

Gambar B. 7 Pengujian jarak node pada jarak 100 Meter



Gambar B. 8 Pengujian jarak node pada jarak 150 Meter